PISCES II: POISSON AND CONTINUITY EQUATION SOLVER

By

Mark R. Pinto, Conor S. Rafferty, and
Robert W. Dutton

September 1984

Prepared under
U.S. Army Research Office
Contract No. DAAG-29-83-K-0125

PISCES-II

User's Manual

Mark R. Pinto

Conor S. Rafferty

Robert W. Dutton

Stanford Electronics Laboratories
Department of Electrical Engineering
Stanford University, Stanford Ca 94305.

CONTENTS

Chapter 1 **Introduction** 1

Chapter 2 **Physical Models** 2

Chapter 3 **Numerical Methods** 10

Chapter 4 **Grid in PISCES-II** 19

Chapter 5 **Examples** 27

Appendix 1 **PISCES-II input** A.1

Appendix 2 **IGGI input** A.64

Appendix 3 **Programming notes** A.72

ACKNOWLEDGEMENT

The authors are pleased to acknowledge the intellectual stimulation, encouragement and support of Dr. Wolfgang Fichtner and Dr. Donald Rose† of AT&T Bell Laboratories. A number of ideas for some of the numerical algorithms within PISCES-II were motivated by the course offered by Don Rose (CS238) during a technical leave at Stanford University in the Winter quarter of 1982-83.

Several people have contributed to early versions of PISCES. The doctoral thesis of Craig Price laid the initial framework, including the development of PISCES-I, a single carrier solver. Additional contributions to PISCES-I were made by Professor Ronald Lomax of the University of Michigan and Dr. Soo Young Oh of Hewlett-Packard. The authors also acknowledge the significant contributions of Mark Law to PISCES-II.

The continued support of Army Research Office, contract DAAG29-83-K-0125 is gratefully acknowledged. One of the authors (MRP) acknowledges financial support through AT&T Bell Laboratories. Additional industrial grants and collaboration with Hewlett-Packard, Tektronix, Toshiba and Texas Instruments have helped greatly. The Semiconductor Research Corporation is currently supporting on-going efforts to extend PISCES-II for compound semiconductor applications through contract 84-02-047.

† now at Duke University

Chapter 1

Introduction

PISCES-II is a two-dimensional, two-carrier semiconductor device modeling program which simulates the electrical behavior of devices under either steady-state or transient conditions. PISCES-II can analyze physical structures with completely arbitrary geometries (non-planar as well as planar) with general doping profiles, obtained either from analytical functions or SUPREM-III [1]. A wide variety of materials and physical models are incorporated and are user-accessible. The program supports non-uniform triangular grids which can be specified through input cards or by using an interactive grid generation pre-processor (IGGI). Further, PISCES-II itself can refine the specified grid during the solution process. A menu-driven post-processor is available as well for plotting solution data.

The following is a user's manual for PISCES-II and is organized into five chapters and three appendices. After this introduction (chapter 1), the physical equations and models will be described in chapter 2. The numerical methods and grid generation procedures will be outlined in chapters 3 and 4 respectively. Chapter 5 contains several representative examples. Appendix A is a comprehensive description of the input language and cards. Appendix B discusses the interactive grid generator, and in Appendix C some guidelines are given for installing PISCES-II.

1. References

[1] Charles P. Ho, Stephen E. Hansen, *SUPREM III - A program for integrated circuit process modeling and simulation*, Stanford University Technical Report SEL 83-001 (July 1983).

Chapter 2

Physical Description

1. Basic equations

The electrical behavior of semiconductor devices is governed by Poisson's equation

$$\epsilon \nabla^2 \psi = -q(p - n + N_D^+ - N_A^-) - \rho_F \tag{2.1}$$

and the continuity equations for electrons and holes

$$\frac{\partial n}{\partial t} = \frac{1}{q} \nabla \cdot J_n - U_n \tag{2.2}$$

$$\frac{\partial p}{\partial t} = -\frac{1}{q} \nabla \cdot J_p - U_p \tag{2.3}$$

The primary function of PISCES-II is to solve these three partial differential equations self-consistently for the electric potential, ψ, and the electron and hole concentrations, n and p respectively. Throughout PISCES-II, ψ is always defined as the intrinsic Fermi potential, i.e. $\psi = \psi_i$. Carrier statistics will be discussed in section 2. N_D^+ and N_A^- are the ionized impurity concentrations and ρ_F is a fixed charge density which may be present in insulating materials.

From Boltzmann transport theory, J_n and J_p in (2.2) and (2.3) can be written as functions of electrostatic potential, ψ, and the quasi-fermi levels for electrons and holes, ϕ_n and ϕ_p

$$J_n = -q \mu_n n \nabla \phi_n \tag{2.4a}$$

$$J_p = -q \mu_p p \nabla \phi_p \tag{2.5a}$$

or alternatively, J_n and J_p can be written as functions of ψ, n and p, consisting of drift and diffusion components:

$$J_n = q \mu_n E_n n + q D_n \nabla n \tag{2.4b}$$

$$J_p = q \mu_p E_p p - q D_p \nabla p \tag{2.5b}$$

where μ_n and μ_p are the electron and hole mobilities and D_n and D_p are the electron and hole diffusivities (see sections 2 and 3). Neglecting the effects of band-gap narrowing and assuming Boltzmann carrier statistics (see section 2),

$$E_n = E_p = E = -\nabla \psi$$

U_n and U_p in (2.2) and (2.3) represent net electron and hole recombination respectively. Currently PISCES-II supports both Shockley-Reed-Hall and Auger recombination, i.e. $U = U_n = U_p = U_{SRH} + U_{Auger}$ where

$$U_{SRH} = \frac{pn - n_{ie}^2}{\tau_p \left[n + n_{ie} \exp\left(\frac{E_t - E_i}{kT} \right) \right] + \tau_n \left[p + n_{ie} \exp\left(\frac{E_i - E_t}{kT} \right) \right]} \tag{2.6}$$

$$U_{Auger} = c_n \left(pn^2 - nn_{ie}^2 \right) + c_p \left(np^2 - pn_{ie}^2 \right) \tag{2.7}$$

In the above, c_n and c_p are specified constants, n_{ie} is the effective intrinsic concentration, E_i is the intrinsic Fermi energy, E_t is the trap energy level and τ_n and τ_p are the electron and hole lifetimes which may be concentration dependent as follows [1]

$$\tau_n(x,y) = \frac{\tau_{n0}}{1 + N(x,y)/N_{SRH-n}} \tag{2.8}$$

$$\tau_p(x,y) = \frac{\tau_{p0}}{1 + N(x,y)/N_{SRH-p}} \tag{2.9}$$

where $N(x,y)$ is the local (total) impurity concentration and N_{SRH-n}, N_{SRH-p}, τ_{n0} and τ_{p0} are specified constants.

2. Carrier statistics

The electron and hole concentrations in semiconductors are defined by Fermi-Dirac distributions and a parabolic density of states, which, when integrated, yield

$$n = N_C F_{1/2}\left\{ \frac{1}{kT} \left[E_{Fn} - E_C \right] \right\} \tag{2.10}$$

$$p = N_V F_{1/2}\left\{ \frac{1}{kT} \left[E_V - E_{Fp} \right] \right\} \tag{2.11}$$

where N_C and N_V are the effective density of states in the conduction and valence bands, E_C and E_V are the conduction and valence band energies, and E_{Fn} and E_{Fp} are the electron and hole Fermi energies, i.e. $E_{Fn} = -q\phi_n$ and $E_{Fp} = -q\phi_p$. The Fermi-Dirac integral of order one-half is defined as

$$F_{1/2}(\eta_F) = \frac{2}{\sqrt{\pi}} \int_0^\infty \frac{\eta^{1/2}}{1 + \exp(\eta - \eta_F)} d\eta \tag{2.12}$$

For the range of operation of most semiconductor devices, (2.10) and (2.11) can be simplified using Boltzmann statistics

$$n \approx N_C \exp\left[\frac{1}{kT} \left(E_{Fn} - E_C \right) \right] = n_{ie} \exp\left[\frac{q}{kT} \left(\psi - \phi_n \right) \right] \tag{2.13}$$

$$p \approx N_V \exp\left[\frac{1}{kT}(E_V - E_{Fp})\right] = n_{ie} \exp\left[\frac{q}{kT}(\phi_p - \psi)\right] \tag{2.14}$$

where, neglecting band-gap narrowing, the intrinsic carrier concentration is

$$n_{ie}(T) = n_i(T) = \sqrt{N_C N_V}\, e^{-E_g/2kT} \tag{2.15}$$

E_g is the band-gap energy of the semiconductor, i.e. $E_g = E_C - E_V$. The band-gap and density of states have temperature dependencies as follows [2]

$$E_g(T) = E_g(0) - \frac{\alpha T^2}{T+\beta} \tag{2.16}$$

$$= E_g(300) + \alpha\left[\frac{300^2}{300+\beta} - \frac{T^2}{T+\beta}\right]$$

$$N_C(T) = 2\left(\frac{2\pi m_{de} kT}{h^2}\right)^{3/2} M_C = \left(\frac{T}{300}\right)^{3/2} N_C(300) \tag{2.17}$$

$$N_V(T) = 2\left(\frac{2\pi m_{dh} kT}{h^2}\right)^{3/2} = \left(\frac{T}{300}\right)^{3/2} N_V(300) \tag{2.18}$$

Note that

$$-q\psi = E_c - \frac{E_g}{2} - \frac{kT}{2q}\ln\frac{N_c}{N_v} \tag{2.19}$$

In PISCES-II, band-gap narrowing effects due to heavy doping are included as spatial variations in the intrinsic concentration [3],

$$n_{ie}(x,y) = n_i \exp\left\{\frac{9\times10^{-3} q}{2kT}\left[\ln\frac{N(x,y)}{10^{17}} + \sqrt{\left(\ln\frac{N(x,y)}{10^{17}}\right)^2 + \frac{1}{2}}\right]\right\} \tag{2.20}$$

The spatial dependence of n_{ie} results in an adjustment to the electric field terms in the transport equations (2.4b) and (2.5b), obtained by substitution of (2.13) and (2.14) into (2.4a) and (2.5a)

$$E_n = -\nabla\left(\psi + \frac{kT}{q}\ln n_{ie}\right) \tag{2.21}$$

$$E_p = -\nabla\left(\psi - \frac{kT}{q}\ln n_{ie}\right) \tag{2.22}$$

As an aside, we note that PISCES-II† actually implements Fermi-Dirac statistics in a form quite similar to Boltzmann statistics after Yu [4]. The form of (2.13) and (2.14) is adjusted by introducing degeneracy factors γ_n and γ_p

† Fermi-Dirac statistics have not been implemented in version II-A.

where

$$\gamma_n = F_{1/2}\left[\frac{1}{kT}(E_{Fn}-E_C)\right]\exp\left[\frac{-1}{kT}(E_{Fn}-E_C)\right] \tag{2.23}$$

$$\gamma_p = F_{1/2}\left[\frac{1}{kT}(E_V-E_{Fp})\right]\exp\left[\frac{-1}{kT}(E_V-E_{Fp})\right] \tag{2.24}$$

so (2.13) and (2.14) generalize to

$$n = \gamma_n n_{ie} e^{\psi-\phi_n} \tag{2.25}$$

$$p = \gamma_p n_{ie} e^{\phi_p-\psi} \tag{2.26}$$

where $\gamma_n = \gamma_p = 1$ for Boltzmann statistics, but are less than 1 for Fermi-Dirac statistics.

With regard to the correlation between the mobilities and diffusivities in (2.4b) and (2.5b), we simply note that by assuming Boltzmann statistics, the Einstein relationship has been tacitly assumed, i.e.

$$D_n = \frac{kT}{q}\mu_n \tag{2.27}$$

$$D_p = \frac{kT}{q}\mu_p \tag{2.28}$$

However, using Fermi-Dirac statistics,

$$D_n = \left(\frac{kT}{q}\mu_n\right)F_{1/2}\left\{\frac{q}{kT}\left[E_{Fn}-E_C\right]\right\} \Big/ F_{-1/2}\left\{\frac{q}{kT}\left[E_{Fn}-E_C\right]\right\} \tag{2.29}$$

$$D_p = \left(\frac{kT}{q}\mu_p\right)F_{1/2}\left\{\frac{q}{kT}\left[E_V-E_{Fp}\right]\right\} \Big/ F_{-1/2}\left\{\frac{q}{kT}\left[E_V-E_{Fp}\right]\right\} \tag{2.30}$$

Poisson's equation (2.1) includes the ionized impurity concentrations N_D^+ and N_A^- in the expression for space charge. Although for most practical cases, full impurity ionization may be assumed, i.e., $N_D^+ = N_D$ and $N_A^+ = N_A$, PISCES-II† can treat impurity freeze-out [5] using Fermi-Dirac statistics with appropriate degeneracy factors for the conduction and valence bands g_c and g_v

$$N_D^+ = \frac{N_D}{1 + g_c\exp\left[-\dfrac{q\phi_n+E_D}{kT}\right]} \tag{2.31}$$

$$N_A^- = \frac{N_A}{1 + g_v\exp\left[\dfrac{q\phi_p+E_A}{kT}\right]} \tag{2.32}$$

† Incomplete-ionization has not been implemented in version II-A.

where E_D and E_A are the donor and acceptor energy levels respectively.

3. Mobility models

The carrier mobilities μ_n and μ_p account for scattering mechanisms in electrical transport. PISCES-II contains several options with regard to mobility.

The simplest alternative is to choose constant mobilities μ_{n0} and μ_{p0} for electrons and holes throughout the device, dependent on the semiconductor material. With little additional overhead, the effect of impurity scattering can be included by using local low-field mobilities, i.e. $\mu_0(x,y) = \mu_0[N(x,y)]$, where $N(x,y)$ is the total impurity concentration as a function of position in the device. For Si and GaAs, PISCES-II has a table of low-field mobilities versus total impurity concentration for both electrons and holes at 300K. An empirical function [6,7] can be used to obtain low-field mobilities for silicon at temperatures other than 300K.

Along oxide-semiconductor interfaces, the carrier mobilities can be substantially lower than in the bulk of the semiconductor due to surface scattering. PISCES-II accounts for this difference by including a degradation factor g_{surf} in the low-field mobility μ_0 used for calculating current along the surface only, i.e.

$$\mu_0(surface) = g_{surf}\,\mu_0(bulk) \tag{2.33}$$

where $0 < g_{surf} \leq 1$.

Using analytic expressions for the drift velocity v_d as a function of the electric-field E in the direction of current flow and defining $\mu(E) = v_d(E)/E$, field-dependent models for mobility can be derived and have been implemented in PISCES-II for Si and GaAs. For Si [6],

$$\mu(E) = \left[\frac{1}{1 + \left(\frac{\mu_0 E}{v_{sat}}\right)^{\beta}}\right]^{1/\beta} \mu_0 \tag{2.34}$$

where v_{sat} is the saturation velocity, β is a constant (usually 1 or 2) and μ_0 is the low-field mobility, which may include the scattering mechanisms discussed above. For GaAs [7],

$$\mu(E) = \frac{\mu_0 + \frac{v_{sat}}{E}\left(\frac{E}{E_0}\right)^4}{1 + \left(\frac{E}{E_0}\right)^4} \tag{2.35}$$

where μ_0 and v_{sat} have the same meaning as above and E_0 is a constant. PISCES-II includes empirical temperature-dependent models for saturation velocity in both Si and GaAs. For Si [2],

$$v_{sat}(T) = \frac{2.4\times10^7}{1 + 0.8\left(\frac{T}{600}\right)} \tag{2.36}$$

and for GaAs [9],

$$v_{sat}(T) = 11.3\times10^6 - 1.2\times10^4\, T \tag{2.37}$$

where v_{sat} is in cm/sec and T is the temperature in Kelvin units.

4. Boundary conditions

PISCES-II supports four types of boundary conditions: ohmic contacts, Schottky contacts, insulator contacts and Neumann (reflective) boundaries.

Ohmic contacts are implemented as simple Dirichlet boundary conditions, where the surface potential and electron and hole concentrations (ψ_s, n_s, p_s) are fixed. The minority and majority carrier quasi-fermi potentials are equal and are set to the applied bias of that electrode, i.e. $\phi_n = \phi_p = V_{applied}$. The potential, ψ_s, is fixed at a value consistent with zero space charge, i.e.

$$n_s + N_A^- = p_s + N_D^+ \tag{2.38}$$

Substituting (2.10) and (2.11) for n_s and p_s, (2.38) can be solved for ψ_s and hence n_s and p_s, since ϕ_n and ϕ_p are known. If Boltzmann statistics are used, substitution of (2.13) and (2.14) into (2.38) will yield

$$n_s = \frac{1}{2}\left[\left(N_D^+ - N_A^-\right) + \sqrt{\left(N_D^+ - N_A^-\right)^2 + 4n_{ie}^2}\right] \tag{2.39}$$

$$p_s = \frac{n_{ie}^2}{n_s} \tag{2.40}$$

where

$$\psi_s = \phi_n + \frac{kT}{q}\ln\frac{n_s}{n_{ie}} = \phi_p - \frac{kT}{q}\ln\frac{p_s}{n_{ie}} \tag{2.41}$$

Schottky contacts to the semiconductor are defined by a work function of the electrode metal and an optional surface recombination velocity. The surface potential at a Schottky contact is defined by

$$\psi_s = \chi + \frac{E_g}{2q} + \frac{kT}{2q}\ln\frac{N_C}{N_V} - \phi_m + V_{applied} \tag{2.42}$$

where χ is the electron affinity of the semiconductor and ϕ_m is the work function of the metal. If a finite surface recombination velocity is imposed, ϕ_n and ϕ_p are no longer equal to $V_{applied}$ and instead are defined by a current boundary condition at the surface [10]

$$J_{sn} = qv_{sn}\left(n_s - n_{eq}\right) \tag{2.43}$$

$$J_{sp} = qv_{sp}\left(p_s - p_{eq}\right) \tag{2.44}$$

where J_{sn} and J_{sp} are the electron and hole currents at the contact, n_s and p_s are the actual surface electron and hole concentrations and n_{eq} and p_{eq} are the equilibrium electron and hole concentrations assuming infinite surface recombination velocities ($\phi_n = \phi_p = V_{applied}$). The actual surface recombination velocities for electrons and holes, v_{sn} and v_{sp}, are given by

$$v_{sn} = \frac{A_n^{**}T^2}{qN_C} \tag{2.45}$$

$$v_{sp} = \frac{A_p^{**}T^2}{qN_V} \tag{2.46}$$

where A_n^{**} and A_p^{**} are the effective Richardson constants for electrons and holes, which take into account quantum mechanical reflection and tunneling.

Insulating contacts generally have a work function, dictating a value for ψ_s similar to (2.38). The electron and hole concentrations within the insulator and at the insulating contact are forced to be zero, i.e. $n_s = p_s = 0$.

Along the outer (non-contacted) edges of devices to be simulated, homogeneous (reflecting) Neumann boundary conditions are imposed so that current only flows out of the device through the contacts. Additionally, in the absence of surface charge along such edges, the normal component of the electric field goes to zero, i.e. $\hat{n}\cdot\vec{\nabla}\psi = 0$. In a similar fashion, current is not permitted to flow from the semiconductor into an insulating region, and at the interface between two different materials, the difference between the normal components of the respective electric displacements must be equal to any surface charge ρ_s present along the interface:

$$\hat{n}\cdot\epsilon_1\vec{\nabla}\psi_1 - \hat{n}\cdot\epsilon_2\vec{\nabla}\psi_2 = \rho_s \tag{2.47}$$

5. Adjustments for simple Poisson or single carrier analysis

In order to simplify the analysis of particular devices (and hence save a large amount of computational time), it may be desirable to only solve Poisson's equation and perhaps only one continuity equation (or just Poisson alone). Typical candidate problems for such a treatment would be majority carrier devices such as MOSFETs, JFETs and MESFETs or any devices where all junctions are not forward biased and currents may not be required (CCDs, capacitors, etc.). If a carrier is not explicitly solved for, it still must have a value consistent with the electrostatic potential throughout the device. PISCES-II therefore chooses appropriate quasi-fermi potentials for these carriers. The

quasi-fermi levels are chosen to be locally constant and change only at metallurgical junctions within the device, thereby contributing no current component. For example, if holes are not being solved for, then in a p-type region, ϕ_p is set to the local bias voltage. In any n-type region, ϕ_p is set to the lowest applied (semiconductor) potential in the device, so that excess holes do not occur as minority carriers. Similarly, if electrons are not explicitly solved for, ϕ_n is set to the local bias in any n-type region and to the maximum (semiconductor) bias in the system for all p-type regions.

6. References

[1] D. J. Roulston, N. D. Arora and S. G. Chamberlain, "Modeling and measurement of minority-carrier lifetime versus doping in diffused layers of n^+-p silicon diodes," *IEEE Trans. on Electron Devices*, ED-29, Feb. 1982, pp. 284-291.

[2] S. M. Sze, *Physics of Semiconductor Devices*, 2nd ed., John Wiley and Sons, New York, 1982.

[3] J. W. Slotboom, "The *pn* product in silicon, " *Solid State Electronics*, 20, 1977, pp. 279-283.

[4] Z. Yu, Ph.D. dissertation, Stanford Electronic Laboratories, Stanford University (to be published).

[5] R. C. Jaeger and F. H. Gaensslen, "Simulation of impurity freezeout through numerical solution of Poisson's equation and application to MOS device behavior," *IEEE Trans. on Electron Devices*, ED-27, May 1980, pp. 914-920.

[6] D. M. Caughey and R. E. Thomas, "Carrier mobilities in silicon empirically related to doping and field," *Proc. IEEE*, vol. 55, 1967, pp. 2192-2193.

[7] S. Selberherr, "Process and device modeling for VLSI," *Microelectron. Reliab.*, vol. 24, no. 2, pp. 225-257, 1984.

[8] J. J. Barnes, R. J. Lomax and G. I. Haddad, "Finite-element simulation of GaAs MESFET's with lateral doping profiles and sub-micron gates," *IEEE Trans. on Electron Devices*, ED-23, Sept. 1976, pp. 1042-1048.

[9] M. A. Littlejohn, J. R. Hauser and T. H. Glisson, "Velocity-field characteristics of GaAs with $\Gamma_6^c - L_6^c - X_6^c$ conduction band ordering," *J. Applied Physics*, v. 48, no. 11, Nov. 1977, pp. 4587-4590.

[10] C. R. Crowell and S. M. Sze, "Current transport in metal-semiconductor barriers," *Solid State Electronics*, 9, 1966, pp. 1035-1048.

Chapter 3

Numerical Methods

1. Introduction

Three partial differential equations (PDEs) describe the bulk behavior of semiconductor devices. The Poisson equation governs the electrostatic potential, and the electron and hole continuity equations govern the carrier concentrations. For reference, they are listed below:

$$\vec{\nabla}\cdot(\epsilon\vec{\nabla}\psi) = -q\,(p-n+N) \qquad (3.1) \qquad \text{Poisson's equation}$$

$$\vec{\nabla}\cdot\vec{J_n} = qU(n,p) + q\frac{\partial p}{\partial t} \qquad (3.2) \qquad \text{Electron continuity}$$

$$\vec{\nabla}\cdot\vec{J_p} = -qU(n,p) + q\frac{\partial p}{\partial t} \qquad (3.3) \qquad \text{Hole continuity}$$

where

$$\vec{J_n} = q\mu_n(-n\vec{\nabla}\psi + \frac{kT}{q}\vec{\nabla}n)$$

and

$$\vec{J_p} = q\mu_p(-p\vec{\nabla}\psi - \frac{kT}{q}\vec{\nabla}p)$$

These differential equations are discretized as described in the next section. The resulting set of algebraic equations is coupled and nonlinear. Consequently there is no method to solve the equations in one direct step. Instead, solutions must be obtained by a nonlinear iteration method, starting from some initial guess. The various solution methods are detailed in Section 3, and the choice of initial guess is explained in Section 4.

2. Discretization

To solve the device equations on a computer, they must be discretized on a simulation grid. That is, the continuous functions of the PDEs are represented by vectors of function values at the nodes, and the differential operators are replaced by suitable difference operators. Instead of solving for three unknown functions, PISCES-II solves for $3N$ unknown real numbers, where N is the number of grid points.

The key to discretizing the differential operators on a general triangular grid is the box method (Varga[1]). Each equation is integrated over a small polygon enclosing each node, yielding $3N$ nonlinear algebraic equations for the unknown potentials and concentrations. The integration equates the flux into the polygon with the sources and sinks inside it, so that conservation of current and electric flux are built into the solution. The integrals involved are performed on a triangle-by-triangle basis, leading to a simple and elegant way of handling general surfaces and boundary conditions.

In the case of the continuity equation, the carrier fluxes must be evaluated with care; the classic finite difference formulas are modified as first demonstrated by Scharfetter and Gummel in 1969[2]. For further details on the discretization of the device equations, and particularly for a discussion of the impact of obtuse triangles in the grid, see Price[3]. From the user's point of view, the discretization is completely automatic and no intervention is required.

3. Solution Methods

The discretization of the semiconductor device equations gives rise to a set of coupled nonlinear algebraic equations, which must be solved by a nonlinear iteration method. Two approaches are widely used, Gummel's method and Newton's method. Either approach involves solving several large linear systems, of order 1-3 times the number of grid points, depending on the number of carriers being solved for. The total cost of a simulation is the product of the number of matrix solutions and the cost of each solution. The objective of the various methods detailed below is to minimize one component or the other of that cost.

Given a particular device and a range of operation, one solution method or another may be most suitable. No one method is optimal in all cases. At zero bias, for instance, a Poisson solution alone is sufficient. For MOSFET characteristics, only one carrier need be solved for. In bipolar simulations both carriers are needed, and the most efficient solution method depends on the operating condition. The solution method and the number of carriers to be solved are specified at symbolic time, that is, at the time the program generates a map of the matrix. The various parameters, acceleration factors and iteration limits are specified later on the method card, and if none are given reasonable defaults are chosen.

Several ideas are common to all methods of solving the equations.

The nonlinear iteration usually converges either at a linear rate or at a quadratic rate. In the former, the error decreases by about the same factor at each iteration. In a quadratic method, the error is approximately squared at each iteration, giving rapid convergence. For accurate solutions, there is a large advantage in using a quadratic method. Newton's method is quadratic, Gummel's is linear in most cases.

The error remaining at each step can be measured in two ways. The first, known as a right-hand-side norm (RHS norm), is the difference between the left and right hand sides of the equations (3.1). Since that is the quantity we wish to reduce to zero, the RHS norm is in a sense the most natural measure of the error. It is measured in C/μ for the Poisson equation and in A/μ for the continuity equations. At zero bias there is always a residual current due to numerical error. The rhs norm may be interpreted as the size of this current. As an alternative, the error may be measured as the size of the updates at each iteration. The updates are the unknowns "x" at each step, so this is called an X norm. The potential updates are measured in kT/q. Updates to the carrier concentrations are measured relative to the previous value at the point. In all cases the iteration will terminate when one of three conditions is satisfied :

- The X norm falls below a certain tolerance. The default tolerance is $10^{-5}kT/q$ for the potentials and 10^{-5} relative change in the concentrations.
- The RHS norm falls below a certain tolerance. The tolerance is $10^{-26}C/\mu$ for the potential norm and $5\times10^{-18}A/\mu$ for the carrier concentrations. These numbers are the maximum acceptable divergences of electric and carrier fluxes.
- A fixed number of loops is executed. This criterion is used only for special purposes.

When there is a "reasonable" amount of current flowing in the device, the norms are well-behaved. The X norm is usually the most suitable, because as the current level increases, so does numerical error in the continuity equation, and the absolute criterion on the RHS becomes harder to satisfy. The *relative* error remains the same, however. At low current levels, a physical problem arises. The exact value of the smallest minority concentrations has little influence on the current, and these small concentrations are almost indeterminate. The size of the updates may remain as large as 1% of the carrier concentrations long after convergence has been reached. Therefore at low current, it is advisable to increase the tolerance to 10^{-2} for the X norm, or to use the RHS norm.

Given an outer nonlinear iteration, the resulting linearized system can be solved either by a direct method (Gaussian elimination) or by an *inner* (linear) iteration method. In general the direct method is more stable, but the cost of the inner iteration increases less rapidly with the number of grid points. There

is a tradeoff between stability and speed which must be considered in choosing an appropriate method.

A common theme of the inner iteration loop in this situation is to maintain quadratic convergence in the outer loop, while doing as little work as possible in the inner loop. If the inner loop exits too early, the outer loop will converge only linearly. If the inner loop solves to more than the required accuracy, the extra work done is wasted. It is known [4] that an effective way of meeting both demands is to ensure that the error in the inner loop is smaller by some amount than the *expected* error in the outer loop. This criterion is used to terminate convergence of inner loops.

We now consider the different possibilities available.

3.1. Decoupled solution (Gummel's method)

In Gummel's method, the equations are solved sequentially. The Poisson equation is solved assuming fixed quasi-fermi levels; since the Poisson equation is nonlinear it is itself solved by an inner Newton loop. Then the new potential is substituted into the continuity equations, which are linear and can be solved directly. The new carrier concentrations are substituted back into the charge term of Poisson's equation and another cycle begins.

At each stage only one equation is being solved, so the matrix has N rows and N columns regardless of the number of carriers (0/1/2) being solved. This is a decoupled method; one set of variables is held fixed while another set is solved for. The success of the method depends therefore on the degree of coupling between the equations. The most important coupling is the drift term of carrier current, which is directly related to the Poisson solution. Whenever drift terms are unimportant, for instance in isolation structures, Gummel's method is suitable. When the current is drift-dominated, for example in a pure resistive structure, convergence is slow.

3.1.1. ICCG iteration

Several options are available to maximize the speed of the Gummel algorithm. The first is to use an iterative method for the innermost linear equation solver. (There are then three loops - the outer Gummel cycle, the linearization loop of the Poisson equation, and the inner loop which solves the linearized system.) Thus on each cycle of the Gummel algorithm, only one direct matrix inversion for each carrier plus one iterative solution of Poisson's equation is performed. Without any carriers (a Poisson-only solution), no Gaussian elimination is necessary. This is the fastest possible solution mode with PISCES-II and is suitable for capacitance analysis or zero bias conditions. It can sometimes be used to provide initial guesses for other methods.

The iterative method chosen in PISCES-II is incomplete Cholesky conjugate gradients (ICCG). ICCG solves the Poisson equation essentially by minimizing the energy of the electrostatic field, and is to our knowledge the fastest iterative

solver which can be used on an irregular grid. It is recommended whenever the usual Gummel algorithm is being used, particularly for large grids. Two parameters govern the behavior of the iteration, relating to the termination criterion. The defaults are chosen as 'safe' numbers - more speed can be obtained by loosening the bounds.

3.1.2. Damping

When the bias electrodes are abruptly changed by large steps (more than 1V) it is usually necessary to damp numerical ringing in the Poisson iteration. The simplest mechanism is to limit the maximum voltage change per iteration inside the device. Too tight a bound slows convergence while too large a value can allow overflow. We have chosen 0.1V, a conservative figure. For faster but slightly more risky simulations, a value of 0.3-0.7V is possible. The speed advantage is usually significant.

A more sophisticated mechanism is the Newton damping method advocated by Rose & Bank[4]. This damping scheme frequently speeds convergence by rejecting updates which would cause the error norm to increase. The default parameters controlling the damping are usually satisfactory.

3.1.3. Single-Poisson Iteration

The range of usefulness of Gummel's method can be extended to a somewhat higher current domain by limiting the number of Poisson solutions per cycle (Price[3]). This is the 'single-Poisson' solution mode. It has found some use for low current bipolar and saturation MOS applications. In this mode the convergence can be aided by over-relaxation, that is, by scaling up the update by a factor between one and two. The factor is chosen automatically according to a procedure published by Carré [5].

3.2. Coupled solution (Newton's method)

In Newton's method, all of the variables in the problem are allowed to change during each iteration, and all of the coupling between variables is taken into account. Due to this, the Newton algorithm is very stable, and the solution time is nearly independent of bias condition, even into high-level injection. The basic algorithm is a generalization of the Newton-Raphson method for the root of a single equation. It can be expressed as follows. The equations (3.1-3.3) are rewritten as

$$G_\psi(\psi, n, p) = 0$$
$$G_n(\psi, n, p) = 0$$
$$G_p(\psi, n, p) = 0$$

Given an initial guess for the unknowns at each node (ψ_0, n_0, p_0), we calculate a new update $(\Delta\psi, \Delta n, \Delta p)$ by solving the linear system

$$\begin{bmatrix} \frac{\partial G_\psi}{\partial \psi} & \frac{\partial G_\psi}{\partial n} & \frac{\partial G_\psi}{\partial p} \\ \frac{\partial G_n}{\partial \psi} & \frac{\partial G_n}{\partial n} & \frac{\partial G_n}{\partial p} \\ \frac{\partial G_p}{\partial \psi} & \frac{\partial G_p}{\partial n} & \frac{\partial G_p}{\partial p} \end{bmatrix} \begin{bmatrix} \Delta\psi \\ \Delta n \\ \Delta p \end{bmatrix} = - \begin{bmatrix} G_\psi \\ G_n \\ G_p \end{bmatrix}$$

This Jacobian matrix has 3 times as many columns and rows (or twice as many when solving for only one carrier) as the matrix for a single variable. The disadvantage of Newton's method is that for large grids, the memory and time necessary to invert it may be excessive. Typically the $3N \times 3N$ matrix takes twenty times longer to invert, and the $2N \times 2N$ seven times longer, than an $N \times N$ matrix on the same grid.

Thus the overhead per iteration is high, but the number of iterations is small, typically between three and eight. (In fact, a large number of Newton iterations is almost a guarantee that the problem is in some way ill-posed. The most frequent cause is the non-physical boundary condition set up by a depletion layer which extends beyond the bottom of the device but intersects a neutral contact). Several options are built in to minimize the overhead. The minimum degree algorithm from the Yale Sparse Matrix Package[6] is called to reorder the nodes in a way which reduces the size of the factorized matrix. A second procedure removes coupling coefficients from the matrix if they are zero for geometric reasons (along the hypotenuse of a right triangle). Both of these features are by default turned on, and would only be turned off in exceptional circumstances.

The single biggest acceleration of a Newton iteration is the Newton-Richardson method, which only refactors the Jacobian matrix when necessary. Frequently the Jacobian need only be factorized twice per bias point using Newton-Richardson, as opposed to the forty or fifty factorizations required in a decoupled approach. The decision to refactor is made on the basis of the decrease per step of the error norm. When the norm of the error falls by more than a certain criterion, the Jacobian is considered sufficiently accurate and no refactorization is done. The default criterion is set at 0.1, and can be adjusted downwards to increase stability, or upwards to increase speed. It should not be increased above 0.5, to preserve the stability of the Newton iteration in high level injection.

With these refinements, full Newton is the method of choice for solving one-carrier problems after turn-on. The solution time is typically a factor of three below the corresponding Gummel time.

For two-carrier simulations, the best choice is not so clear. Full Newton with its high overhead is rather expensive both in time and in memory. On the other hand Gummel's method becomes increasingly slow as the power level increases, and ceases to converge for on-state bipolar problems. To preserve the convergence characteristics of Newton's method while maintaining a lower overhead, two iterative solutions of the Newton matrix are available.

3.2.1. Iterative solutions of the Newton matrix

The Newton matrix is not symmetric nor is it positive definite, so the ICCG method used for the Poisson matrix can not be used as before. Instead two asymmetric methods are implemented.

The iterative methods proceed by finding a matrix $\tilde{J}$ which is close to the Jacobian J, but easier to invert. Then instead of solving

$$J\,x = -G$$

we solve instead

$$(J\tilde{J}^{-1})(\tilde{J}x) = -G$$

Since $\tilde{J}$ is close to J, the matrix $J\tilde{J}^{-1}$ is close to the identity matrix, and the resulting system is more amenable to iteration than the original matrix.

In a conjugate residual iteration, at each step a direction is chosen, and the point along that direction which minimizes the residual of the linear system is chosen as the next iterate. A new direction is then chosen, and a new cycle begins. Each direction should be orthogonal ("conjugate") to the previous directions for fastest convergence. In practice it is not possible to keep all previous directions; the latest k only are kept, where k is usually between one and ten. The method is particularly efficient for matrices which have a small number of independent eigenvalues or are close to the identity, and it is frequently used for this reason. The rate of convergence depends on the matrix $\tilde{J}$ used to 'precondition' the Jacobian. An attractive feature is that the method always seeks a descent direction, so that at worst it converges slowly but never diverges.

In the **block iterative** method, the matrix $\tilde{J}$ is chosen as the three diagonal blocks of J. Its inverse is thus three times as expensive as a single matrix inversion, considerably better than the factor of twenty for the full matrix. Depending on the bias condition, the iteration then proceeds to a solution. The iteration is much faster when the matrix is written in terms of the exponentials of the quasi-fermi levels (known as Slotboom variables) and these should be selected.

The block method is intermediate between Gummel's method and Newton's method, as it is not entirely bias-independent. We have found it to be the method of choice for bipolar simulations up to moderate level injection.

The **knot incomplete factorization** (knot ILU) method takes a different approach. The Jacobian J is reorganized as an $N \times N$ array of 3×3 micro-matrices, each of which contains all the equations for a single node. Then partial Gaussian elimination is performed on this matrix, by omitting small elements in the lower and upper triangular factors. This procedure economizes on storage, and because the equation-to-equation coupling is maintained, the method is fairly stable with respect to operating condition. The size of elements which are dropped from the factorization can be controlled; the default is to drop all elements which are not in the original matrix. Again this criterion is rather conservative, and the solution speed can be increased by retaining more of the matrix, at the risk of overflowing storage. Knot ILUdoes not seem to be significantly faster than Gaussian elimination, but it uses only half as much storage. Therefore we use it primarily to solve very large problems.

4. Initial guesses

Five types of initial guesses are used in PISCES-II. The first is simply the charge neutral assumption used to to obtain the first (equilibrium) bias point. This is the starting point of any device simulation. Any later solution with applied bias needs an initial guess of some type, obtained by modifying one or two previous solutions. With a *previous* initial guess, the solution currently loaded is used as the initial guess, modified by setting the applied bias at the contacts. In some cases a better guess can be obtained with a *local* guess. This takes the solution in memory, sets the applied bias, and changes the majority fermi levels throughout heavily doped regions to be equal to the bias applied at that region. This procedure is effective in the context of a Gummel iteration, particularly in reverse bias, but less so for a Newton method. Frequently the best initial guess is obtained by *projection*, which takes two previous solutions whose bias conditions differ only at one contact, and extrapolates to a new applied bias at that contact. This is particularly economical in generating I-V data. Finally, a special initial guess is present immediately after performing a regrid. This guess is an interpolation of the solution on a coarse grid to the new grid, and can be used to start the solution of the same bias point on the new grid. It is interesting to note that in spite of being an interpolation of an exact solution, this type of guess does not give rise to particularly fast convergence.

PISCES-II selects the projection method whenever two appropriate solutions are available, otherwise it defaults to a previous guess. In some cases the user may wish to override this choice. For instance, given two solutions of a MOSFET with 0 and 1V on the gate, it is unwise to extrapolate to 2V on the gate if the device has a threshold close to 1V. The reason is that the extrapolation will make the incorrect assumption that the change in surface potential stepping from $1 \rightarrow 2$ is the same as when stepping $0 \rightarrow 1$. In this case a previous guess using 1V on the gate would converge more rapidly. Similar situations can arise whenever a device changes operating region within one bias step.

5. Summary

A large number of solution methods have been presented, but this should not be cause for confusion. Only a small subset of the possible combinations are used in day-to-day simulations, and the default parameters rarely need adjustment. Broadly speaking, Newton's method with Gaussian elimination of the Jacobian is by far the most stable method of solution. Unfortunately it can be expensive for two-carrier simulations, both in time and memory, and iterative methods are available to speed convergence over most of the device characteristic. For low current solutions the Gummel method offers an attractive alternative to inverting the full Jacobian.

The following table indicates the choices we have found to be fastest for typical devices.

Technology	Operating condition	Method	Parameters
MOS	Subthreshold	Gummel	ICCG damped
	Linear	Direct 1-carr	
	Saturation	Direct 1-carr	
Bipolar	Linear	Block 2-carr	Slotboom scaled
	Saturation	Direct 2-carr	

References

[1] R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.

[2] D. L. Scharfetter, H. K. Gummel, *IEEE Trans. Electron Devices*, vol. ED-16, pp.64-77, 1969.

[3] C. H. Price, Ph.D Thesis, Stanford University (May 1982).

[4] D. J. Rose, R. E. Bank, *Global Approximate Newton Methods*, Numerische Mathematik, 37, 279-295(1981)

[5] B.A. Carré, *Computer Journal*, 4, 1961, p.73.

[6] S. C. Eisenstat, M. C. Gursky, M. H. Schultz, A. H. Sherman, Dep. Comput. Sci., Yale Univ., Tech. Rep. 112, 1977.

Chapter 4

Grid in PISCES-II

1. Introduction

The correct allocation of grid is a crucial issue in device simulation. The number of nodes in the grid (N_p) has a direct influence on the simulation time. The number of arithmetic operations necessary to achieve a solution is proportional to ($N_p{}^{\alpha}$) where α usually varies between 1.5 and 2. Because the different parts of a device have very different electrical behavior, it is usually necessary to allocate fine grid in some regions and coarse grid in others. As far as possible, it is desirable not to allow the fine grid to spill over into regions where it is unnecessary, in order to maintain the simulation time within reasonable bounds.

Another aspect of grid allocation is the accurate representation of small device geometries. In order to model the carrier flows correctly, the grid must be a reasonable fit to the device shape. This consideration becomes more and more important as smaller, more non-planar devices are simulated.

For the reasons stated above, PISCES-II supports a general irregular grid structure. This permits the analysis of arbitrarily shaped devices, and allows the refinement of particular regions with minimum impact on others.

2. Grid Specification

The most important disadvantage associated with a general grid structure is the difficulty of user specification, and one of our primary goals has been to minimize this effort. The main tool in this regard is the regridding mechanism, which automatically refines an initial grid wherever key variables vary rapidly. We apply the technique in a different way to most workers, in that the grid is changed between solutions rather than during the solution. We prefer this approach because we feel the overhead of repeatedly recalculating grid geometry and symbolic factorizations during every solution is unacceptable. The implementation follows the proposals in [1] quite closely.

There are two methods to specify the initial grid in PISCES-II. The first is the generation of a crude rectangular grid using input cards. The second is a stand-alone interactive grid generator which uses a graphics device to define the boundary of the simulation region and generates a coarse grid to cover it. Both methods are developed to the point where they can be, and have been, used to generate the entire grid from beginning to end. The regrid algorithm has made most of this capability redundant and direct specification is now used mainly to prepare the initial grid.

3. Rectangular mesh specification

The preceding commments notwithstanding, a distorted rectangular mesh can be a very effective solution mesh in some cases. For a planar device or a long-channel MOSFET, a rectangular grid is the method of choice. The same set of fine grid lines which follow the channel can be diverted around the junctions to provide reasonable resolution through all active areas of the device. Large aspect rectangles can be used to minimize the amount of grid allocated, and the resulting matrix has some properties which can be exploited to reduce solution time. Alternatively, a coarse rectangular grid is a suitable candidate for regridding, particularly if the device has a complicated doping profile.

Rectangular meshes are specified by a series of mesh cards, detailed in Appendix A. In order of appearance, the required input is

- mesh card, specifying rectangular mesh and number of x,y lines
- x.mesh cards
- y.mesh cards
- spread cards (optional)
- eliminate cards (optional)
- region cards
- electrode cards

The mesh begins as a set of (non-uniformly) spaced x and y lines comprising a simple rectangle. The rectangle can then be distorted to track non-planar geometry or match the doping profile, although strongly non-planar structures are difficult to treat in this way. Mesh lines may be terminated inside the device, and redundant nodes removed from the grid. Material regions and electrodes can then be specified as a union of (distorted) rectangles, completing the mesh specification.

Two cautionary remarks are necessary in relation to rectangular grids. When a rectangular grid is distorted, a large number of obtuse triangles are unavoidably introduced. Secondly, when regridding a rectangular grid, large aspect ratio rectangles ($\geq$4) can also give rise to very obtuse triangles. The issues involved are discussed in Section 6.

4. Irregular grid generation.

To handle general non-planar devices, we have developed an interactive grid generator, IGGI, which is separate from PISCES-II and produces as its final output a text file which PISCES-II can read.

IGGI knows nothing of device simulation and could be used as input to any two-dimensional simulator which supports general triangular grids. Facilities are provided to generate and subdivide arbitrarily shaped regions, to add internal nodes and to triangulate the structure. A grid editor allows the user to access the grid directly, zoom in on particular regions, and add/delete points as desired. In practice, we have found that the limited graphics capabilities of most terminals precludes direct manipulation of medium or large (400 points or more) grids. Consequently IGGI is normally used to generate small, coarse grids

representing the geometry of a device, which are then passed to PISCES-II for refinement.

4.1. IGGI command structure

IGGI is organized as a two-level menu-driven system. The main command level contains global commands, including file input/output, automatic point generation and triangulation, node re-ordering and mesh evaluation. The editor is a subsidiary command level and contains mesh primitives such as adding/removing nodes, region specification, and x and y stretch routines. The various options are described in detail in Appendix B, and the following is a simple overview of the mesh generation process.

The initial mesh is an outline of the structure to be examined. All subsequent regions are subsections of the original outline. The outline can be entered by drawing it on the display device with a graphics cursor, or by using a text editor to generate a file in its input format. The structure can then be divided into different regions corresponding to insulator and semiconductor parts of the device. If the outline is entered as a file, region definitions can also be given in that file, otherwise the divisions are made using the graphics device. Each region has associated with it a "grid spacing" which is a measure of the desired average spacing of points in that region. The spacing is initialized depending on the geometry of the boundary; it can be changed to increase or decrease the number of nodes in that region. Electrodes can also be specified via the grid editor or in the initial file. Once the boundary description is complete internal nodes can be generated, using the spacing parameters defined above. Following this, the grid may be triangulated. The nodes are then optionally re-ordered from left to right and a text file written for PISCES-II.

5. Regrid

The regrid facility allows the user to refine parts of the mesh which satisfy some criterion. Using regrid, fine mesh can be allocated only where necessary, with automatic mesh grading between high and low density regions. The criteria for refinement are based on physical, not geometrical, grounds, so they do not need to be tailored to fit a particular structure.

5.1. Regrid algorithm

The regrid algorithm proceeds by searching the initial grid for triangles satisfying the refinement criterion. Each triangle found is subdivided into four congruent subtriangles, and the various grid quantities (doping, potential, carrier concentrations, mobility,...) are interpolated onto the new nodes using linear or logarithmic interpolation as appropriate for that quantity. The new triangles are on "level 1", while the initial grid is on level 0. After all level-0 triangles have been examined, the same procedure is applied in turn to level-1 triangles, and any subtriangles become level-2 triangles. At each level the grid is checked for consistency and updated to avoid abrupt changes of size from one triangle to the next.

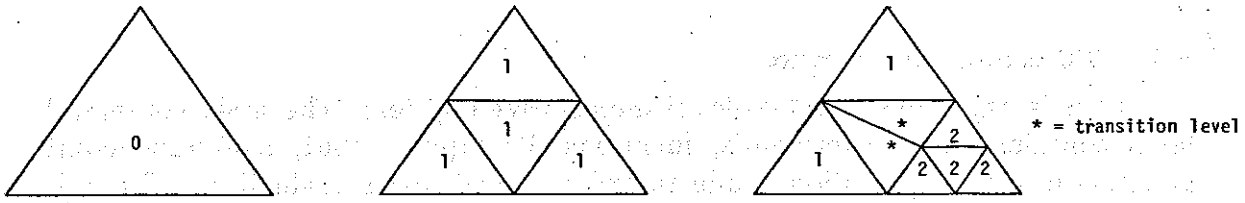


Figure 4.1 - Levels in a grid

5.2. Termination

Refinement is continued until no triangles satisfy the criterion, or until a specified maximum level is reached. Of necessity, grids for semiconductor problems are usually considerably coarser than one would like, so the maximum level is usually the key factor in determining the size of a grid. As discussed below, the final grid is not generated directly from the initial grid, but rather in several steps. At each step the maximum level may be chosen to limit the amount of refinement at that step. The default action is to set the maximum level equal to one more than the highest level in the existing mesh. User specification of the maximum level is sometimes necessary. When a mesh has already been refined several times, and it is desired only to update a coarse region without regridding the finer regions, the maximum level should be set below the level of the finer regions.

5.3. Interpolation

If several levels of regrid are performed in immediate succession, the refinement decisions at the higher levels are made using interpolated data. The nonlinearity of semiconductor problems makes this inadvisable; the data used to refine the grid should be updated as soon as possible. The figure below illustrates this in one dimension. In this hypothetical case, a sharp bend in the potential contour is being refined so that all elements with steps of more than 0.1V across them are being refined. With interpolated data, the whole interval would be refined. If a new solution is performed between levels however, the regrid can detect that the change in potential is localized.

To allow for this phenomenon, it is normal procedure to regrid one level at a time, and to reread the doping cards and perform a new solution between levels.

5.4. Refinement criterion

It is an open question as to what is the correct criterion to refine the grid in semiconductor problems. When solving Poisson's equation by finite-element methods, there is a natural choice, in the error estimate. The grid can be refined so as to equidistribute the Poisson error. For on-state simulation

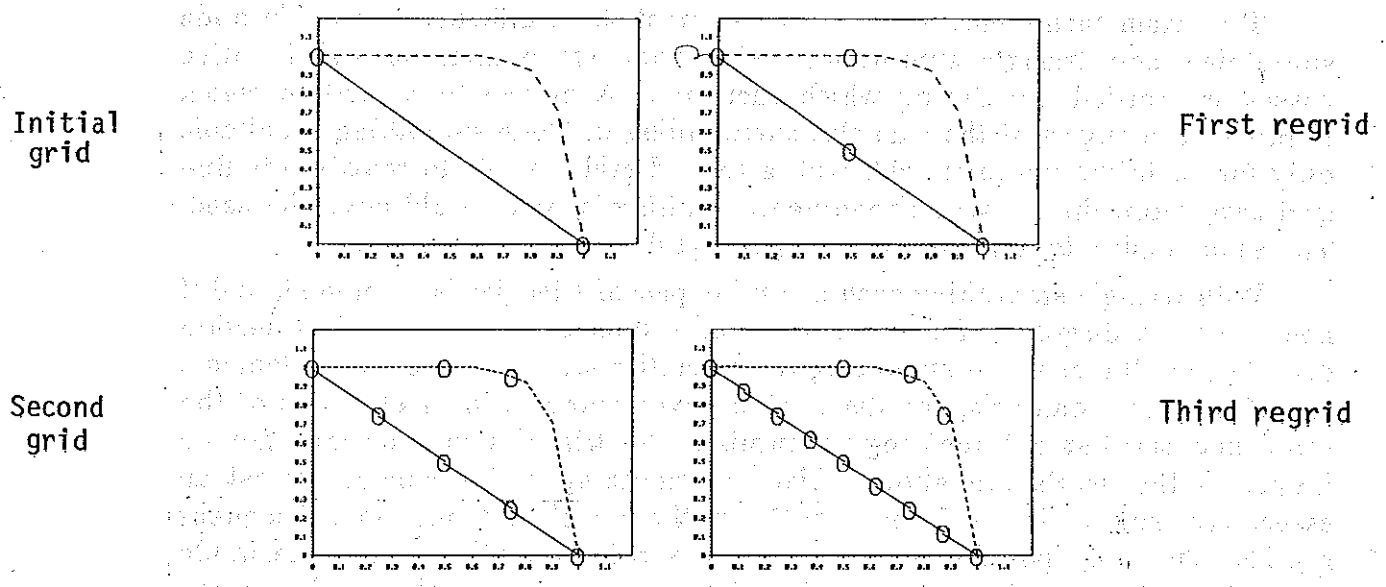


Figure 4.2 - Interpolated vs. Solution Data

however, estimates for the error in the continuity equation are rather unreliable. In addition there is the problem of having three different error estimates, one from each equation.

Rather than deriving mathematically-based error estimates, we have chosen physically plausible heuristics as our refinement criteria. While not optimal, this approach gives considerable control over grid generation. There are two main options, to refine where a particular variable exceeds some value, or to refine where the change in that variable across a triangle exceeds a given value. The variable can be any of the key quantities in the problem $\psi, n, p, \phi_n, \phi_p$, doping, electric field or minority carrier concentration. The value to choose depends on the size of the structure and the accuracy desired. Ideally, no triangle would have a step of more than kT/q in potential or quasi-fermi level across it, but in practice the refinement criterion is 10-20 kT/q. Similarly doping ideally should not change by more than half an order of magnitude across a triangle, but a refinement criterion of two-three orders is more practical. In high level injection, regridding wherever the value of minority carrier exceeds the local doping appears to be useful.

6. Mesh smoothing

Several procedures are available for dealing with poorly shaped elements in the mesh, and in particular with very obtuse triangles. Although every step of grid generation can introduce obtuse triangles, two steps in particular can cause problems.

(1) Distorting a rectangular mesh unavoidably introduces a large number of very obtuse elements.

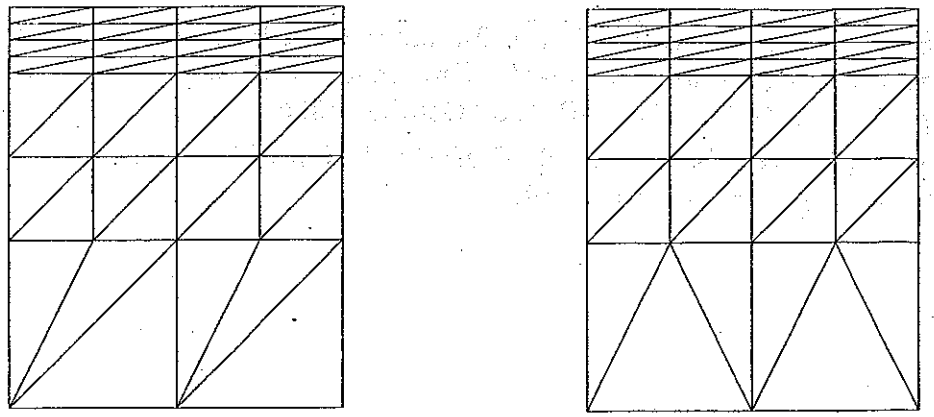
(2) When a grid containing high aspect ratio (≥ 4) triangles is refined, very obtuse elements can be created in the transition region between high and low grid density.

Two main techniques are available to treat these difficulties, namely node smoothing and triangle smoothing. With node smoothing, several iterative passes are carried out during which each node is moved to a position which improves the angles of the triangles surrounding it. Node smoothing is suitable only for an initial irregular grid; with a refined grid it tends to redistribute fine grid away from the physical phenomena requiring it, and should never be used. The same applies to a distorted rectangular grid.

With triangle smoothing each adjoining pair of triangles is examined, and if necessary the diagonal of the quadrilateral is flipped. As described in section 6.1, this has the effect of stabilizing the discretization. When the two elements are of different materials, the diagonal is never redrawn. With elements of the same material but different region number, the triangles may or may not be flipped at the user's discretion. Triangle smoothing is desirable in almost all cases, and should be performed both on the initial grid and on subsequent regrids. The only possible exception to this rule arises from an undesirable interaction of three elements : regrid, high aspect ratio triangles, and smoothing. It occurs frequently in oxide regions, since they often have very long triangles. In the transition region between refined and unrefined regions, nodes are added to the sides of unrefined triangles. If a thin triangle is so modified, a very obtuse angle is created. If the resulting subtriangles are smoothed, strange patterns can result. Usually this is not cause for concern, since no current flows in the oxide, but the triangles may look more pleasing to the eye if smoothing is omitted. There is also a provision to automatically add more points in the oxide, thus avoiding the issue entirely. Figures 4.3a-d illustrate the point just made. Figure 4.3a is a grid which has been refined without smoothing. Figure 4.3b is the same grid, with smoothing. The triangles in the bulk are better shaped. In both these pictures, more points have been added in the oxide , so the mesh maintains good angles throughout. However the extra nodes are redundant, and to avoid this, the extra nodes might not be requested (Fig 4.3c). The smoothing algorithm, in its efforts to stabilize the matrix, creates a mesh hole. If desired, the smoothing can be turned off (Fig 4.3d). It should be borne in mind that the last two grids differ only in how the nodes are connected; the nodes themselves lie in the same positions.

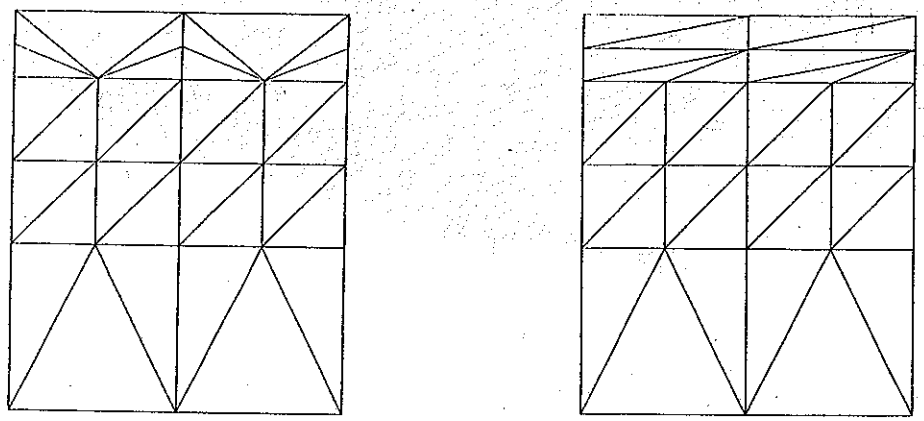
6.1. Obtuse triangles

It is rather difficult to triangulate a general region without obtuse triangles (though an algorithm has been developed[2]), and they have two undesirable side effects on a simulation. The first is that they exacerbate any inherent roughness in the solution, making contour plots more difficult to interpret. More seriously, and less commonly known, they can cause any solution technique to fail. This can occur whenever the sum of opposite angles in a pair of elements exceeds 180 °, causing the matrix coefficient coupling neighboring nodes to change sign. By flipping the diagonal, the sum of opposite angles is made less than 180 °. Figure 4.3a and Figure 4.3b show a grid which is refined, first without flipping and then with flipping. Without flipping, the discretization is potentially unstable, and may lead to unphysical solutions or poor



a b

Figure 4,3



c d

convergence. Figure 4.4a illustrates the *n* quasi-Fermi level obtained after several Newton iterations for an n-channel MOSFET. The grid was an unsmoothed distorted rectangular mesh. A spike of about 50V appeared in the most obtuse part of the mesh, and prevented convergence. Figure 4.4b illustrates the same bias point for the same device, where the triangles have been smoothed. No spike appears. Though this is not a common occurrence, triangle smoothing avoids the problem entirely.

The roughness obtuse triangles bring into the solution is less easy to cure. Generally the best that can be done is to ensure the grid is sufficiently fine where the solution varies rapidly, and that the initial grid has well-shaped elements.

References

[1] R. E. Bank, A. H. Sherman, "A refinement algorithm and dynamic data structure for finite element meshes", University of Texas at Austin Technical Report CS TR-159/CNA TR-166 (October 1980)

[2] B. S. Baker, E. H. Grosse, C. S. Rafferty, "Non-obtuse triangulation of general polygons", to be published.

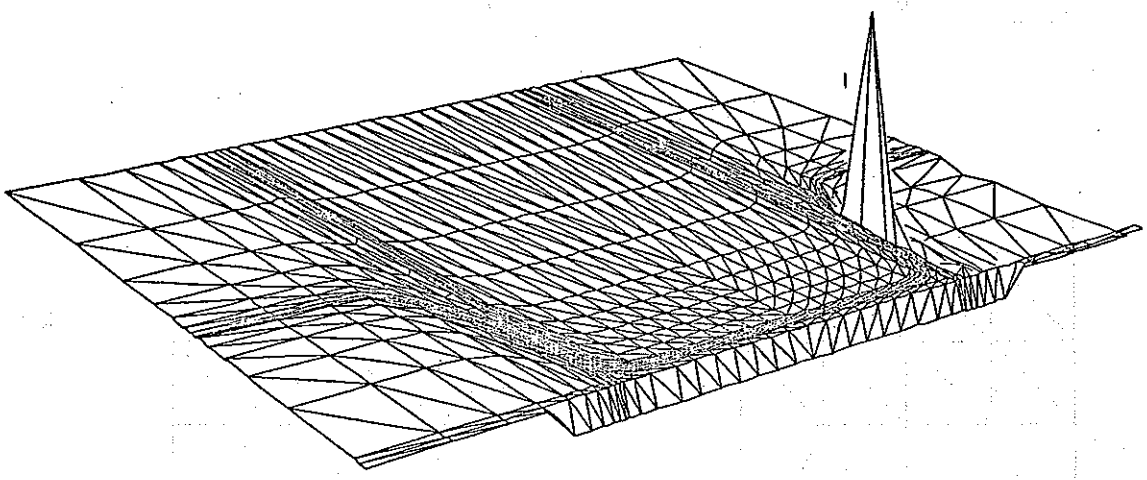
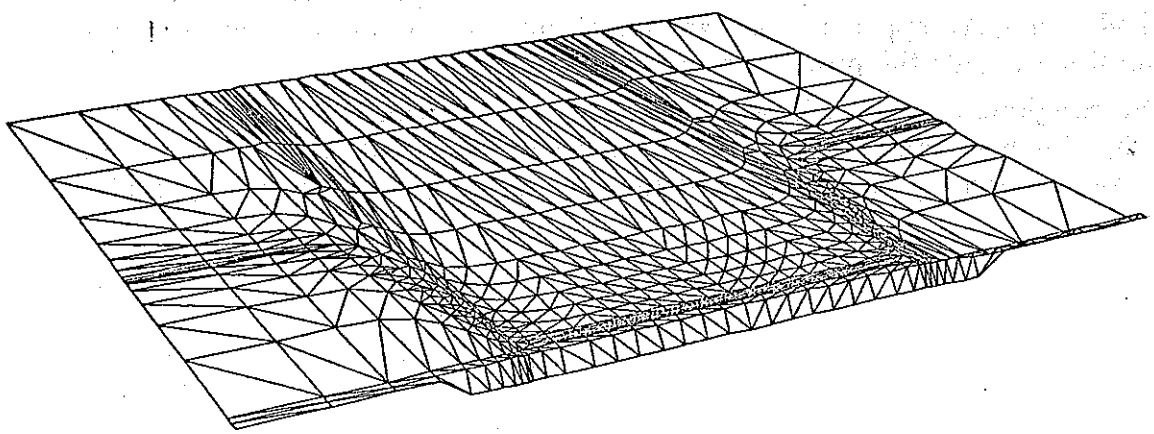


Figure 4.4a

Figure 4.4b



Chapter 5

Examples

1. Outline

In the following chapter, several sample problems are presented to demonstrate some of the features of the PISCES-II program. Sample input and output is shown for a MOS transistor and a CMOS trench isolation cross-section. Except for the initial grid for the trench case, all the input to PISCES-II was through input decks in GENII format. A brief explanation of the function of some of the cards used will be given here as necessary; a complete description of all the PISCES-II input cards is given in appendix A.

In the following examples, a basic simulation procedure is followed in each case. There are usually three separate activities in a PISCES-II simulation. First, a grid is created based on the geometry of the device structure and perhaps the doping and equilibrium potential distributions. Second, solutions are obtained for various bias points using this grid, with optional grid refinement dependent on the solution. Finally, solutions are analyzed and plotted. The three phases are usually (but not necessarily) maintained in separate input decks. The IGGI pre-processor can be involved at the first stage, and the plotting post-processor is of use for solution analysis.

2. MOSFET Example

Two-dimensional simulation has been extensively used in the analysis and design of MOS transistors. Because they are primarily majority carrier devices, MOSFETs are somewhat easier to treat numerically than, for example, bipolar transistors, since under most bias conditions of interest, only a single continuity equation needs to be solved. PISCES-II has the ability to solve only for one carrier by fixing the quasi-fermi level of the second carrier throughout the device (see section 2.5), thereby greatly decreasing the computational expense. Additionally, because PISCES-II can use fully-coupled Newton non-linear iteration schemes, an additional savings in cpu time can be gained over more conventional decoupled methods (see chapter 3).

In this section, a planar MOSFET is presented to illustrate several features of the PISCES-II program. Among the points to be emphasized are (1) the use of non-rectangular grids, (2) the use of SUPREM-III input (with appropriate 2D extensions) for doping profiles, (3) basic grid refinement strategies, (4) algorithmic hints and (5) some of the basic post-processing capability of

PISCES-II. This example is not intended to emphasize fundamental two-dimensional or non-planar effects so that the results can be easily verified. A more challenging example will be presented later in the chapter.

The device under consideration (see figure 5.1) has the following characteristics: a gate oxide thickness of 350 angstroms, a $2\mu m$ gate (mask length), a B threshold-adjust implant and $0.35\mu m$ deep As source/drains, resulting in a metallurgical channel length of approximately $1.5\mu m$. The channel and source/drain profiles were obtained from SUPREM-III and are shown, as plotted by PISCES-II, in figures 5.2a and 5.2b.

Two approaches to grid generation are presented for this device. Both approaches begin with a rectangular grid as a basis. In the first, an initial rectangular grid is distorted to conform to junction edges, while maintaining adequate grid in the channel. The second method starts with a very coarse rectangular grid which is then locally refined as desired. All the simulations presented in this section were done on the distorted rectangular grid.

2.1. Distorted rectangular grid

Figure 5.3 shows the input file for a distorted rectangular grid. There are basically three geometric definition/alteration sections, as noted and described below.

The first section (labeled "A") defines the rectangular basis grid and contains the MESH, X.MESH and Y.MESH cards. Using only such a definition, i.e., ignoring the effects of the remaining sections, one would obtain a 38 by 23 grid as in figure 5.4. Note the non-uniform spacing in the x and y directions as specified by the "ratio" parameter on the X.MESH and Y.MESH cards.

The second section (labeled "B") contains ELIMINATE cards which terminate grid lines within the rectangular grid in order to save points in areas where little solution non-linearity occurs. In this example, 81 points were saved out of 874. Figure 5.5 shows the cumulative effects of sections "A" and "B".

The third section (labeled "C") performs the grid distortion using SPREAD cards. The purpose is to keep the high density of points in the channel region, while following the junction contours out to the left and right edges of the device. For this reason, the SPREAD card is used almost exclusively with MOSFETs.

Following the basic geometric definition in this file are the REGION and ELECTRODE cards, which define material regions and electrode nodes for any rectangular-based grid. For this run, region #1 is oxide and region #2 is silicon while electrode #1 is the gate, electrode #2 is the bulk, electrode #3 is the source and electrode #4 is the drain. Note that the oxide region runs along the entire top of the device, but the gate electrode is only defined from $0.5\mu m \leq x \leq 2.5\mu m$.

Finally, the doping profiles and interface charge density are defined. As previously noted, SUPREM-III input can be used for doping. For this MOSFET, three DOPING cards are required. The first defines the p-type threshold adjust profile which is assumed to cover the entire device region (unmasked);

i.e., the profile is uniform in the x-direction. The second DOPING card defines the As source profile which is uniform in x from the left edge of the device (x=0) to the point x=0.5 μm (the mask/gate edge). From this point on, the profile has a lateral extension defined as 0.75 ("ratio.lat") of that in the vertical direction. The third DOPING card defines the drain in a similar way. The mesh definition concludes by defining a fixed charge density of $10^{10}/cm^2$ along the oxide-semiconductor interface (only under the gate electrode). The mesh is written to a binary file called "hmesh" as designated by the MESH card.

The PLOT/CONTOUR cards at the bottom of the input deck plot the final grid (793 points) and doping contours as shown in figures 5.6 and 5.7.

2.2. Refined rectangular mesh

The second approach presented for setting up a grid for this MOS transistor is accomplished through the input file shown in figure 5.8. The initial rectangular grid, as specified by the MESH and X.MESH/Y.MESH cards, is only 130 points (figure 5.9). After REGION, ELECTRODE, DOPING and QF cards (as above - note that the ELIMINATE card could be applied here as well, but there is no real need to do so), two consecutive REGRID cards are supplied to refine the grid based on doping. Figures 5.10 (176 nodes) and 5.11 (312 nodes) show the intermediate grids after each of these steps. The regrid threshold is specified by "ratio" which for this case is specified as 6 - i.e., if the logarithm of the absolute value of doping changes by 6 or more across a triangle, refine that triangle. Note that although the regrid criterion is only being checked against triangles in the silicon ("region=2"), oxide triangles are being refined as well to maintain reasonably shaped triangles in the thin oxide.

The third and fourth regrids are performed on the basis of potential solutions. To obtain valid potential distributions, PISCES-II must first solve the device equation(s) for at a particular bias point using the grid to be refined (the details of the procedure to simulate particular bias points will be described below). The first regrid on potential uses "initial" conditions, i.e. zero bias, as solved for by simple Poisson analysis ("carriers=0" on the SYMBOLIC card). The second, after necessarily solving for the initial condition again, uses a Poisson solution obtained by applying one volt to the gate electrode to use as a basis for regrid. Figures 5.12 (731 points) and 5.13 (854 points) show the results of each of these steps. Further regrids on potential should probably be done as solutions to other bias points are obtained. A more thorough example of this procedure will be shown in the next section.

Comparing the two alternatives presented here, several statements can be made. First, the distorted mesh must generally suffice for most of the device operating conditions of interest without alteration. Refining distorted meshes, although possible, yields very poor meshes. Second, the refining approach is simpler to perform and requires somewhat less pre-acquired intuition. Third, for the MOS case, because of the thin oxide layer, a large number of points and elements were wasted by the regrid algorithm to keep "good" triangles in the oxide. In fact, if the "ignore=1" option is used to ignore claen-up in region number 1 (the oxide), a much more efficient grid can be generated (see figure

5.14 which has only 808 points). At present, we do not feel that there is any loss of accuracy when these triangles are present in the oxide. A fourth observation regarding the two mesh generation techniques: the distorted grid creates a good number more obtuse triangles (see chapter 4) - approximately 40% - as opposed to only 1.7% with the refined mesh (or about 5% if we prevent oxide clean-up). Fortunately, almost all these triangles lie with an orientation such that very little current flows along the side in violation. Of course in the case of the obtuse triangles in the oxide, no current flow occurs along those sides.

2.3. Simulation

Proceeding with the analysis using the distorted rectangular grid, the gate characteristics are simulated for $V_{DS} = 0.1$ using the input file shown in figure 5.15. Before trying to perform any solutions, PISCES-II requires a mesh and some specifics about the type of analysis and numerical method to be used. In this example, the MESH card specifies that a previously defined mesh in the binary file "hmesh" (the distorted rectangular mesh) be loaded for use in the simulation. Following the MESH card is a SYMBOLIC card, which performs a symbolic factorization in order to save time during the solution phase, when a number of structurely equivalent linear systems are to be solved. In order to set up a symbolic factorization, PISCES-II needs to know which solution method is to be used (Gummel, Newton, Block-Newton or Knot-Newton) and how many carriers are to be solved for (0, 1 or 2). In the initial stage of this simulation, an equilibrium (0 bias) solution will be performed, so the Gummel method will be employed. Further, since linear-saturation bias points and terminal currents are desired, "carriers=1 electrons" is specified. The METHOD card following the SYMBOLIC card includes more detailed (optional) specifics about the numerical algorithms to be used. Any number of METHOD (or SYMBOLIC) cards can be included.

PISCES-II has several cards (MATERIAL, CONTACT, MODELS) to optionally set physical constants, material characteristics and model parameters. The MATERIAL card allows the user to specify material parameters by regions, as defined by the region numbers given during mesh generation. At present, there can be any number of semiconducting regions, but they must all be the same element (Si, GaAs, ...) and must have the same material parameters (with the single exception of permittivity). For example, only one Richardson constant for electrons may be specified for the semiconductor throughout any device. The only parameter set in this run is the surface mobility degradation factor.

The CONTACT card is used to specify the contact work function and surface recombination velocities. Contacts are referred to by number, corresponding to the electrode number specified during mesh generation. Contacts that are not defined by a CONTACT card are assumed to be neutral (no space charge). For this MOS device, the source (#3), drain (#4) and substrate (#2) can be left as neutral contacts; however, the gate contact (#1) is n^+ poly and has a significant work function difference to the p-substrate. The flag "n.poly" is therefore included on a CONTACT card for electrode #1.

The MODEL card specifies global model specifics and parameters; in this simulation, concentration-dependent and field-dependent mobility have been selected, and the temperature is 300K. PISCES-II allows global models (like mobility, recombination, etc.) to be changed through the run by including multiple MODEL cards. However, only one temperature per simulation is allowed. Further, contacts and materials can not be redefined after the first solution is performed in a run. The "print" flag on the MODEL card instructs PISCES-II to print information on the materials, contacts and models defined.

As mentioned above, because of the nature of the initial bias condition, we intend to use the Gummel method. We have also chosen ICCG and damping (both highly recommended for Gummel) on the METHOD card. The first SOLVE card performs a 0 bias solution according to these specifications. The solution is saved in a file called "hout0" for later use.

If the Gummel method was used for the rest of the simulation, computational costs would go up significantly, especially as the device goes from subthreshold directly into saturation (see figure 5.16). It is therefore our recommendation to use the full Newton method for one carrier problems for every bias point with current flow. The Newton method seems to be relatively impervious to bias condition; additionally, a factor of two or more improvement in speed can be obtained by including the "autonr" (automatic Newton-Richardson - see chapter 3) parameter on a subsequent METHOD card.

We are now ready to trace the gate characteristics. Before starting the main body of the simulation, a log file - "IV.sub" - is set up to store the terminal voltages/currents for subsequent plotting (see PLOT.1D cards at the end of the input cards). The following SOLVE card applies 0.1 volts to the drain and saves the solution in "hout.1". Finally, most of the work is done by the next SOLVE card which starts at $V_{GS} = -0.5$ volts and increments the gate voltage by 0.1 volts 25 times (i.e., the final gate voltage will be $-0.5 + 25(0.1) = 2$ volts). The IV characteristics are plotted and displayed in figures 5.17 (log I_D) and 5.18 (linear).

In figure 5.19, an PISCES-II input file is shown which simulates the drain characteristics of the proposed device. After a preliminary definition section as above, the LOAD card is used to load "hout0" (the 0 bias solution) as an initial bias point. The first SOLVE card uses the Gummel method to step the gate voltage to 2 volts, saving $V_{GS} = 1$ volt in "hout1" and $V_{GS} = 2$ volts in "hout2". In the next two SOLVE cards, the drain characteristics are simulated for $V_{GS} = 3$ volts where V_{DS} runs from 0 volts to 3 volts in steps of 0.2 volts. The solution for $V_{DS} = 0.2$ volts is saved in "houtda". As the drain voltage is incremented, the last character of the output file name is also incremented so that the solution for $V_{DS} = 0.4$ volts is saved in "houtdb", $V_{DS} = 0.6$ volts is saved in "houtdc", etc., until V_{DS} reaches 3 volts, and the solution is saved in "houtdo". The IV data is logged into "IV.drain". Note again that as the drain was turned on, the solution method is changed from Gummel to Newton. Figure 5.20 shows the drain characteristics using different mobility models with respect to field (no field dependency and $\beta = 1$ and 2 using eq. 2.34). The curve was obtained by running the program three times with different

parameters on the MODEL card and different solution files and log files.

To demonstrate the final phase of the typical PISCES-II simulation sequence, several examples of post-processing are presented. Figure 5.21 shows an input file to produce several plots of interest. A plot of potential and electron quasi-fermi potential through the middle of the gate electrode for $V_{GS} = 1$ volts from "houtl" (figure 5.22), a plot of channel potential for $V_{GS} = 3$ volts and $V_{DS} = 3$ volts for each of the mobility models (figure 5.23) and a contour plot of potential for the same bias conditions and the $\beta = 2$ (default) mobility case from "houtdo" (figure 5.24) are included. Finally, we show three-dimensional output from the separate post-processor program. Plotted are potential distributions in the semiconductor only for the 0 bias case (figure 5.25) and the $V_{GS} = 3$ volts, $V_{DS} = 3$ volts, $\beta = 2$ case, as above (figure 5.26). These plots were made using the cross-section option on the plotting menu.

3. Trench-isolated CMOS

The following example demonstrates PISCES-II's capability to handle both non-planar and bipolar devices. The structure implemented here has been taken directly from Yamaguchi, et. al. [1] and features a 6 μm deep trench, a $4\mu m$ deep n-well, a p^- epitaxial layer on a p^+ substrate and 0.25 μm source/drain junction depths. The cross-section to be studied includes one source/drain from each of the complementary transistors, along with an n^+ well contact and contacts to the substrate on both the surface and on the back-side (see figure 5.27). The IGGI grid pre-processor was used to input the device geometry.

Figure 5.28 shows the outline of the particular cross-section of interest, as entered into IGGI directly from an SEM photograph. Figure 5.29 shows the structure after generating points and defining regions/electrodes within IGGI (for simplicity, we have made the trench completely oxide-filled, although this is not necessary; we could have added another silicon region in the trench to represent the poly refill). It is most efficient to let PISCES-II determine as much of the grid allocation as possible. Therefore, the densities of points were chosen to be as coarse as possible, making sure, however, that the non-planarities are well represented and that there are no "bad" triangles (i.e., obtuse and/or very thin). Figure 5.30 shows the mesh triangulation while in process. Figure 5.31 is a plot of the finished grid, as output by IGGI (274 points).

Figure 5.32 is a PISCES-II input file which reads the IGGI grid and refines it based on doping. Remember that the regions and electrodes were specified in IGGI, so there are no REGION or ELECTRODE cards in this run file. Doping is supplied in an identical manner to that for the rectangularly based grids. For this particular device, analytical models are used to generate the impurity profiles. Figures 5.33 (376 points) and 5.34 (472 points) show plots of the grid after each respective regrid on doping. Figures 5.35 and 5.36 show the results of the two doping plots requested in the input file; the first is a contour plot of

doping throughout the entire structure, and the second is a vertical slice taken through the p^+ source drain in the n-well.

In a manner similar to that for the refined MOSFET described above, zero bias solutions are used to refine the grid further. A sample input file and output grids are shown in figures 5.37, 5.38 (680 points) and 5.39 (956 points). The structure is then biased up to its normal operating state, with 5 volts on both the p^+ source/drain and the n^+ well contact. Again, the grid is refined on the new potential distributions. Figures 5.40 (1290 points) and 5.41 (1303 points) show the results of these regrid steps. Note that the initial guess for the last two solutions to the 5 volt bias condition were obtained by interpolation from the respective parent grid, rather than re-starting from "initial" conditions. Figure 5.42 shows the final potential contours, obtained from the last grid.

Figure 5.43 is a PISCES-II input file to examine characteristics of the vertical bipolar p-n-p transistor embedded in the CMOS cross-section. In order to prevent the p-n-p-n structure from triggering into a latched state, the n^+ source drain in the substrate is first biased to 5 volts using the Gummel method (with ICCG and damping). From this point, bias on the n^+ well contact (the base contact of the p-n-p) is lowered, forward biasing the base-emitter junction. Because of the aforementioned problems with the Gummel method under such bias conditions, we switch to the block-Newton method, which tends to be just as stable as the full-Newton up to high-level injection but runs quite a bit faster. Before the device starts to enter high-level injection ($V_{BE} = 0.65$ volts), we switch to the full-Newton method. As mentioned in chapter 3, this switch is advisable due to the very strong coupling between Poisson's equation and the continuity equation through the drift terms, resulting in poor inner-loop convergence for the block-Newton. After lowering the well potential to 4.1 volts ($V_{BE} = 0.9$ volts), the collector (substrate) and base (n-well) currents (log) are plotted as a function of the well potential (figure 5.44).

Finally, a post-processing run is made to analyze the solutions; the input file for this run is shown in figure 5.45. Figure 5.46 shows a cross-section through the p^+ "emitter", including doping, electron and hole concentrations, and figure 5.47 is a two-dimensional vector plot of total current for high-level injection conditions ($V_{BE} = 0.9$ volts).

4. References

[1] T. Yamaguchi, S. Morimoto, G. H. Kawamoto, H. K. Park and G. C. Eiden, "High-speed latch-up free 0.5 μm-channel CMOS using self-aligned $TiSi_2$ and deep-trench isolation technologies," IEDM 1983, pp. 522-525.

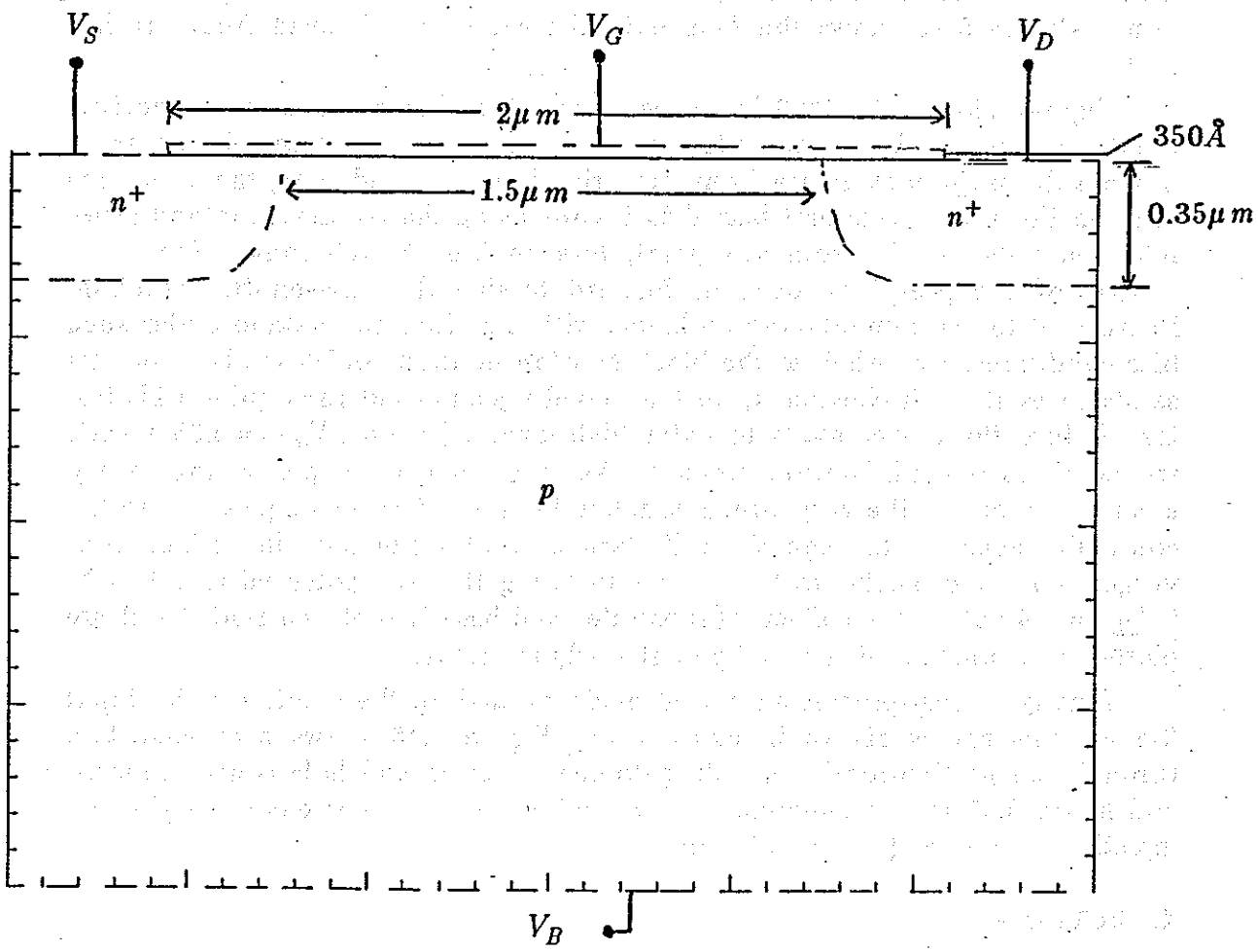


Fig. 5.1. MOSFET to be simulated.

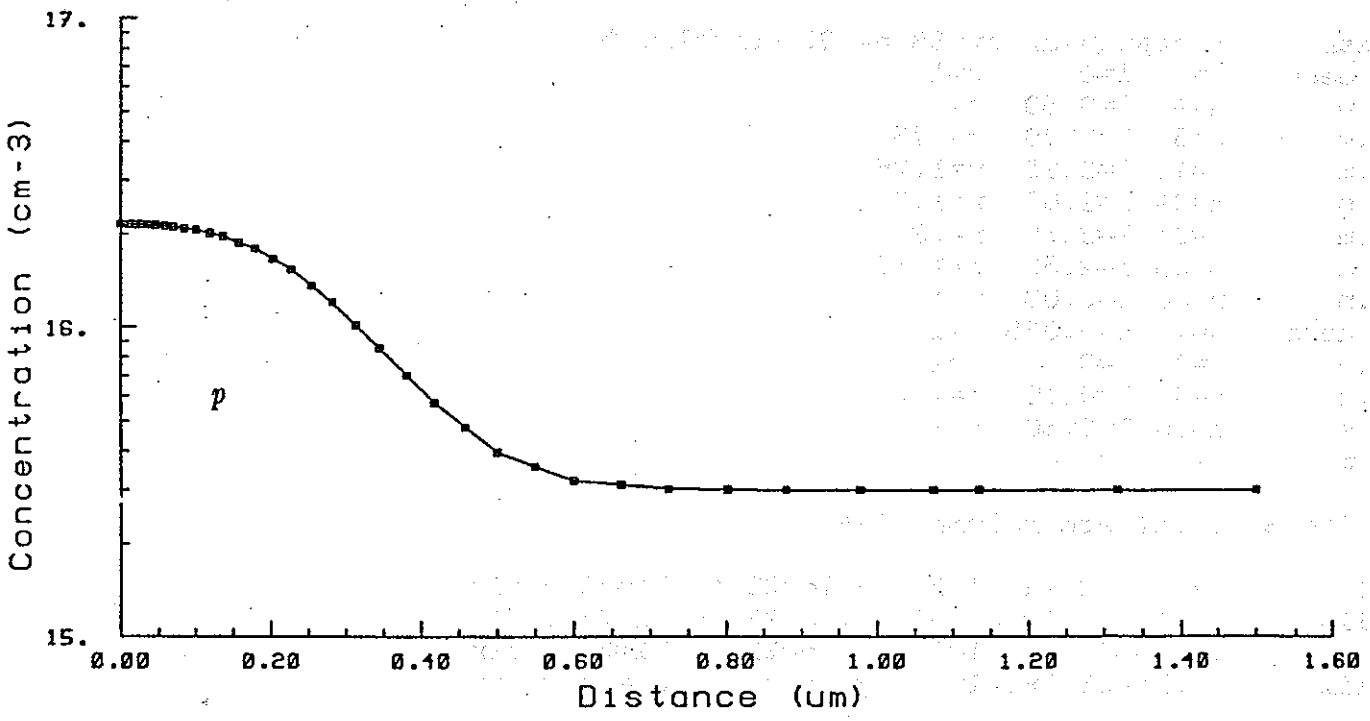


Fig. 5.2a. Channel impurity profile obtained from SUPREM-III .

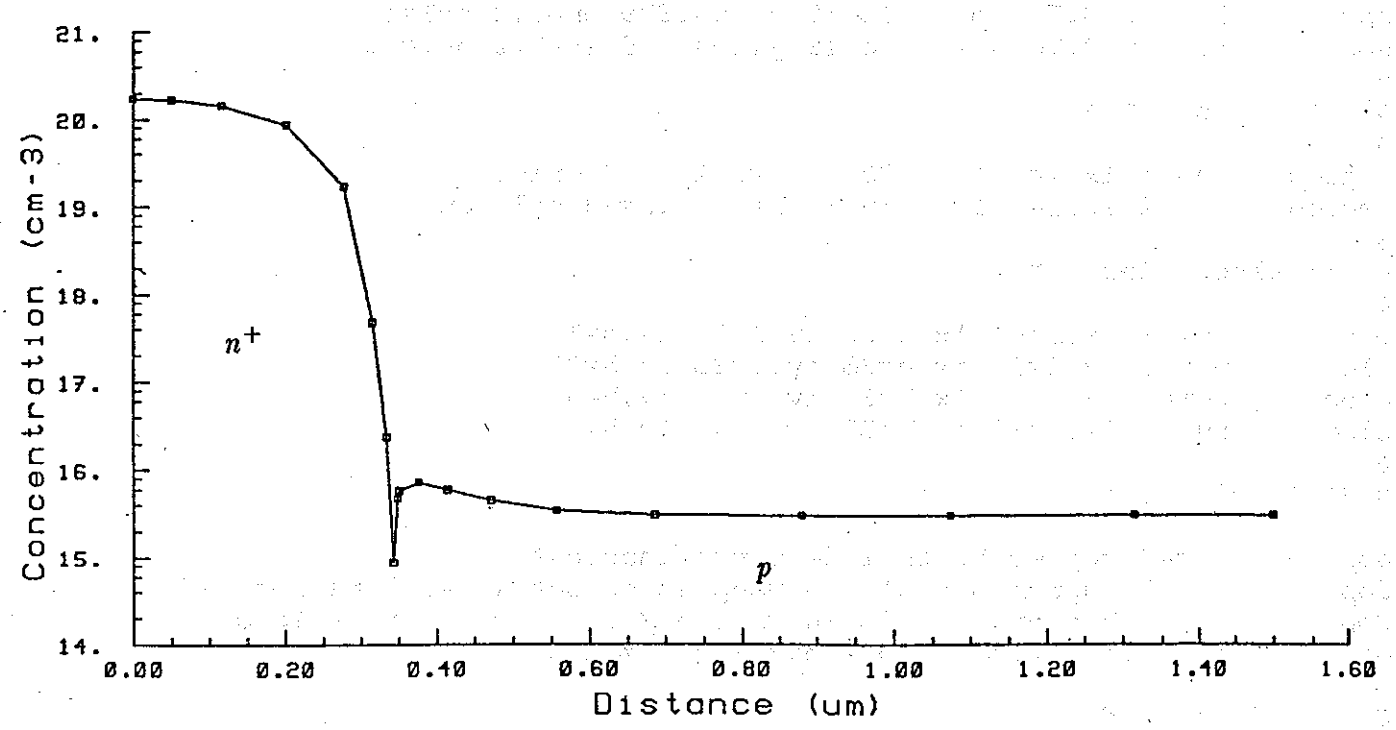


Fig. 5.2b. Source/drain impurity profile obtained from SUPREM-III .

```
Title    HMOS FET (1.5um channel)
$
$ *** A : define rectangular grid ***
$
Mesh     rectangular nx=38 ny=23 outf=hmesh
X.mesh   n=1  l=0     r=1
x.m      n=4  l=0.50  r=.7
x.m      n=8  l=0.70  r=.75
x.m      n=15 l=0.95  r=1.25
x.m      n=24 l=2.05  r=1.0
x.m      n=31 l=2.30  r=.8
x.m      n=35 l=2.50  r=1.33
x.m      n=38 l=3.00  r=1.40
Y.mesh   n=1  l=-.035 r=1
y.m      n=4  l=0     r=1
y.m      n=9  l=0.10  r=1.25
y.m      n=16 l=0.50  r=1.15
y.m      n=23 l=2.0   r=1.25
$
$ *** B: eliminate columns ***
$
elim     ix.lo=13 ix.hi=26 iy.lo=20 iy.hi=23 y.dir
elim     ix.lo=1  ix.hi=38 iy.lo=21 iy.hi=23 y.dir
elim     ix.lo=5  ix.hi=13 iy.lo=21 iy.hi=23 y.dir
elim     ix.lo=25 ix.hi=33 iy.lo=21 iy.hi=23 y.dir
$
$ *** C : distort ***
$
spread   left w=0.70 up=4  lo=13 y.lo=0.35 en=1.2 gr1=1.3
+        gr2=.5 mid=7 y.mid=0.2
spr      righ w=0.70 up=4  lo=13 y.lo=0.35 en=1.2 gr1=1.3
+        gr2=.5 mid=7 y.mid=0.2
spr      left w=0.70 up=13 lo=19 y.lo=.879 en=1.2 gr1=1.5
spr      righ w=0.70 up=13 lo=19 y.lo=.879 en=1.2 gr1=1.5
$
$ *** regions ***
$
region   num=1 ix.l=1 ix.h=38 iy.l=1 iy.h=4 oxide
region   num=2 ix.l=1 ix.h=38 iy.l=4 iy.h=23 silicon
$
$ *** electrodes ***
$
elec     num=1 ix.l=4  ix.h=35 iy.l=1  iy.h=1
elec     num=2 ix.l=1  ix.h=38 iy.l=23 iy.h=23
elec     num=3 ix.l=1  ix.h=3  iy.l=4  iy.h=4
elec     num=4 ix.l=36 ix.h=38 iy.l=4  iy.h=4
$
$ *** doping and fixed charge ***
$
dop      reg=2 suprem boron inf=suprem/chan.out
dop      reg=2 suprem arsenic inf=suprem/sd.out x.r=0.5 ratio=0.75
dop      reg=2 suprem arsenic inf=suprem/sd.out x.l=2.5 ratio=0.75
qf       conc=1e10 x.min=0.5 x.max=2.5 y.min=0 y.max=0

$......Plot grid
plot.2d  grid no.top boundary no.fill pause

$......Plot doping contours and 1d slices
plot.2d  no.top boundary no.fill
contour  doping absolute log min=15 max=20 del=.5 pause
plot.1d  log abs doping x.start=0 y.start=0 x.end=0 y.end=1.5 points pause
plot.1d  log abs doping x.start=1 y.start=0 x.end=1 y.end=1.5 points
```

Fig. 5.3. PISCES-II input file for distorted rectangular mesh.

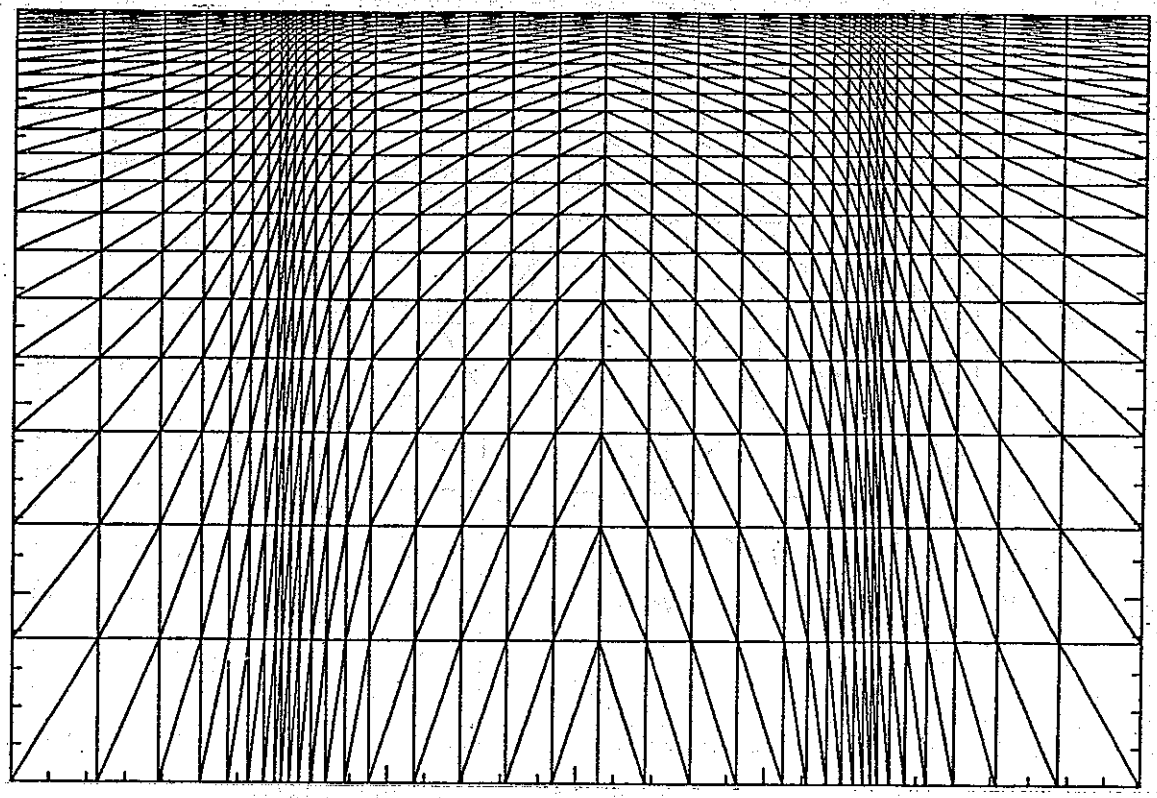


Fig. 5.4. Initial rectangular grid (874 points).

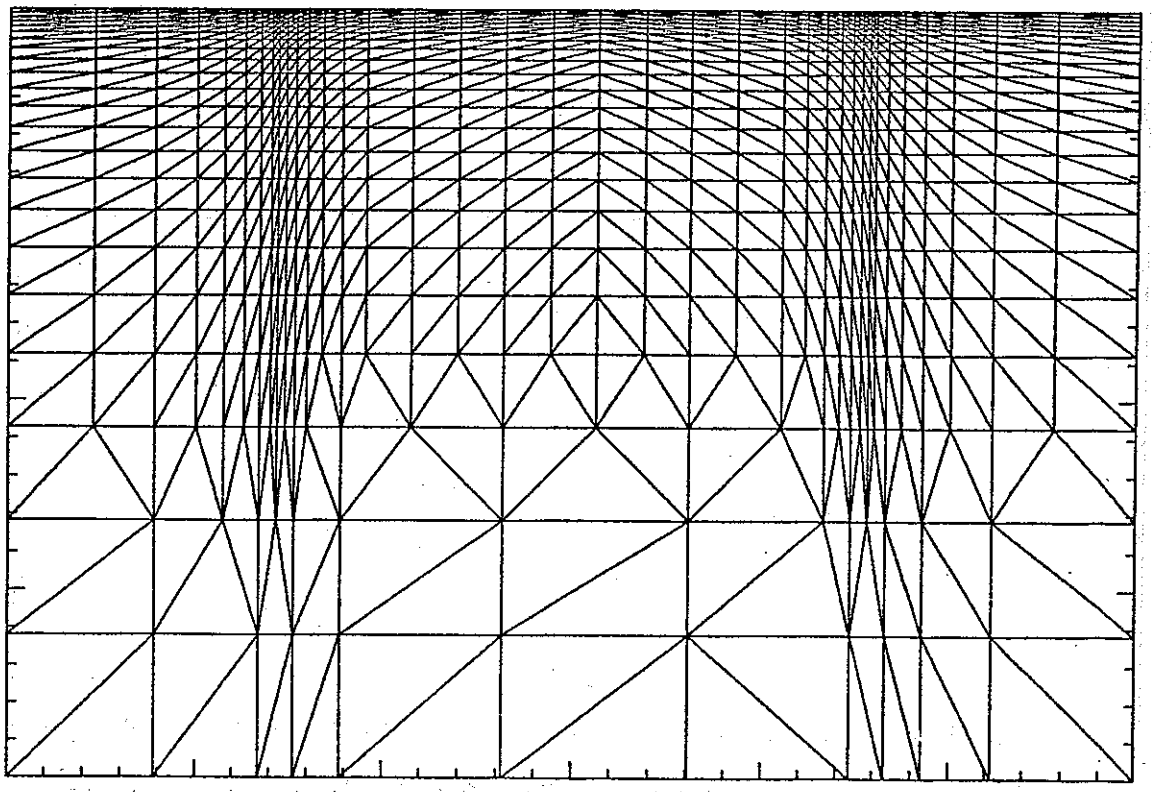


Fig. 5.5. Rectangular grid after eliminate operation (793 points).

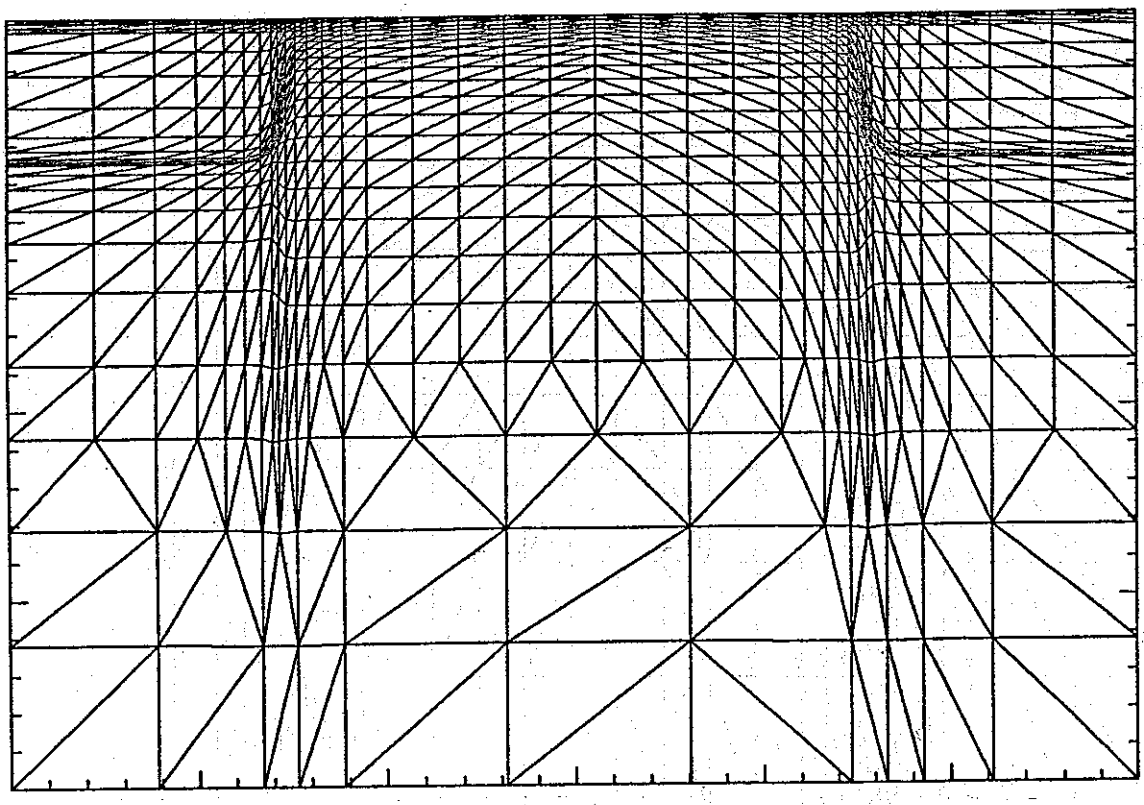


Fig. 5.6. Final grid after distortion (793 points).

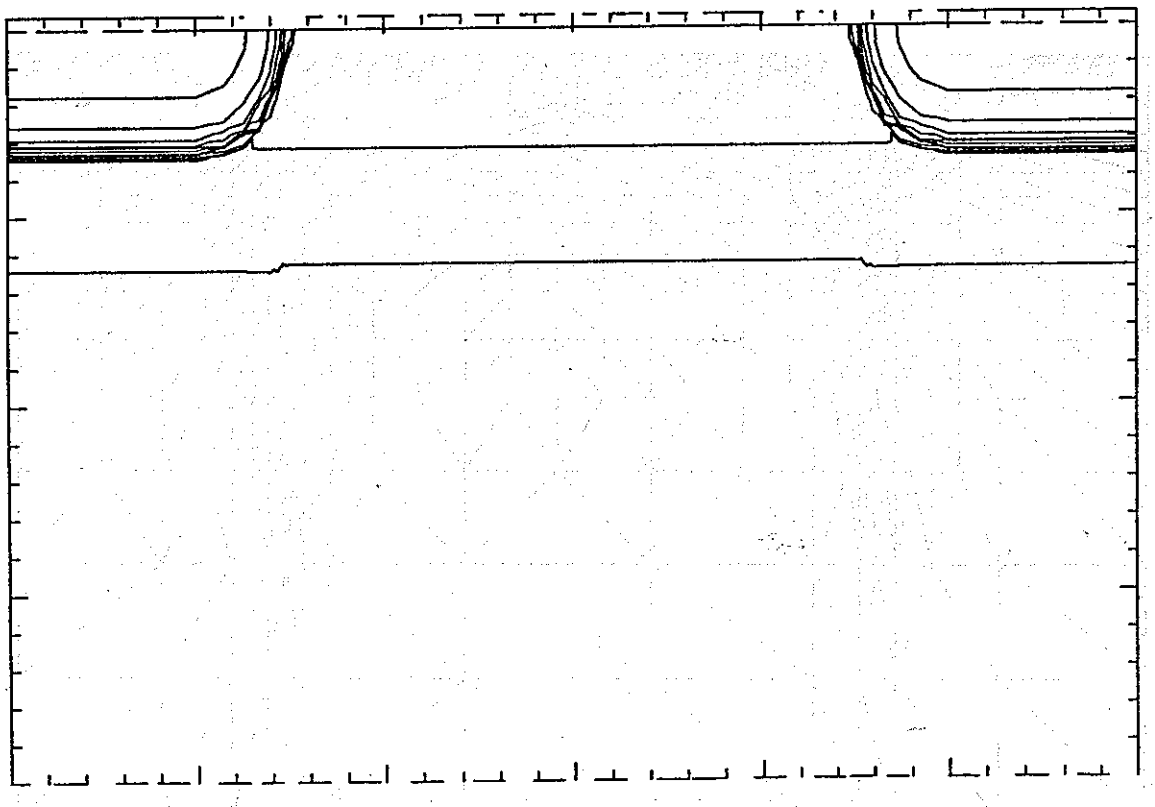


Fig. 5.7. Doping contours for MOSFET.

```
title    HMOS FET (1.5um channel) - Regrid example
$
$ *** define rectangular grid ***
$
mesh     rectangular nx=13 ny=10 outf=hmesh0
x.m      n=1  l=0     r=1
x.m      n=13 l=3.00  r=1
y.m      n=1  l=-.035 r=1
y.m      n=3  l=0     r=1
y.m      n=10 l=2.0   r=1
$
$ *** regions ***
$
region   num=1 ix.l=1 ix.h=13 iy.l=1 iy.h=3 oxide
region   num=2 ix.l=1 ix.h=13 iy.l=3 iy.h=10 silicon
$
$ *** electrodes ***
$
elec     num=1 ix.l=3  ix.h=11 iy.l=1  iy.h=1
elec     num=2 ix.l=1  ix.h=13 iy.l=10 iy.h=10
elec     num=3 ix.l=1  ix.h=2  iy.l=3  iy.h=3
elec     num=4 ix.l=12 ix.h=13 iy.l=3  iy.h=3
$
$ *** doping and fixed charge ***
$
dop      reg=2 suprem boron inf=../suprem/chan.out outf=doping
dop      reg=2 suprem arsenic inf=../suprem/sd.out x.r=0.5 ratio=0.75
dop      reg=2 suprem arsenic inf=../suprem/sd.out x.l=2.5 ratio=0.75
qf       conc=1e10 x.min=0.5 x.max=2.5 y.min=0 y.max=0
$
$ *** regrid on doping ***
$
regrid   doping log reg=2 ratio=6 smooth.k=1 outf=hmesh1 dopf=doping
regrid   doping log reg=2 ratio=6 smooth.k=1 outf=hmesh2 dopf=doping
$
$ *** first regrid on potential ***
$
contac   num=1 n.poly
symb     carriers=0
solve    init
regrid   poten ratio=0.2 reg=2 smooth.k=1 dopf=doping outf=hmesh3
$
$ *** second regrid on potential ***
$
symb     carriers=0
solve    init
regrid   poten ratio=0.2 reg=2 smooth.k=1 dopf=doping outf=hmesh4
```

Fig. 5.8. PISCES-II input file for refined rectangular mesh.

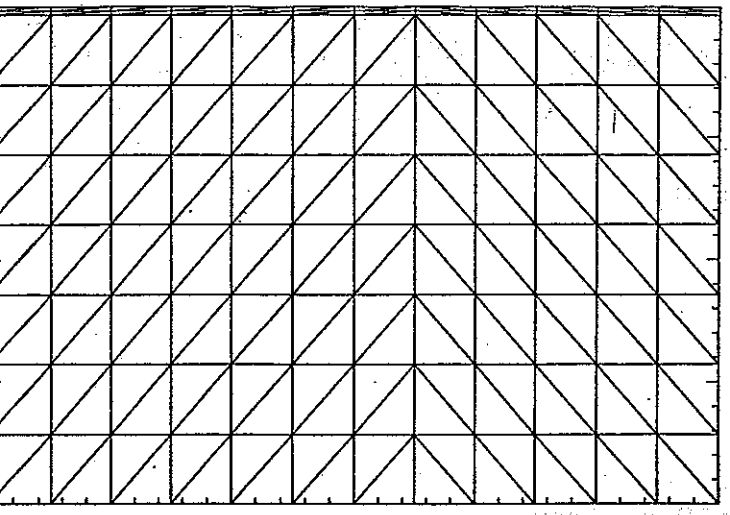


Fig. 5.9. 130 points.

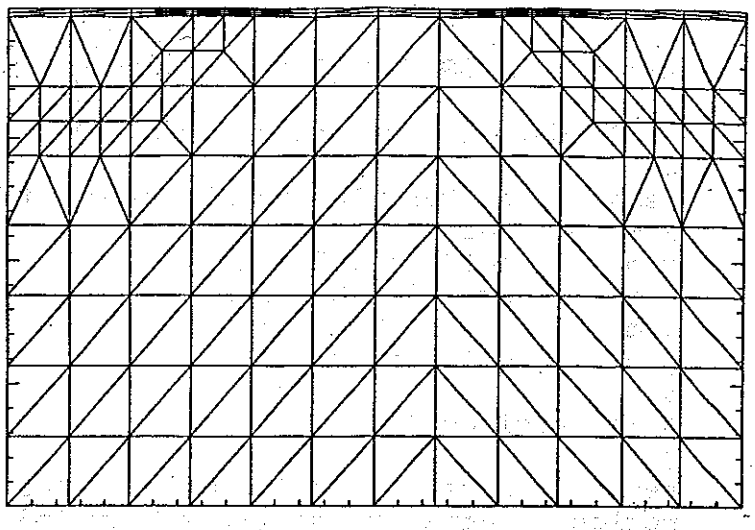


Fig. 5.10. 176 points.

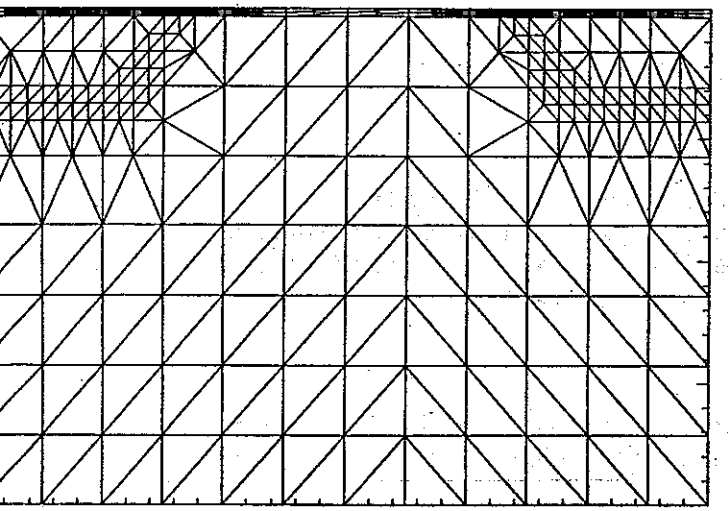


Fig. 5.11. 312 points.

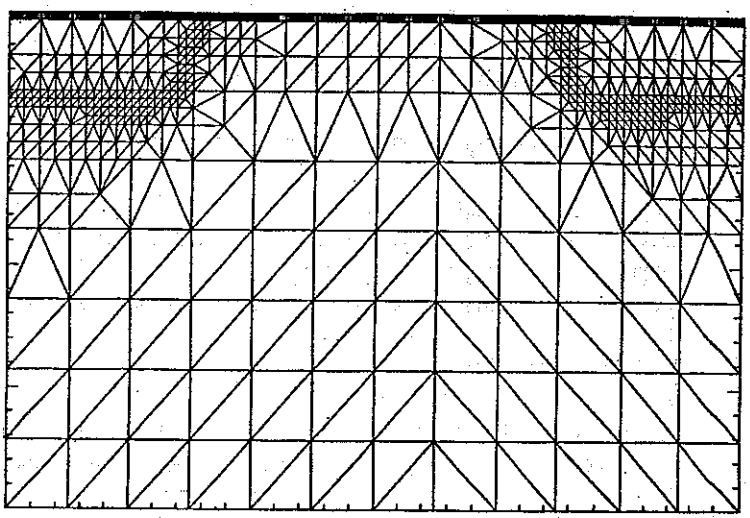


Fig. 5.12. 731 points.

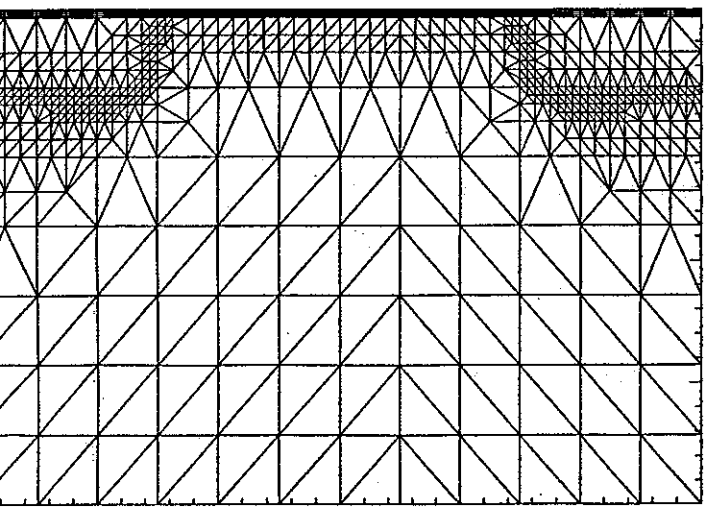


Fig. 5.13. 854 points.

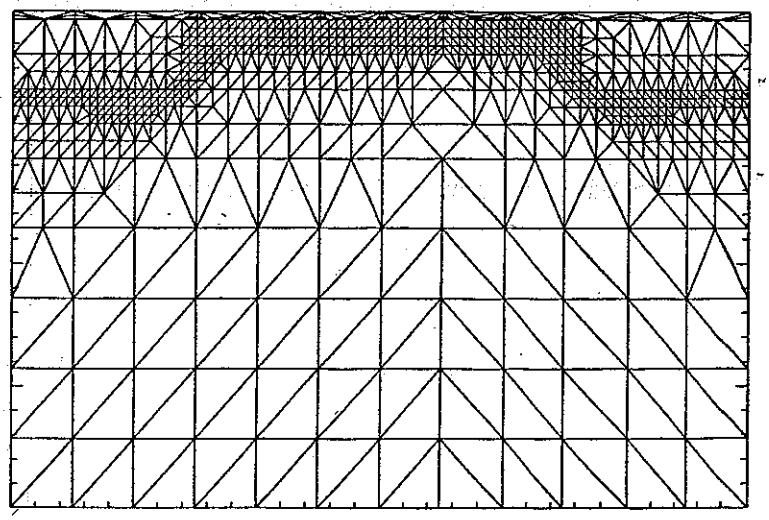


Fig. 5.14. 808 points.

Refinement sequence for MOSFET

```
Title     HMOS FET - subthreshold

$.....Load mesh
mesh      infile=hmesh

$.....Symbolic factorization (Gummel) and parameters
symb      gummel carriers=1 electrons
method    iccg damped

$.....Materials/contacts
mater     num=2 g.surf=0.75
contac    num=1 n.poly

$.....Models
models    conmob temp=300 fldmob print

$*****************************************

$.....Solve initial bias point, save in hout0
solve     initial outfile=hout0

$.....Switch to Newton method
symb      newton carriers=1 electrons
method    autonr

$.....Setup IV log file
log       outfile=IV.sub

$.....Solve for VDS=0.1, save in hout.1
solve     v4=0.1 outfile=hout.1

$.....Step VGS from -0.5 to 2.0 volts (VDS=0.1)
solve     v1=-0.5 vstep=0.1 nsteps=25 electrode=1

$.....Plot ID vs VGS (log and linear scale)
plot.1d   x.axis=v1 y.axis=i4 logarithm
plot.1d   x.axis=v1 y.axis=i4

end
```

Fig. 5.15. PISCES-II input file for simulation of gate characteristics.

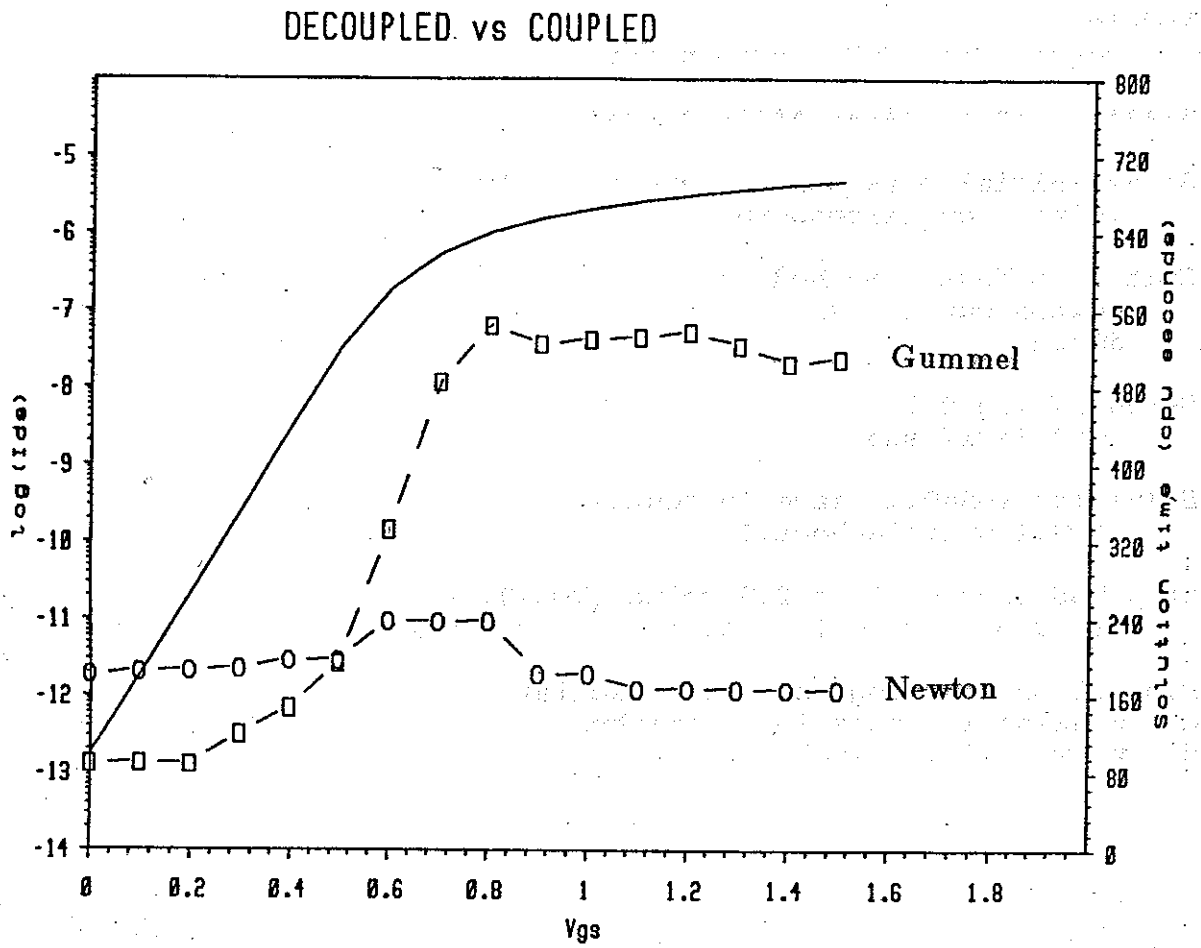


Fig. 5.16. Comparison of Gummel's method with full Newton for a simulation of MOSFET gate characteristics. The computational times cited are for a VAX 11/780 running UNIX.

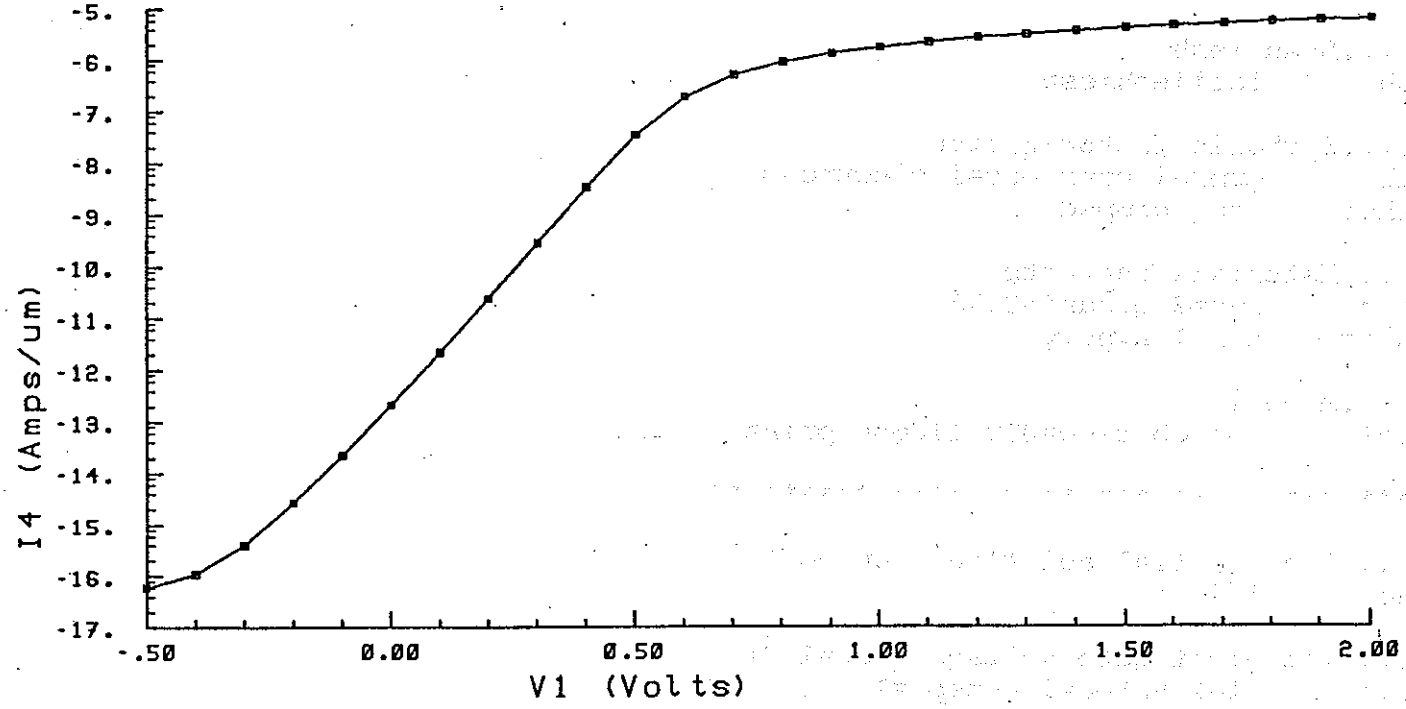


Fig. 5.17. Log I_D vs. V_{GS}.

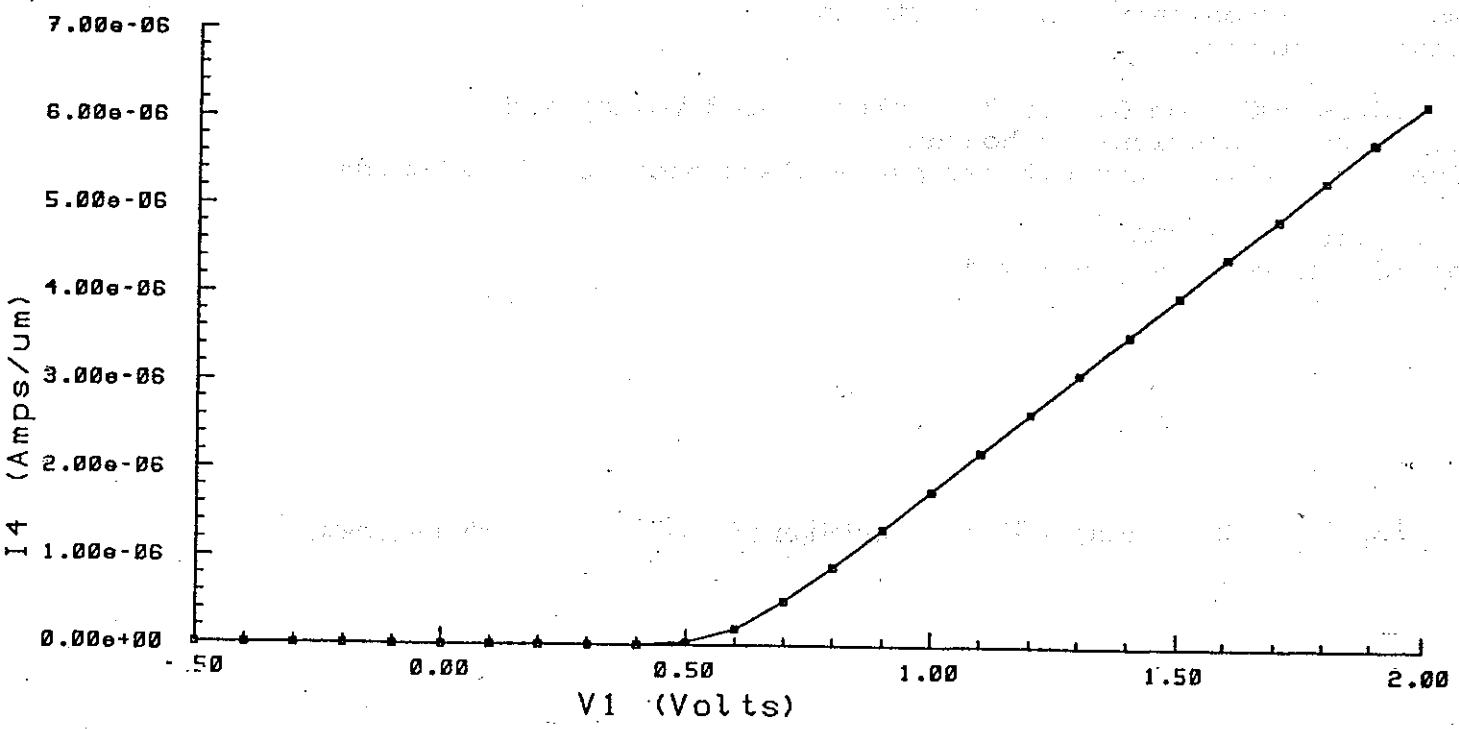


Fig. 5.18. Linear I_D vs. V_{GS}.

```
Title     HMOS FET - drain characteristics

$.....Load mesh
mesh      infile=hmesh

$.....Symbolic factorization
symb      gummel carriers=1 electrons
method    iccg damped

$.....Materials/contacts
mater     num=2 g.surf=0.75
contac    num=1 n.poly

$.....Models
models    conmob temp=300 fldmob print

$*****************************************

$.....Load initial solution from hout0
load      infile=hout0

$.....Increment gate voltage (VGS=1,2)
solve     v1=1 vstep=1 nsteps=1 electrode=1

$.....Setup IV log file
log       outfile=IV.drain

$.....Solve for VGS=3 volts, save in hout3
solve     v1=3 outfile=hout3

$.....Switch methods
symb      newton carriers=1 electrons
method    autonr

$.....Step VDS from 0.2 to 3.0 volts (VGS=3 volts) and
$.....output solutions to houtdx
solve     v4=0.2 vstep=0.2 nsteps=14 electrode=4 outfile=houtda

$.....Plot ID vs VDS
plot.1d   x.axis=v4 y.axis=i4

end
```

Fig. 5.19. PISCES-II input file for simulation of MOSFET drain characteristics.

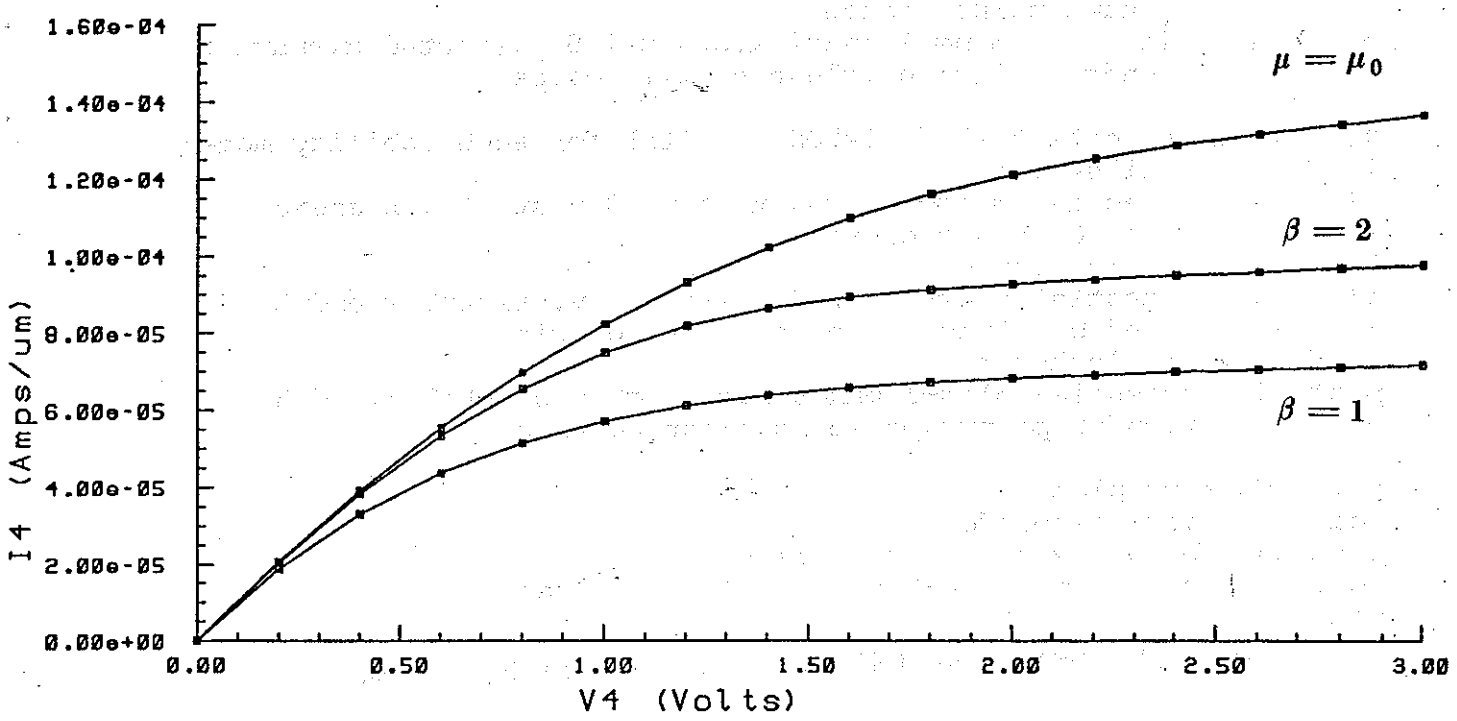


Fig. 5.20. Drain characteristics for $V_{GS} = 3$ volts for different mobility models.

```
title     HMOS FET - plots
$
mesh      in=hmesh
$
mater     num=2 g.surf=0.75
contac    num=1 n.poly
models    conmob temp=300 fldmob
$
$****************************************
$

$.....1D plot of potential through gate (from Si surface)
load      infile=hout1
plot.1d   potential min=-1 max=2 x.start=1.5 y.start=0 x.end=1.5
+         y.end=2 points pause
plot.1d   qfn       min=-1 max=2 x.start=1.5 y.start=0 x.end=1.5
+         y.end=2 points no.clear no.axis pause

$.....Channel potential (VGS=VDS=3 volts) for each mobility model
load      infile=houtdo
plot.1d   potential min=-1 max=4 x.start=0 y.start=0 x.end=3
+         y.end=0 points pause
load      infile=houteo
plot.1d   potential min=-1 max=4 x.start=0 y.start=0 x.end=3
+         y.end=0 points pause no.clear no.axis
load      infile=houtfo
plot.1d   potential min=-1 max=4 x.start=0 y.start=0 x.end=3
+         y.end=0 points pause no.clear no.axis

$.....Contour plot
load      infile=houtdo
plot.2d   boundary no.top no.fill
contour   potential min=-1 max=4 del=0.25 pause
end
```

Fig. 5.21. PISCES-II input file for post-processing solution files.

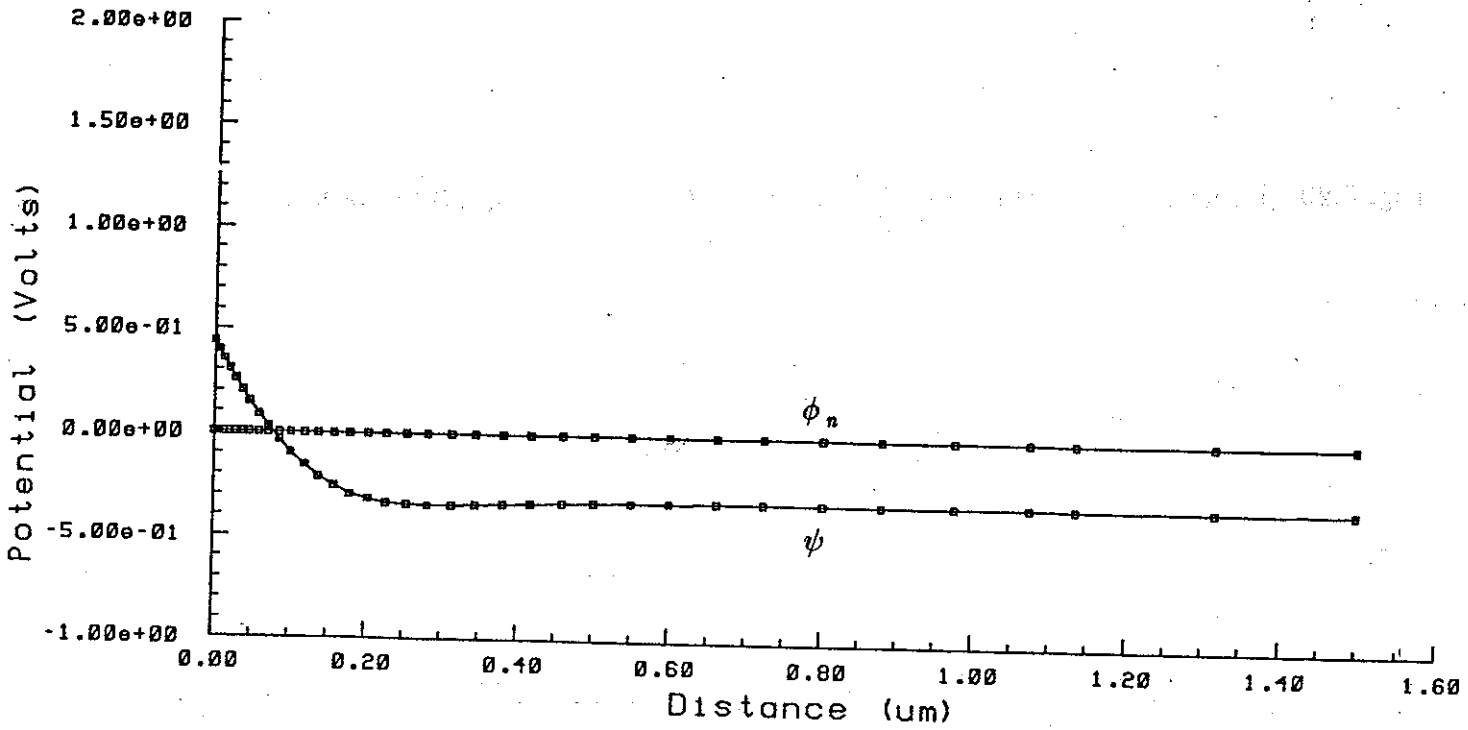


Fig. 5.22. Potential under the gate of the MOSFET from the silicon surface to the bulk for $V_{GS} = 1$ volt (inversion).

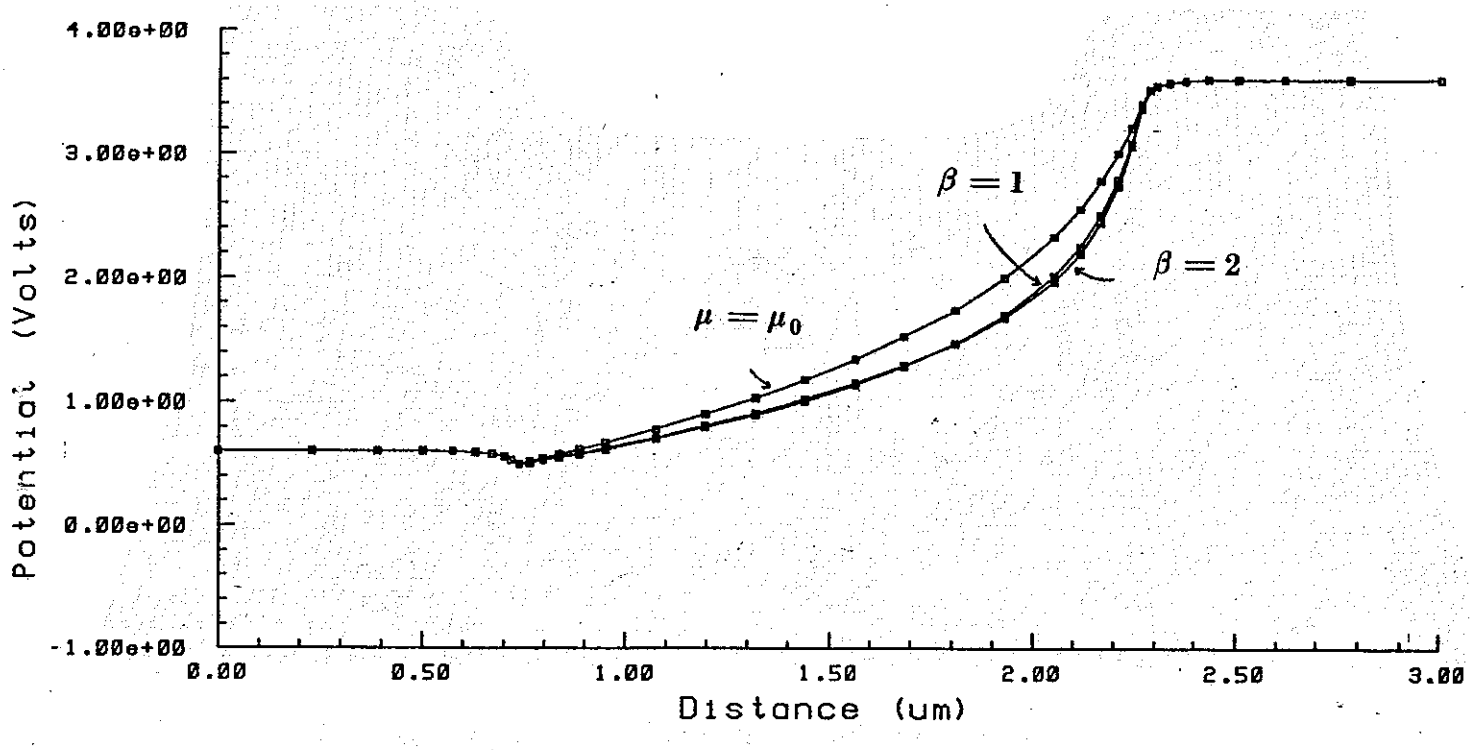


Fig. 5.23. Channel potential for the MOSFET in saturation ($V_{GS} = V_{DS} = 3$ volts) using different mobility models.

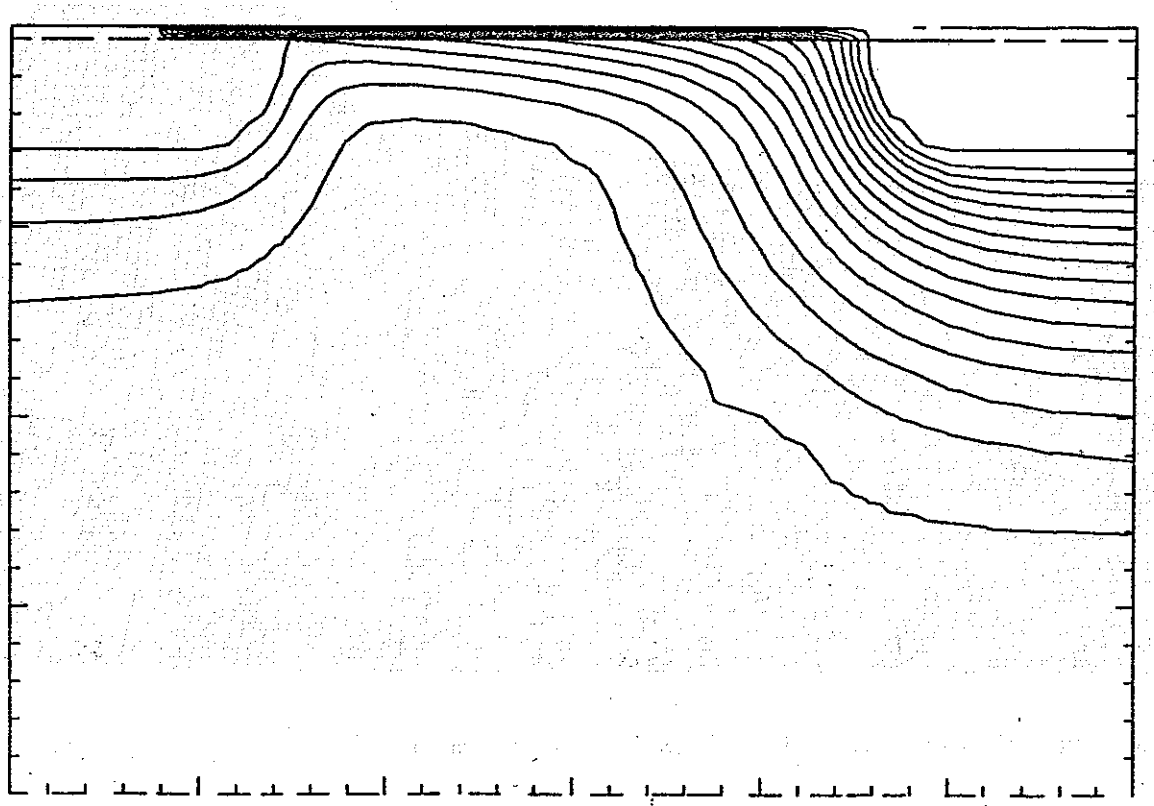


Fig. 5.24. Contour plot of potentials for MOSFET in saturation using field-dependent mobility model with $\beta = 2$.

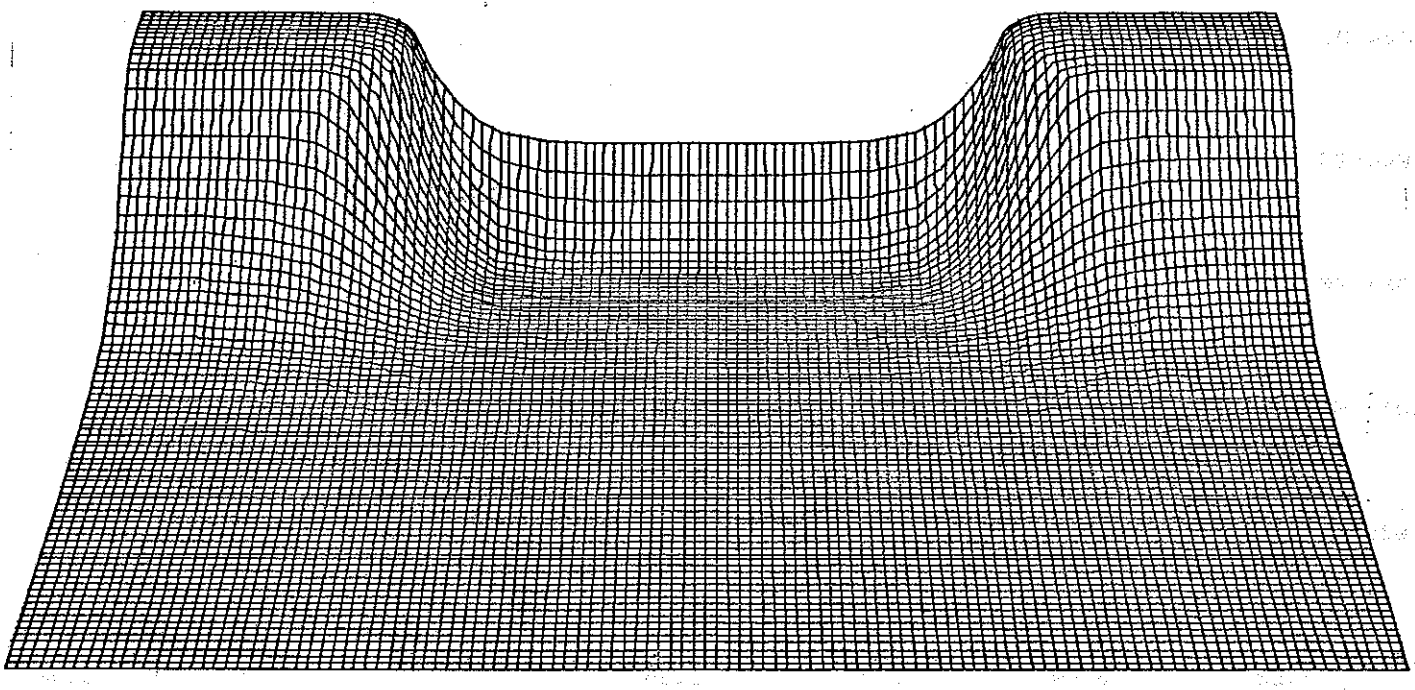


Fig. 5.25. Device potential for 0 bias condition ($V_{DS} = V_{GS} = 0$).

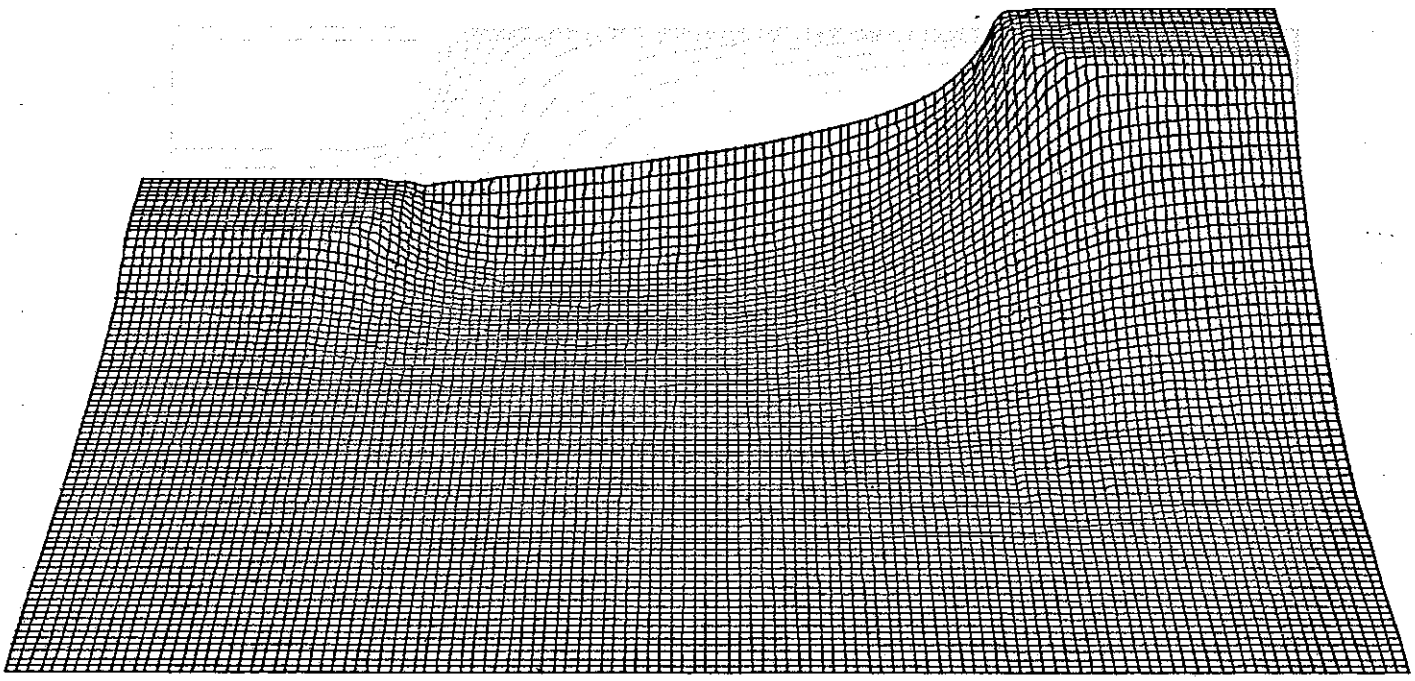


Fig. 5.26. Device potential for MOSFET in saturation ($V_{DS} = V_{GS} = 3$ volts).

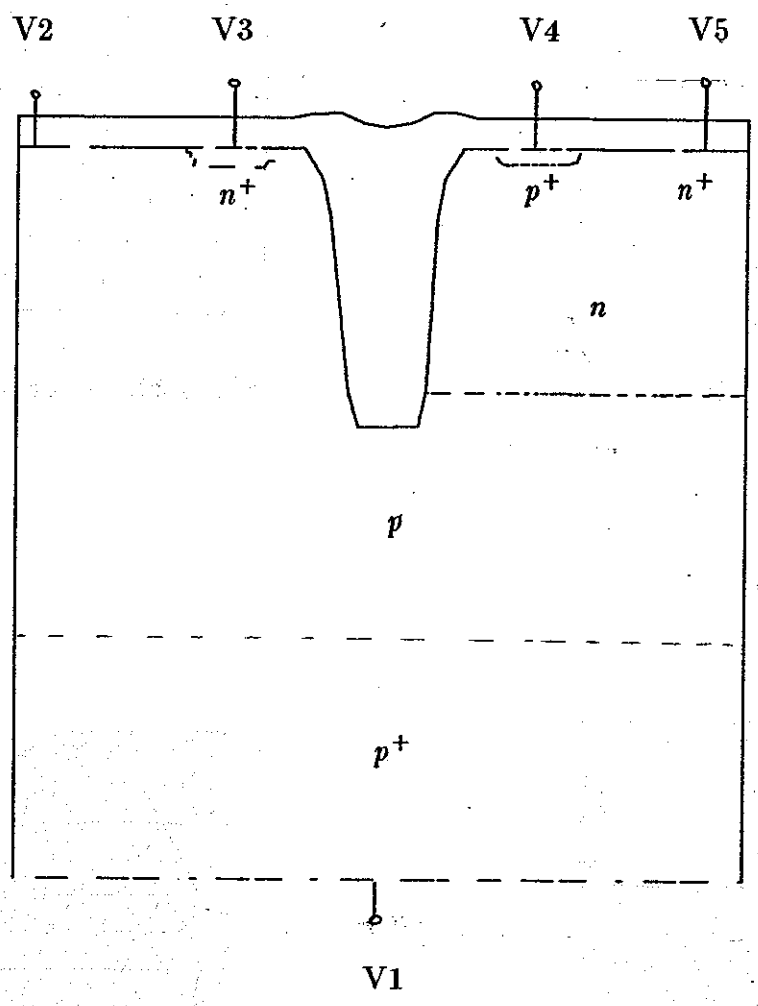


Fig. 5.27. CMOS trench isolation cross-section (after [1]).

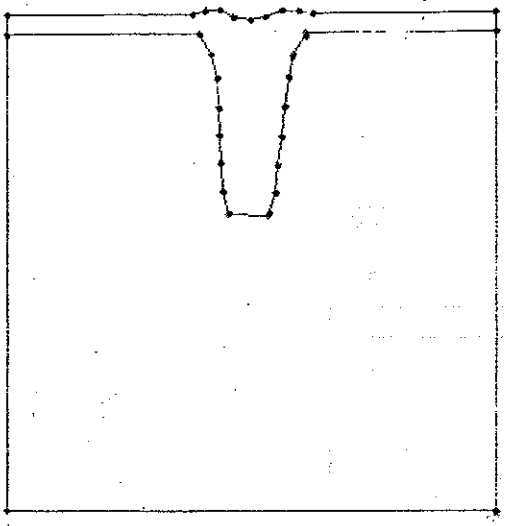


Fig. 5.28. Initial IGGI structure.

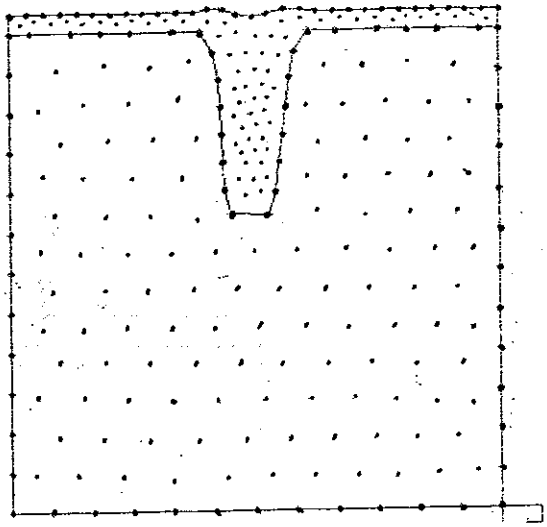


Fig. 5.29. After point generation.

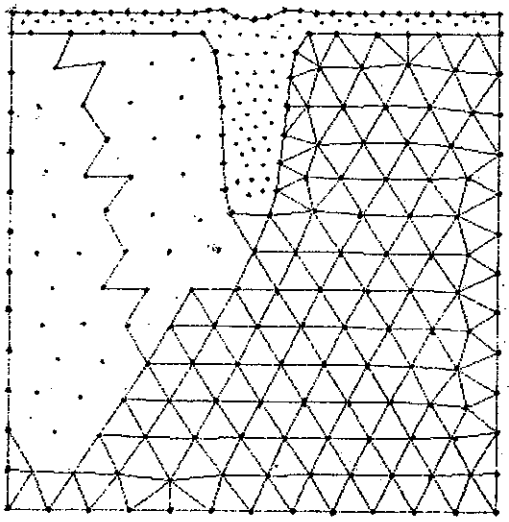


Fig. 5.30. During triangulation.

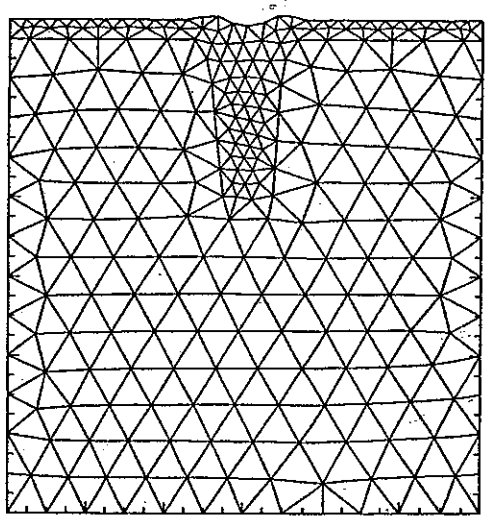


Fig. 5.31. Completed IGGI mesh.

IGGI pre-processor stages.

```
title   TRENCH ISOLATION

$.....Get IGGI mesh
mesh    geom inf=mesh.ig outf=mesh1 smooth.k=13131

$
$ Doping
$
$.....bulk 0.008 ohm-cm
doping  gauss conc=1e19 p.type region=2 outf=doping
+       y.top=8 y.bot=12.1 char=0.5

$.....p-epi 40 ohm-cm
doping  uniform conc=3.5e14 p.type region=2
+       y.top=0 y.bot=8

$.....n-well 2e16, 4um junction
doping  gauss conc=2e16 n.type region=2 junc=4.0
+       y.top=0 y.bot=0    x.left=6 x.right=13 ratio=0.1

$.....Contact implants
doping  gauss conc=2e20 n.type region=2 junc=0.25
+       x.left=3 x.righ=4 ratio=0.8
doping  gauss conc=2e20 p.type region=2 junc=0.25
+       x.left=8 x.righ=9 ratio=0.8
doping  gauss conc=2e20 n.type region=2 junc=0.25
+       x.left=10.5 x.righ=11.5 ratio=0.8

$
$ Regrids on doping
$
regrid   log doping ratio=2 outf=mesh2 smooth.k=1 dopf=doping
regrid   log doping ratio=6 outf=mesh3 smooth.k=1 dopf=doping

$
$ Plots
$
plot.2d  boundary no.fill pause grid no.tic
plot.2d  boundary no.fill pause no.tic
contour  log abs doping min.val=14 max.val=20 del.val=1
plot.1d  log abs doping x.start=8.5 y.start=0 x.end=8.5 y.end=12 points

end
```

Fig. 5.32. Initial PISCES-II input file for reading IGGI grid, doping structure and performing regrid on doping.

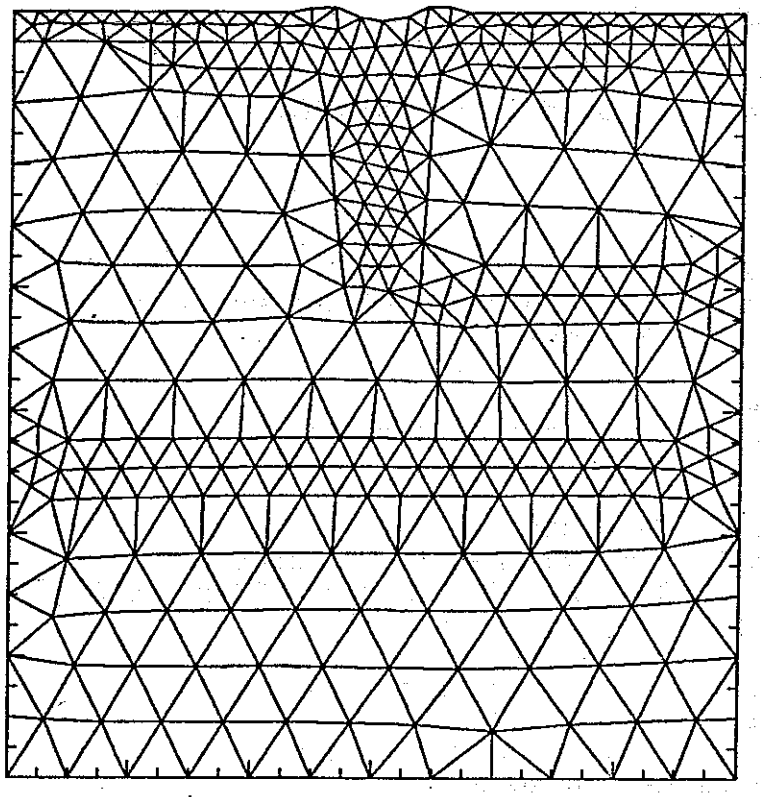


Fig. 5.33. After first doping regrid (376 points).

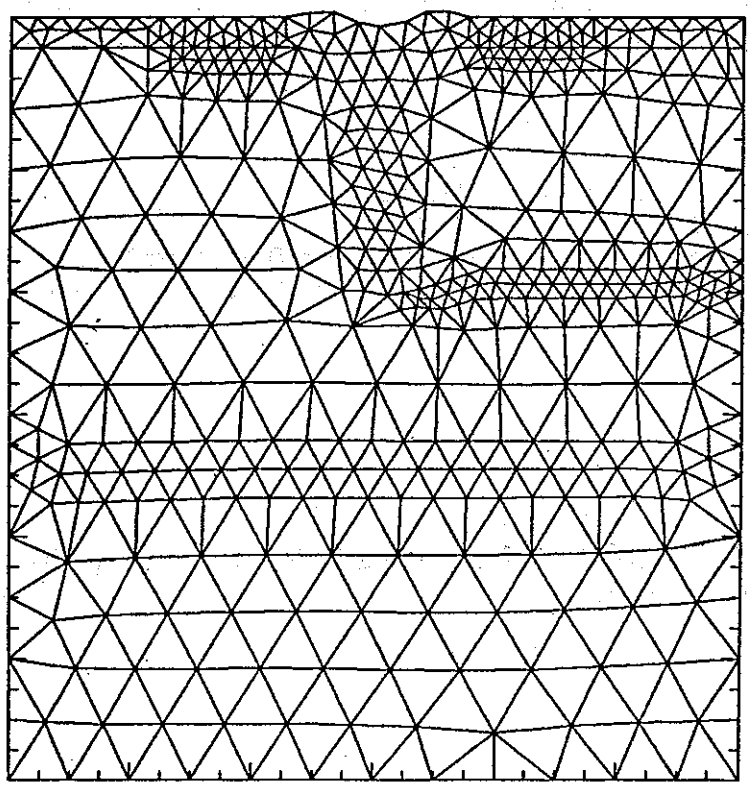


Fig. 5.34. After second doping regrid (472 points).

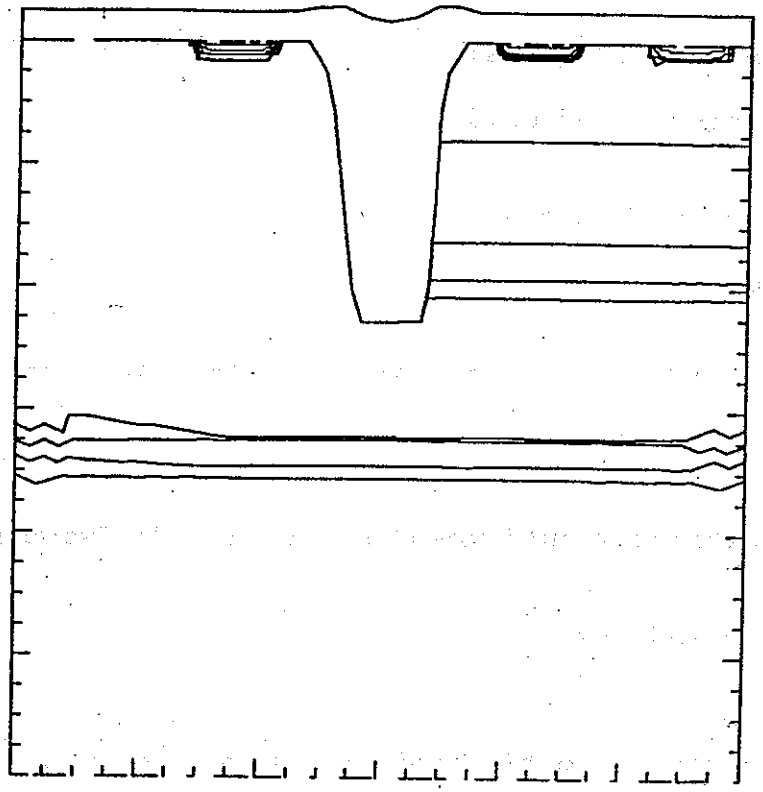


Fig. 5.35. Doping contours for trench device.

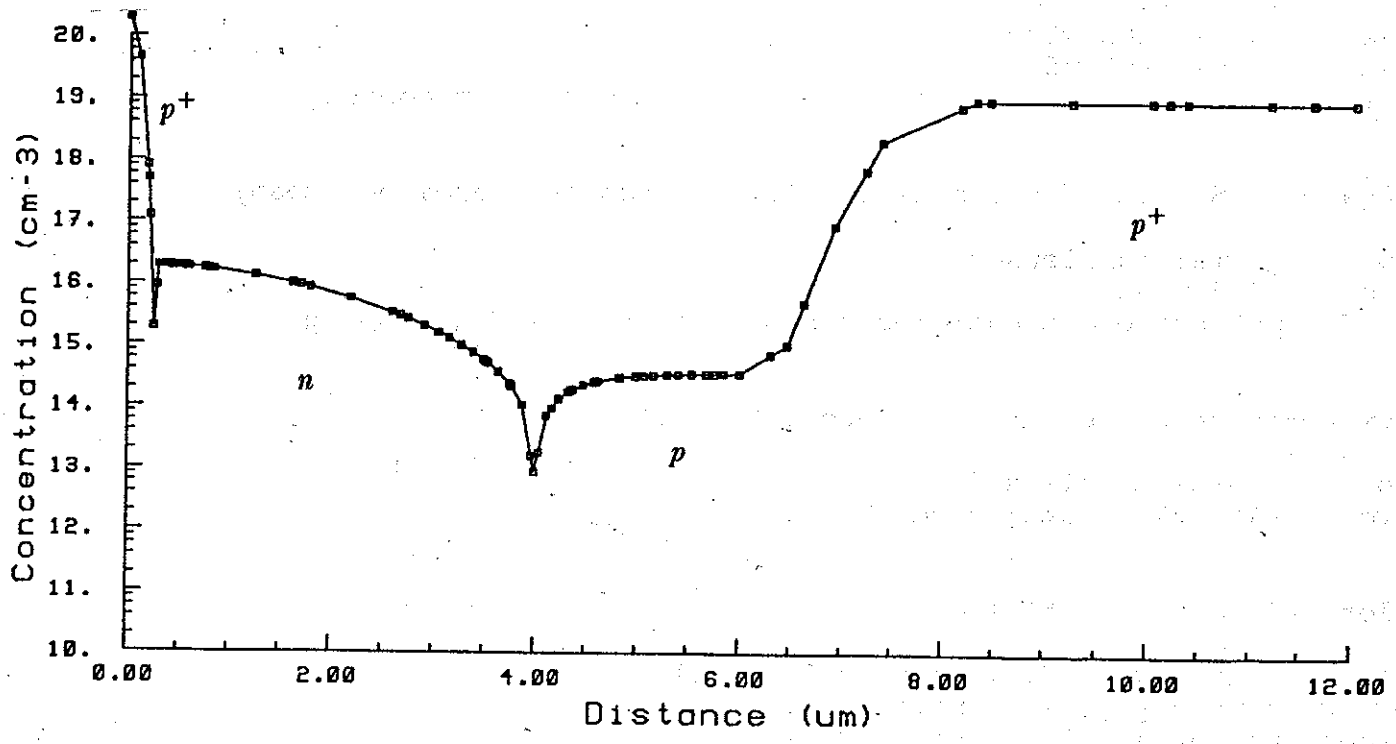


Fig. 5.36. Vertical doping profile through p^+ source/drain in n-well.

```
Title    Trench - potential regrids

$......Get last doping-refined grid
mesh     inf=mesh3

$......Numerical/model parameters
symb     carriers=0
method   iccg damped
model    temp=300

$*****************************************************************
$
$ Initial potential - 1st try
$
solve    initial
regrid   pot reg=2 ratio=0.2 outf=mesh4 smooth.k=1 dopf=doping

$
$ Initial potential - 2nd try
$
symb     carriers=0
solve    initial
regrid   pot reg=2 ratio=0.2 outf=mesh5 smooth.k=1 dopf=doping

$
$ Take to 5 volts and regrid again
$
symb     gummel carriers=2
meth     iccg damped
solve    initial
solve    local v4=1 v5=1
solve    local v4=2 v5=2
solve    local v4=5 v5=5
regrid   pot ratio=0.5 outf=mesh6 reg=2 smooth.k=1 dopf=doping

$
$  Again at 5 volts (interpolate initial guess from previous mesh)
$
symb     gummel carriers=2
solve    v4=5 v5=5
regrid   pot ratio=0.5 outf=mesh7 reg=2 smooth.k=1 dopf=doping

$
$ Now re-solve on new grid (interpolate again)
$
symb     gummel carriers=2
solve    v4=5 v5=5 outfile=out7.5

$
$ Plot grid and potential contours
$
plot.2d  boundary no.fill pause grid no.tic
plot.2d  boundary no.fill no.tic
contour  potential min=-1 max=6 del=0.25

end
```

Fig. 5.37. PISCES-II input file for regrid of CMOS trench structure on potential.

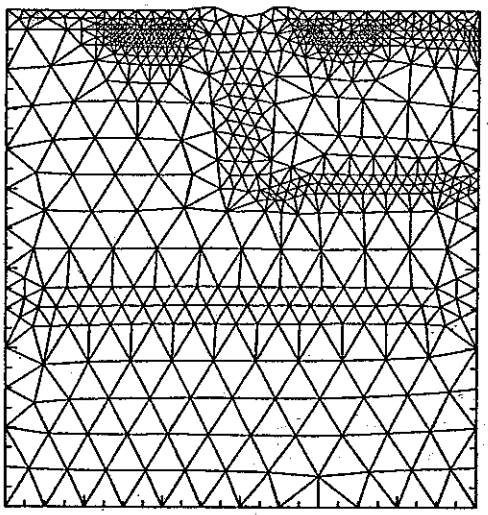


Fig. 5.38. 680 points.

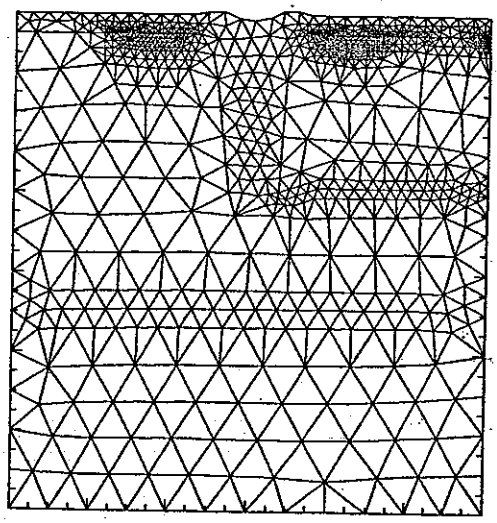


Fig. 5.39. 956 points.

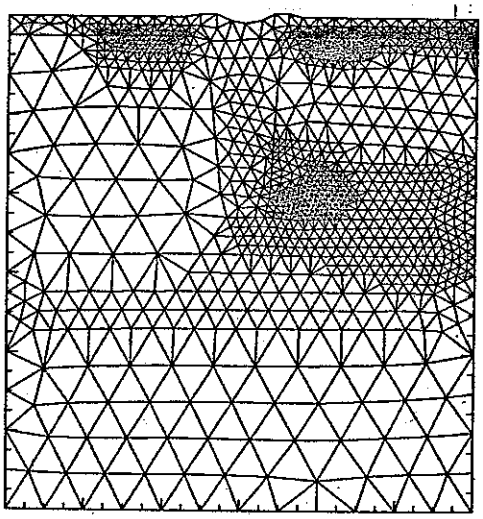


Fig. 5.40. 1290 points.

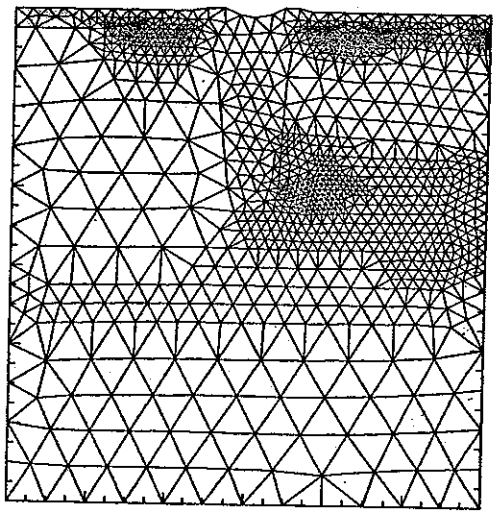


Fig. 5.41. 1303 points.

Refining trench CMOS structure on potential solutions.

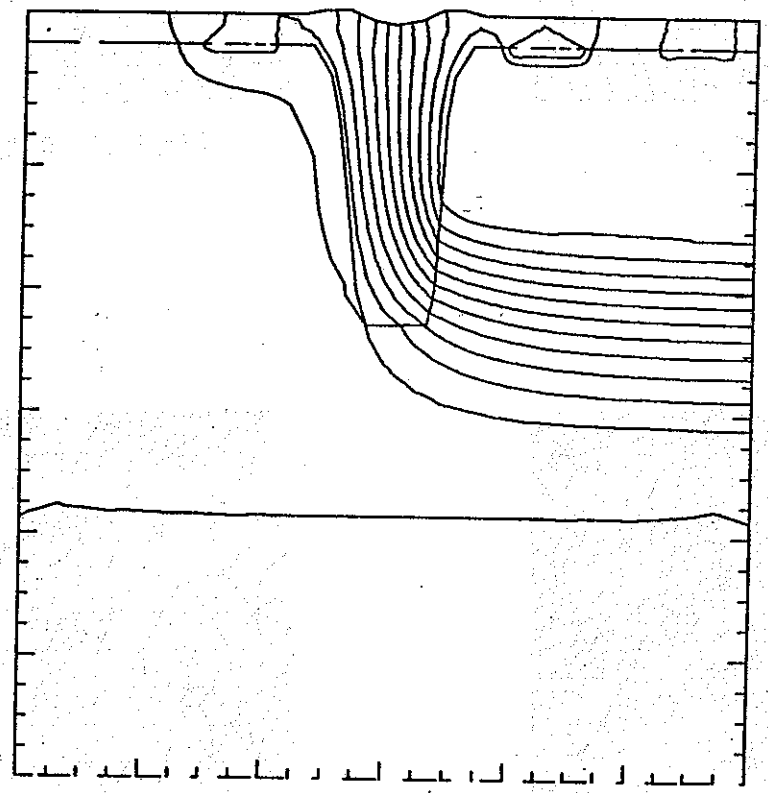


Fig. 5.42. Potential contours for standard MOS bias configuration ($V1 = V2 = V3 = 0,\ V4 = V5 = 5$ volts).

```
title     Forward bias using block CR and full Newton
mesh      infile=mesh7
model     temp=300 conmob srh auger
$
symb      gummel carrers=2
method    iccg damped
$
$*************************************************************
$

$.....Load initial 5v solution and start to take n+ s/d up to
$.....5v to prevent latch-up
load      inf=out7.5
solve     local v3=1
solve     v3=2

$
$.....Log file
log       outf=IV.vertical

$
$.....Now take n+ to 5v
solve     v3=5 outf=out7.v0

$
$.....Switch to block-Newton
symb      newton carrers=2 block
method    slotboom scaled maxinn=30

$
$.....Solve up through medium-level injection
solve     v5=4.9 vstep=-0.1 nsteps=5 electrode=5 outfile=out7.va

$
$.....Change methods for high-level injection
symb      newton carrers=2
method    autonr

$.....High-level injection simulation
solve     v5=4.35 vstep=-.025 nsteps=7 electrode=5 outfile=out.vg

$.....Plot IV curves (collector and base current vs. base voltage)
plot.1d   x.axis=v5 y.axis=i1 log absolute min=-13 max=-2 pause points
plot.1d   x.axis=v5 y.axis=i5 log absolute min=-13 max=-2 pause points
+         no.axis no.clear

end
```

Fig. 5.43. PISCES-II input file to simulate vertical p-n-p transistor.

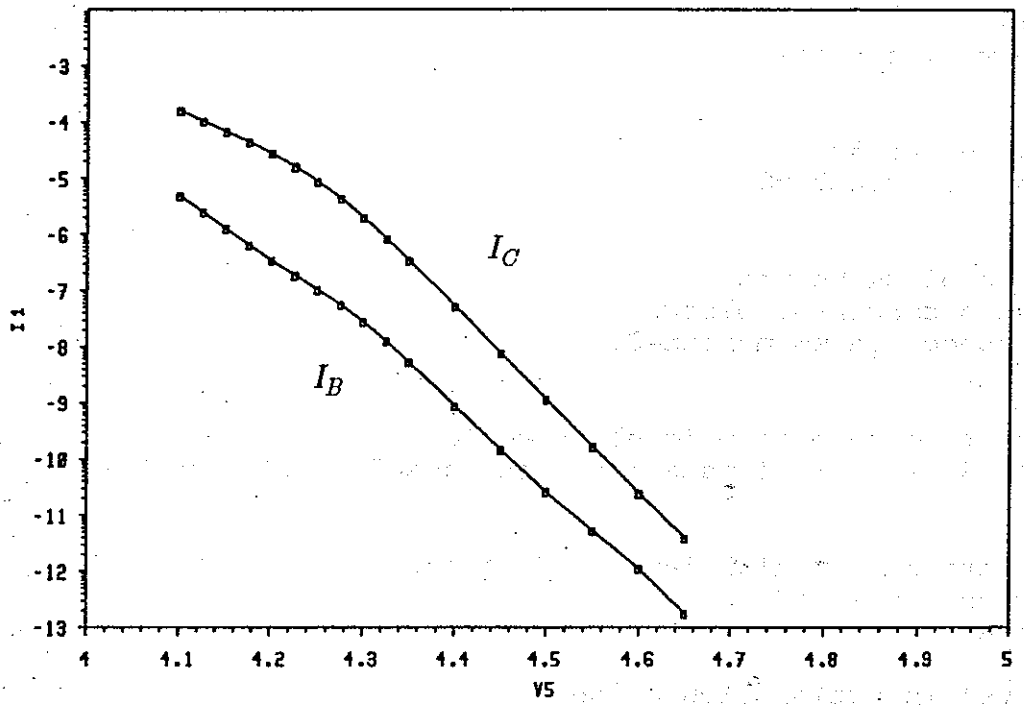


Fig. 5.44. Log I_B, I_C vs. V_{n-well} for the trench CMOS p-n-p transistor.

```
title     Plot high-level injection point
mesh      infile=mesh7
model     temp=300 conmob srh auger
$
$*************************************************************
$

$.....Load solution for VBE=0.9v
load      infile=out.vq

$.....Plot doping/n/p
plot.1d   doping    abs log x.start=8.5 x.end=8.5 y.start=0 y.end=12
+         min=10 max=20 points pause
plot.1d   electrons abs log x.start=8.5 x.end=8.5 y.start=0 y.end=12
+         min=10 max=20 points pause no.clear no.axis
plot.1d   holes     abs log x.start=8.5 x.end=8.5 y.start=0 y.end=12
+         min=10 max=20 points pause no.clear no.axis

$.....Vector plot of current
plot.2d   boundary junction no.tic no.fill pause
vector    j.total scale=50 linear

end
```

Fig. 5.45. Post-processing run on trench CMOS structure.

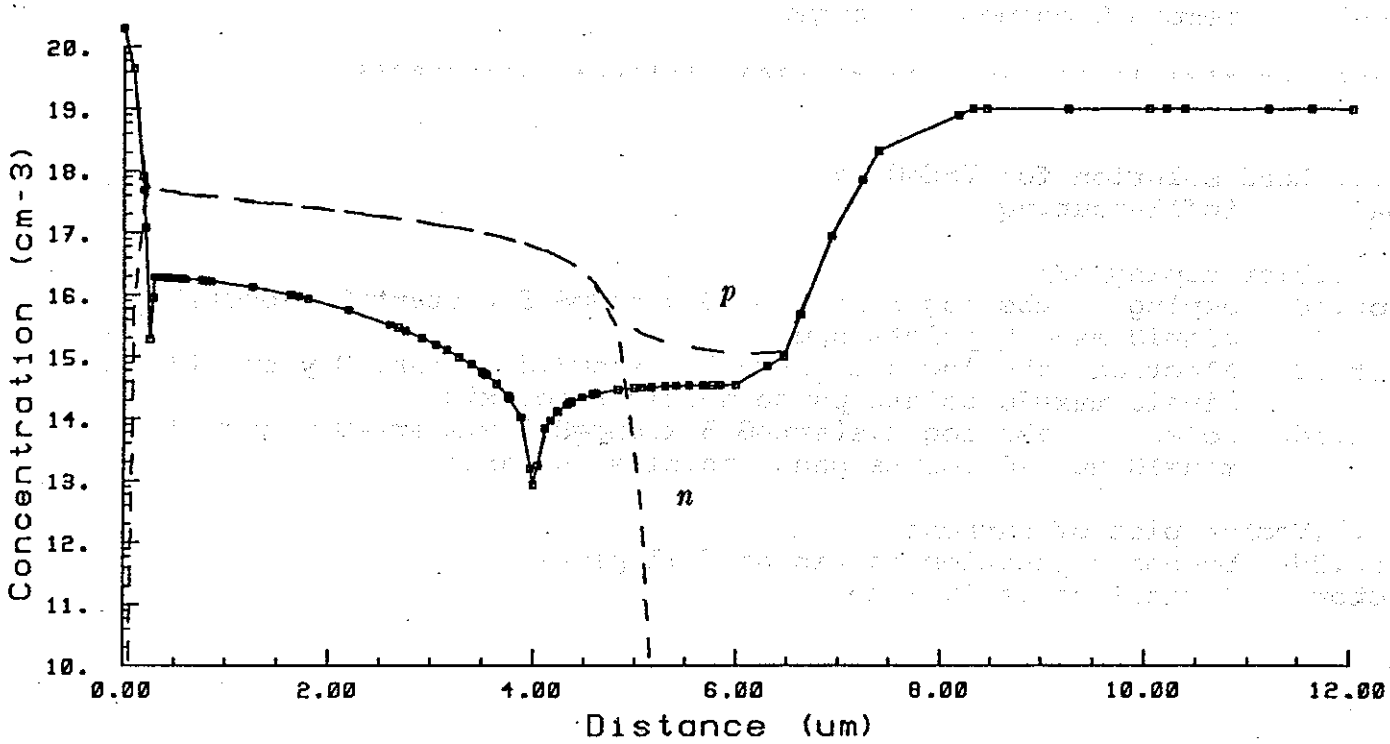


Fig. 5.46. Doping, electron and hole concentrations vertically through the p^+ emitter for high-level injection conditions ($V_{n-well} = 4.1$ volts).

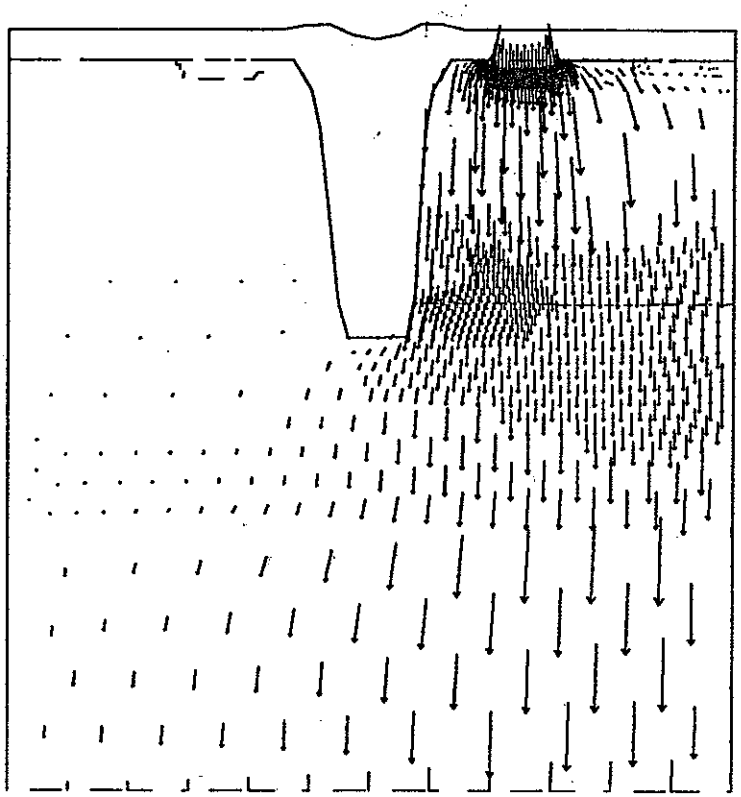


Fig. 5.47. Two-dimensional current vector plot for $V_{n-well} = 4.1$ volts. The length of each vector is proportional to the magnitude of the total current flowing through a plane associated with each node.

Appendix A

PISCES-II Input Specification

Overview 1
Check 3
Comment 4
Contact 5
Contour 7
Doping 9
Electrode 14
Eliminate 15
End 17
Extract 18
Load 20
Log 22
Material 23
Mesh 25
Mathod 28
Models 33
Options 35
Plot.1d 37
Plot.2d 40
Print 42
QF 44
Region 45
Regrid 47
Solve 50
Spread 54
Symb 57
Title 59
Vector 60
X.Mesh 62

Overview

1. Card format

PISCES-II takes its input from a user specified disk file. The input is read by GENII, the same input processor that is used in SUPREM and other Stanford programs. Each line is a particular statement, identified by the first word on the card. The remaining parts of the line are the parameters of that statement. The words on a line are separated by blanks or tabs. If more than one line of input is necessary for a particular statement, it may be continued on subsequent lines by placing a plus sign as the first non-blank character on the continuation lines. Parameter names do not need to be typed in full; only enough characters to ensure unique identification is necessary.

Parameters may be one of three types : numerical, logical or character. Numerical parameters are assigned values by following the name of the parameter by an equal sign and the value. Character parameters are assigned values by following the name of the parameter by an equal sign and a sequence of characters. The first blank or tab delimits the string. Logical values can be set by following the name with an equal sign and the words yes/true or no/false. Alternatively, a logical flag which appears without assigment is taken to have the value true.

In the card descriptions which follow, the letters required to identify a parameter are printed in upper case, and the remainder of the word in lower case. A phrase enclosed in angle brackets <...> represents a parameter list to be explained in further detail.

2. Card sequence

The order of occurence of cards is significant in some cases. Be aware of the following dependencies:

→ The mesh card must precede all other cards, except title and comment.

→ When defining a rectangular mesh, the order of specification is

Mesh
X.Mesh *(all)*
Y.Mesh *(all)*
Eliminate
Spread
Region
Electrode

Eliminate and spread cards are optional but if they occur they must be in that order.

→ Doping and QF cards must follow directly after the mesh definition

→ Before a solution, a symbolic factorization is necessary. Unless solving for the equilibrium condition, a previous solution must also be loaded to provide an initial guess.

- → Physical parameters **may not** be changed using the material, contact or model cards after the first solve or load card is encountered. The material and contact cards precede the model card.
- → A Plot.2d must precede a contour plot, to establish the plot bounds.
- → Plot.2d, Plot.1d, Regrid or Extract cards which access solution quantities ($\psi, n, p, \phi_n, \phi_p$, currents, recombination) must be preceded by a load or solve card to provide those quantities.

The CHECK card

1. Syntax

CHeck <file specification>

2. Description

The CHECK card compares a specified solution against the current solution, returning the maximum and average difference in electrostatic and quasi-fermi potentials. The check card is particularly useful for comparing solutions that have been obtained on different generations of regrids.

3. Parameters

<file specification>

Infile = <filename>
Mesh = <filename>

INFILE specifies the name of the solution file to compare. MESH is the name of the file containing the mesh for that solution.

4. Examples

Compare solution "sol1" obtained using mesh "mesh1" against the current solution.

```
CHECK   INFILE=sol1  MESH=mesh1
```

The COMMENT card

1. Syntax

COMment <character string>

or

$ <character string>

2. Description

The COMMENT card allows comments to be placed in the PISCES input file. PISCES ignores the any information on the COMMENT card.

3. Examples

```
$  **** This is a comment - wow! ****
```

The CONTACT card

1. Syntax

CONTAct <number> <specification>

2. Description

The CONTACT card defines the physical parameters of an electrode. If no contact card is supplied for an electrode it is assumed to be neutral.

3. Parameters

<number>

NUmber = <integer>

or

ALL

The number must be that of a previously defined electrode. Using ALL instead of <number> defines the same properties for all electrodes.

<specification>

Workfunction - One of :

Material	*Work function used*
Neutral	(calculated from doping)
ALUminum	4.17
P.polysilicon	4.17+*Egap*
N.Polysilicon	4.17
MOLybdenum	4.53
TUNgsten	4.63
MO.Disilicide	4.80
TU.Disilicide	4.80

or

Workfunction = <real>

The work function is set to the above values for the standard materials, or to the given value. The value is interpreted in volts.

Schottky contact :

Surf.rec	=	<logical>	(default false)
VSURFN	=	<real>	Surface recombination rate for electrons
VSURFP	=	<real>	Surface recombination rate for holes

If SURF.REC is specified, finite surface recombination velocities are used at the respective contact. The defaults for VSURFN and VSURFP are calculated using equations 2.45 and 2.46.

4. Examples

Define all electrodes except number 2 to be neutral, and number 2 is aluminum. The second card overrides the first.

```
CONTACT ALL NEUTRAL
CONTACT NUM=2 ALUM
```

The CONTOUR card

1. Syntax

CONTOur <plotted quantity> <range definition> <control>

2. Description

The CONTOUR card plots contours on a plotted two-dimensional area of the device, as specified by the most recent PLOT.2D card.

3. Parameters

<plotted quantity>

One of:

POTential	=	<logical>	Mid-gap potential
QFN	=	<logical>	Electron quasi-fermi level
QFP	=	<logical>	Hole quasi-fermi level
Doping	=	<logical>	Doping
Electrons	=	<logical>	Electron concentration
Holes	=	<logical>	Hole concentration
NET.CHarge	=	<logical>	Net charge concentration
NET.CArrier	=	<logical>	Net carrier concentration
J.Conduc	=	<logical>	Conduction current
J.Electr	=	<logical>	Electron current
J.Hole	=	<logical>	Hole current
J.Displa	=	<logical>	Displacement current
J.Total	=	<logical>	Total current
E.Field	=	<logical>	Electric field
Recomb	=	<logical>	Net recombination

The above parameters specify the quantity to be plotted. For vector quantities the magnitude is plotted. **N.B.** Model dependent parameters (current and recombination) are calculated with the models currently defined, ***not*** with the models that were defined when the solution was computed. This allows the display of, for instance, Auger and Shockley-Read-Hall components of recombination separately. For consistent values of current, the models used in the solution should be specified. The quantity to be plotted has no default.

<range definition>

MIn.value	=	<real>
MAx.value	=	<real>
Del.value	=	<real>

MIN.VALUE and MAX.VALUE specify the minimum and maximum contours to be plotted. DEL.VALUE specifies the interval between contours. There is no default! If the plot is logarithmic, the minimum and maximum should be given as the logarithmic bounds.

<control>

LIne.type	=	<integer>	(default is 1)
Logarithm	=	<logical>	(default is false)
ABsolute	=	<logical>	(default is false)
PAuse	=	<logical>	(default is false)

LINE.TYPE defines the plot line type. ABSOLUTE specifies that the absolute value of the variable be taken. For rapidly varying quantities, the LOGARITHM is often more revealing. Since many of the quantities may become negative, PISCES-II actually uses

$$\log(x)=\text{sign}(x) \times \log_{10}(1+|x|)$$

to avoid overflow. To get the true logarithm of a quantity, specify ABSOLUTE and LOGARITHM - the absolute is taken first and there is no danger of negative arguments. The PAUSE option causes PISCES to stop at the end of the plot so that a hardcopy may be made before continuing. Execution can be resumed by hitting a carriage return.

4. Examples

The following plots the contours of potential from -1 volts to 3 volts in steps of .25 volts:

```
CONTOUR  POTEN  MIN=-1  MAX=3  DEL=.25
```

In the next example, the log of the doping concentration is plotted from 1.0e10 to 1.0e20 in steps of 10. By specifying ABSOLUTE, both the n-type and p-type contours are shown.

```
CONTOUR  DOPING  MIN=10  MAX=20  DEL=1  LOG  ABS
```

The DOPING card

1. Syntax

DOping <profile type> <region> <location> <profile specification> <save>

2. Description

The DOPING card dopes selected regions of the device.

3. Parameters

<profile type>

One of the following types must be selected.

Gaussian
Uniform
SUprem3

<location>

Parameter			DEFAULT		
			(Uniform)	(Gaussian or Suprem3)	
				(x-direction)	(y-direction)
X.Left	=	<real>	$-\infty$	*SP*	$-\infty$
X.Right	=	<real>	∞	X.LEFT	∞
Y.Top	=	<real>	$-\infty$	$-\infty$	*SP*
Y.Bottom	=	<real>	∞	∞	Y.TOP

The box given by the X and Y bounds locates the profile within a device, and defines an area (or line) where the profile is constant. Outside this area, it falls off along the principal axis according to the profile type, and along the lateral axis according to the lateral specifications. The default bounds of the box are chosen depending on the type and principal direction of the profile. In the Gaussian/Suprem case, the bounds default to a line perpendicular to the principal axis and located at the peak/start of the profile, respectively. This is denoted by the entry SP in the above table.

<region>

REgion = <integer>

Region number where doping is to be added (optional). Multiple regions may be included by concatenating their region numbers into a single integer. If no region is specified, the entire semiconductor portion of the device is used.

If <profile type>=Suprem3, the following profile specifications are relevant:

<Input filename>

INFile = <filename>

<dopant>

BOron	=	<logical>
PHosphorus	=	<logical>
ARsenic	=	<logical>
ANtimony	=	<logical>

The selected dopant profile will be extracted from the SUPREM-III save file.

<Two-dimensional spread>

Parameter			Default
DIrection	=	x or y	y
STart	=	<real>	0
RAtio.Lateral	=	<real>	1.0

This group of parameters specifies where to locate the one-dimensional profile in the 2-dimensional device and how to extend it to the second dimension. DIRECTION is the axis along which the profile is directed. START is the depth in the specified direction where the profile should start, and should normally be at the surface. The lateral profile is assumed to have the same form as the principal profile, but shrunk/expanded by the factor RATIO.LATERAL. The defaults for the location box are set up as a line, parallel to the surface, and located at START.

If <profile type>=Gaussian, the following profile specifications are relevant:

<profile>

COncentration	=	<real>	(no default)	(cm^{-3})
Junction	=	<real>	(no default)	(μm)

or

DOSe	=	<real>	(no default)	(cm^{-2})
CHaracteristic	=	<real>	(no default)	(μm)

or

COncentration	=	<real>	(no default)	(cm^{-3})
CHaracteristic	=	<real>	(no default)	(μm)

and one of :

N.type/DONor	=	<logical>	(no default)
P.type/ACceptor	=	<logical>	(no default)

and any combination of :

RAtio.Lateral	=	<real>	(default: 1)	
Erfc.Lateral	=	<logical>	(default: false)	
PEak	=	<real>	(default: 0)	(μm)
DIrection	=	<character>	(default: y)	

These parameters govern the profile outside the constant box. DIRECTION defines the principal axis of the profile. CONCENTRATION is the peak concentration, DOSE the total dose. JUNCTION is the location of the junction and must be in silicon, outside the constant box; CHARACTERISTIC is the principal characteristic length. The peak concentration and principal characteristic length are computed from the given combination of the first four parameters. When JUNCTION is used, PISCES-II computes the characteristic length by examining the doping at a point half way between the ends of the constant box and at the given depth; if some other lateral position is desired for the computation, use the parameter SLICE.LATERAL=<real>. The lateral diffusion is governed by the .LATERAL parameters; the lateral profile may be an error function instead of gaussian, and its characteristic length is the product of RATIO.LATERAL and the principal characteristic length. PEAK specifies the position of the peak. The defaults for the constant box are set up as a line, parallel to the surface and located at PEAK.

If <type>=UNIFORM the following parameters are relevant:

<concentration>

COncentration	=	<real>
N.type/DONor	=	True or False
P.type/ACceptor	=	True or False

Concentration is the value of the uniform doping level. It should be given in units of atoms/cm^3 and be positive. The polarity is given by the logical parameters. Doping is introduced in the intersection of the box and the region selected. The default box is set up to include the entire region.

<save>

OUTFile = <filename>

The save option allows the user to save a machine-readable copy of all the DOPING (and QF) cards in a file. The first DOPING card should have the OUTFILE parameter, so that the doping information on it and all subsequent DOPING cards are saved in that file. The file can be reread after regrid to calculate the doping on the new grid.

4. Examples

A one-dimensional diode with substrate doping 10^{16} cm^{-3} and Gaussian profile.

```
DOP    UNIF CONC=1E16 P.TYPE
DOP    GAUSS CONC=1E20 JUNC=0.85 N.TYPE PEAK=0
```

An n-channel MOSFET with Gaussian source and drain. Because the default X.RIGHT is $+\infty$, for the source we must limit the constant part to X.RIGHT=4, and conversely for the drain. Thus the profile has a constant part along the surface, falls off as an error function towards the gate, and as a gaussian in the direction of the bulk. In both cases, the vertical junction is at $1.3\mu m$.

```
DOP   UNIF CONC=1E16 P.TYPE
DOP   GAUSS CONC=9E19 N.TYPE
+     X.RIGHT=4 JUNC=1.3 R.LAT=0.6 ERFC.LAT
DOP   GAUSS CONC=9E19 N.TYPE
+     X.LEFT=12 JUNC=1.3 R.LAT=0.6 ERFC.LAT
```

Read a Suprem bipolar profile and add it to a uniform substrate concentration. Add doping only to those points lying in region 1.

```
COM    *** SUBSTRATE ***
DOP    REGION=1 UNIF CONC=1E16 N.TYPE
COM    *** BASE ***
DOP    REGION=1 SUPREM    BORON R.LAT=0.7 INF=plt3.out1
+      START=0
COM    *** EMITTER ***
DOP    REGION=1 SUPREM    PHOS  R.LAT=0.8 INF=plt3.out1
+      X.LEFT=12.0 X.RIGHT=13.0 START=0
```

The ELECTRODE card

1. Syntax

ELEctrode <number> <position>

2. Description

The ELECTRODE card specifies the location of electrodes in a rectangular mesh.

3. Parameters

<number>

Number = <integer>

There may be up to ten electrodes, numbered 0 to 9. They may be assigned in any order.

<location>

IX.Low = <integer>
IX.High = <integer>
IY.Low = <integer>
IY.High = <integer>

Nodes having x and y indices between IX.LOW and IX.HIGH and between IY.LOW and IY.HIGH respectively are designated electrode nodes.

4. Examples

Define a typical back-side contact.

ELEC N=1 IX.LOW=1 IX.HIGH=40 IY.LOW=17 IY.HIGH=17

The ELIMINATE card

1. Syntax

ELIMINATE <range> <direction>

2. Description

The ELIMINATE card terminates mesh points along lines in a rectangular grid.

3. Parameters

<direction>

X.direction = <logical>
Y.direction = <logical>

These parameters determine whether to eliminate points along vertical or horizontal lines. One must be chosen.

<range>

IX.Low = <integer>
IX.High = <integer>
IY.Low = <integer>
IY.High = <integer>

Points along every second line within the chosen range is removed. Successive eliminations of the same range remove points along every fourth, eighth line, and so on. For horizontal elimination, the vertical bounds should be decreased by one at each re-elimination of the same region, and conversely for vertical eliminations.

4. Examples

Points along vertical lines between 10 and 20 are removed.

```
ELIM     Y.DIR  IY.LO=10  IY.HI=20  IX.LO=1  IX.HI=8
ELIM     Y.DIR  IY.LO=10  IY.HI=20  IX.LO=1  IX.HI=7
```

The END card

1. Syntax

ENd

2. Description

The END card specifies the end of a set of PISCES input cards. The END card may be placed anywhere in the input deck; all input lines below the occurrence of the END card will be ignored. If an END card is not included, all cards in the input file are processed.

The EXTRACT card

1. Syntax

EXtract <variable> <bounds>

2. Description

The EXTRACT card extracts selected electrical data from the solution.

3. Parameters

<variable>

NET.CHar	=	<logical>	Integrated net charge
NET.CArr	=	<logical>	Integrated carrier concentration
ELectron	=	<logical>	Integrated electron concentration
Hole	=	<logical>	Integrated hole concentration
Metal.Ch	=	<logical>	Integrated charge on a contact
N.Resist	=	<logical>	n-Resistance of a cross section
P.Resist	=	<logical>	p-Resistance of a cross section
N.Current	=	<logical>	n-current through an electrode
P.Current	=	<logical>	p-current through an electrode

The net carrier, charge, electron or hole concentrations can be integrated over a section of a device. The charge on a part of an electrode can be calculated, as can the current through that part. This is useful for capacitance studies, in conjunction with the difference mode of the load card. The resistance of a cross sectional structure, for instance a diffused line, can be calculated.

<bounds>

X.MIn	=	<real>
X.MAx	=	<real>
Y.MIn	=	<real>
Y.MAx	=	<real>
COntact	=	<integer>

Only nodes falling within the rectangle X.Min-Y.Max are included in the integrations. The default bounds include the entire device. For electrode quantities (current and metal charge) a contact must be chosen; only nodes falling within the bounds and belonging to the contact are included in the integration.

4. Examples

The following extracts the resistance of a p-type line diffused into a lightly doped n substrate. Since the p-conductivity of the substrate is negligible, the bounds of the integration can include the whole device.

```
EXTRACT   P.RESIST
```

In the next example, the charge on the lower surface of a gate electrode is integrated. There is 0.05 μm of gate oxide on the surface, which is at y=0.

```
EXTRACT   ME.CHARGE CONT=1 X.MIN=-2.0 X.MAX=2.0
+         Y.MAX=-0.0499 Y.MIN=-0.0501
```

The LOAD card

1. Syntax

Load <solution files>

2. Description

The LOAD card loads previous solutions from files for plotting or as initial guesses to other bias points.

3. Parameters

<solution files>

INFile (or IN1file)	=	<filename>	
IN2file	=	<filename>	
Outdiff	=	<filename>	
Differ	=	<logical>	(default is false)

The INFILE (or IN1FILE) and IN2FILE parameters specify input files names for solution data and may be up to 20 characters in length. INFILE (or IN1FILE) and IN2FILE represent a present and previous solution respectively. If only one solution is to be loaded (for plotting or as a single initial guess using the PREVIOUS option on the SOLVE card) then INFILE should be used. If two input files are needed to perform an extrapolation for an initial guess (i.e., the PROJECT option on the SOLVE card), IN1FILE and IN2FILE should be used. The solution in IN2FILE is the first to be lost when new solutions are obtained. The difference between two solutions can be analyzed by reading in both with the mode DIFFER set. The difference is stored; this solution may not be used as an initial guess, or for any purpose other than plotting or extracting data. The difference solution may also be stored in another file using the parameter OUTDIFF.

4. Examples

The following specifies that a single solution file called SOL.IN should be loaded.

```
LOAD    INF=SOL.IN
```

In the next example, two solutions are loaded. The present solution is to SOL1.IN and the previous solution is SOL2.IN. We intend to use SOL1.IN and SOL2.IN to project an initial guess for a third bias point.

```
LOAD IN1F=SOL1.IN IN2F=SOL2.IN
```

Finally, two solutions are loaded, and the difference calculated and stored in a third file.

```
LOAD IN1F=SOL1.IN IN2F=SOL2.IN DIFF OUTD=SOL2-1
```

The LOG card

1. Syntax

LOG <file specification>

2. Description

The LOG card allows the I-V characteristics of a run to be logged to disk. Any I-V data subsequent to the card is saved. If a log file is already open, it is closed and a new file opened.

3. Parameters

<file specification>

Outfile = <filename>

The only parameter is the filename to store logged data to.

4. Examples

Save the I-V data in a file called IV1.

LOG OUTF=IV1

The MATERIAL card

1. Syntax

Material <region> <material definitions>

2. Description

The material card associates physical parameters with the materials in the mesh. Many of the parameters are default for standard materials.

3. Parameters

<region>

NUmber = <integer>
or
Region = <integer>

NUMBER (or REGION) specifies the region number to which these parameters apply. Only one set of semiconductor parameters is allowed. Therefore, if the region specified is a semiconductor region, all other semiconductor regions (if there are any) will be changed as well.

<material definitions>

EG300	=	<real>	: Energy gap at 300K (eq. 2.16)	(eV)
EGAlpha	=	<real>	: Alpha (eq. 2.16)	
EGBeta	=	<real>	: Beta (eq. 2.16)	
AFfinity	=	<real>	: Electron affinity	(eV)
Permittivity	=	<real>	: Dielectric permittivity	(F/cm)
Vsaturation	=	<real>	: Saturation velocity (eq. 2.34, 2.35)	(cm/s)
MUN	=	<real>	: Low-field electron mobility	(cm^2/s)
MUP	=	<real>	: Low-field hole mobility	(cm^2/s)
G.surface	=	<real>	: surface mobility reduction (eq. 2.33)	
TAUP0	=	<real>	: SRH Electron lifetime (eq. 2.6, 2.8)	(s)
TAUN0	=	<real>	: SRH Hole lifetime (eq. 2.6, 2.9)	(s)

...continued...

NSRHN	=	<real>	: SRH conc. parameter - electrons (eq. 2.8)	(cm^{-3})
NSRHP	=	<real>	: SRH conc. parameter - holes (eq. 2.9)	(cm^{-3})
ETrap	=	<real>	: Trap level = $E_t - E_i$ (eq. 2.6)	
AUGN	=	<real>	: Auger coefficient (c_n) (eq. 2.7)	(cm^6/s)
AUGP	=	<real>	: Auger coefficient (c_p) (eq. 2.7)	(cm^6/s)
NC300	=	<real>	: Conduction band density at 300K (eq. 2.17)	(cm^{-3})
NV300	=	<real>	: Valence band density at 300K (eq. 2.18)	(cm^{-3})
ARICHN	=	<real>	: Richardson constant for electrons (eq. 2.45)	
ARICHP	=	<real>	: Richardson constant for holes (eq. 2.46)	

Defaults:

Semiconductors

Constant	Silicon	Gallium Arsenide	Arbitrary
Energy gap (300K)	1.08	1.43	0.0
Alpha	4.73×10^{-4}	5.405×10^{-4}	0.0
Beta	636.	204	0.0
Electron affinity	4.17	4.07	0.0
Permittivity	11.8	10.9	0.0
Saturation velocity	(eq. 2.36)	(eq. 2.37)	0.0
Electron mobility	1000	5000	0.0
Hole mobility	500	400	0.0
Surface mobility reduction	1.0	1.0	0.0
SRH Electron lifetime	1.0×10^{-7}	1.0×10^{-7}	0.0
SRH Hole lifetime	1.0×10^{-7}	1.0×10^{-7}	0.0
Auger coefficient (n)	2.8×10^{-31}	2.8×10^{-31}	0.0
Auger coefficient (p)	9.9×10^{-32}	9.9×10^{-32}	0.0
Cond band density (300K)	2.8×10^{19}	4.7×10^{17}	0.0
Val band density (300K)	1.04×10^{19}	7.0×10^{18}	0.0
Eff Richardson const (n)	110	6.2857	0.0
Eff Richardson const (p)	30	105	0.0
SRH conc. parameter (n)	5.0×10^{16}	5.0×10^{16}	0.0
SRH conc. parameter (p)	5.0×10^{16}	5.0×10^{16}	0.0
Trap level	0.0	0.0	0.0

Insulators

Constant	Silicon dioxide	Silicon nitride	Sapphire	Arbitrary
Permittivity	3.9	7.5	12.0	0.0

4. Examples

The following defines SRH lifetimes and concentration-independent low-field mobilities for all the semiconductor regions within the device (all the other parameters are assumed to be their default, consistent with the semiconductor type chosen):

```
MATERIAL  TAUN0=5.0e-6 TAUP0=5.0e-6 MUN=3000
MUP=500
```

The MESH card

1. Syntax

MESh <type> <output files> <smoothing key>

2. Description

The mesh card either initiates the mesh generation phase or reads a previously generated mesh.

3. Parameters

<type>: One of

<Previous>

PRevious = True or False
INFile = <filename>

This is the default; it reads a previously generated mesh from a binary save file.

<Rectangular>

REctangular = True or False
NX = <integer>
NY = <integer>

This initiates the generation of a rectangular mesh. NX is the number of nodes in the x-direction, NY the number in the y-direction.

<Geometry>

GEometry = True or False
INFile = <filename>

This reads a mesh generated by an external grid editor.

<Output files>

OUTFile = <filename>
OUT.asc = <filename>
Flip.y = True or False

Outfile is the binary output file to be read by a later run. Out.asc is an ascii output file intended to be read by an external grid editor. See appendix B for details of the format. When writing a file for the grid generator the direction of the y coordinate may be reversed, using the FLIP.Y parameter.

<smoothing key>

SMooth.Key = <integer>

This causes mesh smoothing as described in section 4.6. The digits of the integer are read in reverse order and decoded as follows:

1 Triangle smoothing, maintaining all region boundaries fixed.
2 Triangle smoothing, maintaining only material boundaries.
3 Node averaging.

Options 1 and 3 are the most common; 2 is used only if a device has several regions of the same material and the border between the different regions is unimportant.

4. Examples

Initiate a rectangular mesh and request it to be stored in mesh1.pg :

```
MESH  RECTANGULAR NX=40 NY=17 OUTF=mesh1.pg
```

Read a previously generated mesh and generate an ascii file for a grid editor (the y axis is inverted because the grid editor obeys the convention that positive y is upward, while PISCES follows the semiconductor convention of positive y being into the bulk) :

```
MESH  INF=mesh1.pg OUT.ASC=mesh1.pa FLIP
```

Read a geometry file, smooth the mesh, and store the binary file for a later run:

```
MESH   GEOM  INF=geom1  SMOOTH.K=13131  OUTF=mesh1.pg
```

The smoothing does several averaging and flipping steps. The digits are read in reverse order, so that the flipping comes first, followed by node averaging, and so on.

The METHOD card

1. Syntax

METhod <general parameters> <method-dependent parameters>

2. Description

The method card sets parameters associated with the particular solution algorithm chosen on the symbolic card. There can be more than one method card in a single simulation, so that parameters can be altered. The default values of the parameters are used on the first occurrence of the method card; subsequent method cards only alter those coefficients specified. (See Chapter 3 for an overview of the uses of the parameters.)

3. Parameters

<general parameters>

Parameter			*Default*
ITlimit	=	<integer>	20
Xnorm	=	True or False	true
RHsnorm	=	True or False	false
P.tol	=	<real>	Depends on choice of norm
C.tol	=	<real>	Depends on choice of norm
LImit	=	<logical>	false
PRint	=	<logical>	false
COntinuation	=	<logical>	false
AContinuation	=	<real>	0.5

The above parameters are used to determine the convergence of the solution methods. ITLIMIT is the maximum number of allowed outer loops (i.e., Newton loops or Gummel continuity iterations). P.TOL and C.TOL are the termination criteria for the Poisson and continuity equations, respectively. If XNORM is chosen as the error norm, the Poisson updates are measured in units of kT/q, and carrier updates are measured relative to the local carrier concentration. In this case the default value for both P.TOL and C.TOL is 1×10^{-5}. If RHSNORM is selected, the Poisson error is measured in $C/\mu m$ and the continuity error in $A/\mu m$. P.TOL then defaults to $1\times10^{-26}C/\mu m$, and C.TOL to $5\times10^{-18}A/\mu m$.

LIMIT indicates that the convergence criterion should be ignored, and iterations are to proceed until ITLIMIT is reached. PRINT prints the terminal fluxes/currents after each continuity iteration; if this parameter is not set, the terminal fluxes/currents are only printed after the solution converges. CONTINUATION specifies that if a solution process starts to diverge, the electrode bias steps taken from the initial guess are reduced by the multiplicative factor ACONTINUATION.

<method-dependent parameters>

For the Gummel method:

The following parameters are for damping the Poisson updates.

DVlimit	=	<real>	(default is 0.1)

or:

DAMPEd	=	<logical>	(default is false)
DElta	=	<real>	(default is 0.5)
DAMPLoop	=	<integer>	(default is 10)
DFactor	=	<real>	(default is 10.0)

The DVLIMIT parameter limits the maximum ψ update for a single loop. The DAMPED parameter indicates the use of a more sophisticated damping scheme proposed by Bank and Rose (this is the recommended option, particularly for large bias steps). The remaining damping parameters are only interpreted if the DAMPED parameter is specified. DELTA is the threshold for determining the damping factor for $\Delta\psi$ and must be between 0 and 1. DAMPLOOP is the maximum number of damping loops allowed to find a suitable damping coefficient. DFACTOR is a factor which serves to increase the initial damping coefficient for the next Newton loop.

The following parameters select acceleration methods for the Gummel iteration.

MULtipoisson	=	<logical>	(default is true)
Singlepoisson	=	<logical>	(default is false)
ICCG	=	<logical>	(default is false)
LU1cri	=	<real>	(default is 3×10^{-3})
LU2cri	=	<real>	(default is 3×10^{-2})
ACCEleration	=	<logical>	(default is false)
ACCSTArt	=	<real>	(default is 0.3)
ACCSTOp	=	<real>	(default is 0.6)
ACCSTEp	=	<real>	(default is 0.04)

The first two parameters specify how the Poisson equation iterations are to be performed. The MULTIPOISSON option indicates the standard Gummel iterative procedure where the continuity equation is only treated after Poisson has fully

converged for every Gummel loop. The SINGLEPOISSON option indicates that only a single Poisson iteration is to be performed per Gummel loop. The ICCG parameter chooses whether or not to use iteration to solve the multi-Poisson loops. It should be set whenever doing multi-Poisson. The next two parameters specify how much work is done per Poisson loop (cf. Section 3.3). The inner norm is required to decrease by at least LU1cri before returning, or to reach a factor of LU2cri below the projected Newton error, whichever is the smaller. (If the inner norm is allowed to exceed the projected Newton error, quadratic convergence is lost). The remaining parameters deal with an acceleration method for attaining faster overall convergence in the single-Poisson mode. The ACCELERATION option specifies that acceleration is to be used. ACCSTART is the starting value of the acceleration parameter, ACCSTOP is the final (limiting) value of the acceleration parameter and ACCSTEP is the step to be added to the value of the acceleration parameter after each iteration (Carré[3.6]).

For the direct Newton method:

AUtonr	=	<logical>	(default is false)
NRcriterion	=	<real>	(default is 0.1)

The above parameters are for implementing an automated Newton-Richardson procedure which attempts to reduce the number of LU decompositions per bias point. The AUTONR option indicates that this algorithm is to be used. NRCRITERION is the ratio by which the norm from the previous Newton loop must go down in order to be able to use the same Jacobian (i.e., LU decomposition) for the current Newton loop. This is strongly recommended for full Newton iteration, but is of little advantage for the iterative methods.

For the iterative Newton methods:

			Block default	Knot default
SLotboom	=	True or False	false	false
SCale	=	True or False	false	false
Maxinner	=	<integer>	16	40
LU1cri	=	<real>	1×10^{-3}	1×10^{-5}
LU2cri	=	<real>	1×10^{-2}	1×10^{-3}
Back.vec	=	<integer>	6	6
Knot.max	=	<integer>	-1	-
Fill.cri	=	<real>	-	-1

These parameters govern the preconditioned conjugate residual solvers. SLOTBOOM specifies whether the matrix is to be expressed in terms of the Slotboom variables or in terms of the carrier concentrations. If SCALE is chosen, the matrix is scaled by its diagonal; this helps prevent overflow. MAXINNER is the number of inner loops allowed per Newton loop. The next two criteria (LU1/2CRI) have a similar

meaning to those used in the ICCG inner loop. BACK.VEC is the number of previous vectors retained for orthogonality purposes (Section 3.3.2.1). Storage is allocated for a maximum of ten in the Block method, and for six in the Knot method. FILL.CRI is used to control the amount of fill-in in the partial knot factorization. With a negative criterion, no fill is allowed. With a positive number α, a potential fill-in in row i and column j is rejected if it is smaller than α times the pivot in position i and j. Thus the smaller the value, the more fill allowed, and with a value of zero an exact factorization is performed. The use of fill is not yet wholly characterized. In general, the more fill, the more stable the iteration, but at a cost of more storage. The criterion should be between 0 and 1. Finally, KNOT.MAX is used for a combined Block/Knot iteration. The block iteration is usually much faster, but can be unstable in the early stages. Thus several (chosen by KNOT.MAX) Newton iterations are sometimes solved with knots before switching over to the block iteration. The default is not to use any knot iterations.

4. Examples

The following specifies that for a simulation using the Gummel method (as previously specified by an appropriate symbolic card), that damping is to be employed and the Poisson error tolerance should be 1×10^{-30} coul/μm. Note that because XNORM defualts to true, XNORM must be turned off to use the rhs norm as a convergence criterion. If XNORM=FALSE had not been specified, the rhs norm and the update norm would have both been printed, but only the update norm would have been used to determine convergence.

```
METHOD  DAMPED P.TOL=1.e-30 RHSNORM
XNORM=FALSE
```

The next example would usually follow a Block-Newton choice on the symbolic card. Slotboom variables are essential for the block iteration, and the scaling reduces the numerical range.

```
METHOD  SLOTBOOM SCALED MAXINNER=20
```

The next example illustrates the continuation feature. The first SOLVE card solves for the initial, zero bias case. On the second SOLVE card, we attempt to solve for V2=3 volts V3=5 volts. If such a large bias change caused the solution algorithms to diverge for this bias point, the bias steps would be multiplied by ACONTINUATION (0.5); i.e., an intermediate point (V2=1.5 volts, V3=2.5 volts) would be attempted before trying to obtain V2=3 volts and V3=5 volts again. If the intermediate point can not be solved for either, PISCES will continue to reduce the bias step (the next would be V2=0.75 volts and V3=1.25 volts) up to 4 times. Note also that the intermediate solutions will be saved in

output files in a manner similar to voltage stepping using the VSTEP parameter on the SOLVE card; i.e., if two intermediate steps to V2/V3=3/5 volts were required, they would be stored in "outa" and "outb" while V2/V3=3/5 volts would be stored in "outc".

```
METHOD   CONT ACONT=0.5
SOLVE    INIT
SOLVE    V2=3 V3=5 OUTFILE=outa
```

The MODELS card

1. Syntax

MOdels <model flags> <numerical parameters>

2. Description

The model card sets the temperature for the simulation and specifies model flags to indicate the inclusion of various physical mechanisms and models.

3. Parameters

<model flags>

Srh	=	<logical>	(default is false)
CONSrh	=	<logical>	(default is false)
AUger	=	<logical>	(default is false)
BGn	=	<logical>	(default is false)
CONMob	=	<logical>	(default is false)
FLdmob	=	<logical>	(default is false)
BOltzmann	=	<logical>	(default is true)
FErmidirac	=	<logical>	(default is false)
Incomplete	=	<logical>	(default is false)

CONSRH/SRH and AUGER specify Shockley-Read-Hall (eq. 2.6) and Auger recombination (eq. 2.7) respectively. SRH uses fixed lifetimes and CONSRH uses concentration-dependent lifetimes. (eq. 2.8, 2.9). BGN is band-gap narrowing (eq. 2.20). CONMOB is concentration dependent mobility (currently only silicon and gallium arsenide have been implemented). FLDMOB specifies a lateral field-dependent model (eq. 2.34, 2.35). BOLTZMANN and FERMIDIRAC indicate the carrier statistics to be used (eq. 2.13, 2.14 and 2.10, 2.11), while INCOMPLETE indicates that incomplete-ionization of impurities should be accounted for (eq. 2.31, 2.32).

<Numerical parameters>

Temperature	=	<real>	(default is 300K)
B.Electrons	=	<real>	(default is 2)
B.Holes	=	<real>	(default is 1)
E0	=	<real>	(default is 4×10^4V/cm)

TEMPERATURE should be specified in Kelvin units. B.ELECTRONS and B.HOLES are parameters used in the field-dependent mobility expression for silicon (eq. 2.34), while E0 is a parameter used in the field-dependent mobility model for gallium arsenide (eq. 2.35).

4. Examples

The following example selects concentration dependent mobility and SRH recombination. Fermi-dirac statistics are used, and the simulation is specified to be performed at 290K.

```
MODELS  CONMOB SRH FERMI TEMP=290
```

The OPTIONS card

1. Syntax

Options <run control> <plot control>

2. Description

The OPTIONS card sets options for an entire run.

3. Parameters

<run control>

G.DEbug	=	<logical>	(default is false)
N.DEbug	=	<logical>	(default is false)
CPUStat	=	<logical>	(default is false)
CPUFile	=	<character>	(default is pisces.cpu)

G.DEBUG and N.DEBUG print debugging information to the PISCES standard output. G.DEBUG prints general information, N.DEBUG more specifically numerical parameters. CPUSTAT is a flag to indicate that a cpu profile of the solution process is to be printed to the file specified by CPUFILE.

<Plot control>

HP2648	=	<logical>	(default true)
HP2623	=	<logical>	(default false)
Tek4107	=	<logical>	(default false)
X.Screen	=	<real>	(default 10 inches)
Y.Screen	=	<real>	(default 5 inches)
X.Offset	=	<real>	(default 0 inches)
Y.Offset	=	<real>	(default 0 inches)

The first three parameters may be used to change the plotting device. In release II-A, the Hewlett-Packard HP2648 and HP2623 graphics terminals, and the Tektronix 4107 color graphics terminal, are supported. On color graphics terminals, the different line types are implemented as different colors; on the black and white monitors they are implemented as dot and line patterns. X.Screen is the physical width of the screen and Y.Screen is the height. They are set automatically depending on the plot device, but can be altered for special effects (split screen plots, for instance). The offset from the bottom left corner of the screen may be set using X.Offset and Y.Offset.

4. Examples

The following sets up a plot for a Tektronix terminal, using a small centered window. Cpu information is also logged to the default file.

```
OPTIONS     DEV=TEK4107  X.S=6  Y.S=5  X.Off=1  Y.OFF=0.5
+            CPUSTAT
```

The PLOT.1D card

1. Syntax

PLOT.1d <line segment definition> <plotted quantity> <control>

2. Description

The PLOT.1D card plots a specific quantity along a line segment through the device (mode A), or plots an I-V curve of data accumulated so far in the run (mode B).

3. Parameters

<line segment definition>

X.Start or A.X	=	<real>
Y.Start or A.Y	=	<real>
X.End or B.X	=	<real>
Y.END or B.Y	=	<real>

The above parameters define the Cartesian coordinates of the start (A.X,A.Y) and the end (B.X,B.Y) of the line segment along which the specified quantity is to be plotted. The data is plotted as a function of distance from the start (A). The line segment may not be defaulted. It is required in mode A.

<plotted quantity>

One of:

POTential	=	<logical>	Mid-gap potential
QFN	=	<logical>	Electron quasi-fermi level
QFP	=	<logical>	Hole quasi-fermi level
Doping	=	<logical>	Doping
Electrons	=	<logical>	Electron concentration
Holes	=	<logical>	Hole concentration
NET.CHarge	=	<logical>	Net charge concentration
NET.CArrier	=	<logical>	Net carrier concentration
J.Conduc	=	<logical>	Conduction current
J.Electr	=	<logical>	Electron current
J.Hole	=	<logical>	Hole current
J.Displa	=	<logical>	Displacement current

J.Total	=	<logical>	Total current
E.Field	=	<logical>	Electric field
Recomb	=	<logical>	Net recombination

or :

X.AXIS	=	<character>
Y.AXIS	=	<character>
INFile	=	<character>

The above parameters specify the quantity to be plotted. There is no default. In mode A, one of the solution variables is plotted versus distance into the device. For vector quantities, the magnitude is plotted. In mode B one of the terminal currents is plotted against applied bias. X.AXIS selects the contact (V1,V2,...,V9) and Y.AXIS selects the current (I1,...,I9). In a transient simulation, X.AXIS can also be set to TIME. The values plotted are the I-V of the present run, provided a log is being kept (see the LOG card). Alternatively, a different log file can be loaded with INFILE.

<control>

LOgarithm	=	<logical>	(default is false)
Absolute	=	<logical>	(default is false)
NO.Clear	=	<logical>	(default is false)
NO.Axis	=	<logical>	(default is false)
POInts	=	<logical>	(default is false)
PAuse	=	<logical>	(default is false)
LIne.type	=	<integer>	(default is 1)
MIn.value	=	<real>	
MAx.value	=	<real>	
Spline	=	<logical>	(default is false)
NSpline	=	<logical>	(default is 100)
Ascii	=	<character>	
Infile	=	<character>	
Outfile	=	<character>	

ABSOLUTE specifies that the absolute value of the variable be taken. For rapidly varying quantities, the LOGARITHM is often more revealing. Since many of the quantities may become negative, PISCES-II actually uses

$$\log(x) = \text{sign}(x) \times \log_{10}(1 + |x|)$$

to avoid overflow. To get the true logarithm of a quantity, specify ABSOLUTE and LOGARITHM - the absolute is taken first and there is no danger of negative arguments. NO.CLEAR indicates that the screen is not to be cleared before the current plot so that several curves can be plotted on the same axis. NO.AXIS indicates that the axes for the graph are not to be plotted. POINTS marks the data points on the plotted curve. The PAUSE option causes PISCES to stop at the end of the plot so that a hardcopy may be made before continuing. Execution can be resumed by hitting a carriage return. LINE.TYPE specifies the line type for the plotted curve. MIN.VALUE and MAX.VALUE specify minimum and maximum values for the ordinate of the graph; their defaults are found automatically from the data to be plotted. The SPLINE option indicates that spline-smoothing should be performed on the data using NSPLINE interpolated points

(maximum is 500). ASCII is the name of an output file (up to 20 characters long) for printed output of the data points plotted. OUTFILE generates a binary plotfile which can be read by the dplot system program (only useful at Stanford).

4. Examples

The following plots a graph of potential along a straight line from (0.0,0.0) to (5.0,0.0):

```
PLOT.1D   POTEN  A.X=0  A.Y=0  B.X=5  B.Y=0
```

In the next example, the log of the electron concentration is plotted from (1.0,-0.5) to (1.0,8.0) with bounds on the plotted electron concentration of 1.0e10 and 1.0e20. A spline interpolation is performed with 300 interpolated points. The non-spline-interpolated points are marked.

```
PLOT.1D   ELECT LOG  A.X=1  A.Y=-.5  B.X=1  B.Y=8
+           MIN=10  MAX=20  SPLINE  NSPL=300  POINTS
```

In the following example, the current in contact 1 is plotted as a function of contact 2 voltage, then the curve is compared with a previous run.

```
PLOT.1D   X.AXIS=V2  Y.AXIS=I1  MIN=0  MAX=1E-6
PLOT.1D   X.AXIS=V2  Y.AXIS=I1  MIN=0  MAX=1E-6  INF=logf0
+           NO.CLEAR  NO.AXIS
```

The PLOT.2D card

1. Syntax

PLOT.2d <area definition> <plotted quantity> <control>

2. Description

The PLOT.2D card plots quantities in a specified two-dimensional area of the device. A PLOT.2D card is required before performing a contour plot (see CONTOUR card) in order to obtain the plot boundaries.

3. Parameters

<area definition>

X.MIn	=	<real>
X.MAx	=	<real>
Y.MIn	=	<real>
Y.MAx	=	<real>

The above parameters define the rectangular area of the device to be plotted. The default area is a rectangle around the entire device.

<plotted quantity>

Grid	=	<logical>	(default is false)
Crosses	=	<logical>	(default is false)
Boundary	=	<logical>	(default is false)
Depl.edg	=	<logical>	(default is false)
Junction		<logical>	(default is false)

The GRID option plots the grid, including lines delineating elements. CROSSES plots crosses at the locations of grid points. BOUNDARY indicates that the boundaries around the device and between regions are to be plotted. DEPL.EDG indicates that depletion edges are to be plotted (note: depletion edges can only be plotted after a solution is present). The JUNCTION option specifies that the junctions from the doping profiles are to be plotted.

<control>

NO.TIc	=	<logical>	(default is false)
NO.TOp	=	<logical>	(default is false)
NO.Fill	=	<logical>	(default is false)
NO.Clear	=	<logical>	(default is false)
PAuse	=	<logical>	(default is false)
Outfile	=	<character	

NO.TIC indicates that tic marks are not to be included around the plotted area. NO.TOP indicates that tic marks are not to be put on the top of the plotted region. The NO.FILL option will force PISCES to draw the device area plotted to scale; if this option is not specified, the plot will fill the screen, and the triangles will appear distorted. NO.CLEAR specifies that the screen is not to be cleared before plotting. The PAUSE option causes PISCES to stop at the end of the plot so that a hardcopy may be made before continuing. Execution can be resumed by hitting a carriage return. Outfile generates a binary plotfile which can be read by the dplot system program (only at Stanford).

4. Examples

The following plots the entire grid to scale with tic marks:

```
PLOT.2D   GRID  NO.FILL
```

In the next example, the device and region boundaries, junctions and depletion edges are plotted in the rectangular area bounded by $0<x<5\mu$ and $0<y<10\mu$. The plot is allowed to fill the screen and tic marks are not included along the top of the plot.

```
PLOT.2D   X.MIN=0  X.MAX=5  Y.MIN=0  Y.MAX=10
+           JUNCT  BOUND  DEPL  NO.TOP
```

The PRINT card

1. Syntax

PRint <location> <quantity>

2. Description

The PRINT card prints specific quantities at points within a defined area of the device.

3. Parameters

<location>

X.MIn	=	<real>
X.MAx	=	<real>
Y.MIn	=	<real>
Y.MAx	=	<real>

or

IX.LOw	=	<integer>
IX.High	=	<integer>
IY.LOw	=	<integer>
IY.High	=	<integer>

The above parameters define area in which the points of interest lie. X.MIN, X.MAX, Y.MIN and Y.MAX specify an area in physical coordinates (in μm). IX.LOW, IX.HIGH, IY.LOW and IY.HIGH specify the area by the bounding indices (valid only for rectangular meshes). The default area is the entire device.

<quantity>

POints	=	<logical>
Elements	=	<logical>
Geometry	=	<logical>
SOlution	=	<logical>
P.SOL1	=	<logical>
P.SOL2	=	<logical>
Current	=	<logical>
P.CURR1	=	<logical>
P.CURR2	=	<logical>
Que	=	<logical>
P.QUE1	=	<logical>
P.QUE2	=	<logical>
Material	=	<logical>

The above parameters specify the quantities to be plotted. There is no default. POINTS prints node information (coordinates, doping, etc.). ELEMENTS prints information on the triangular elements (number, nodes, material). GEOMETRY prints geometrical information on the triangles. SOLUTION prints the *present* solution (ψ, n, p and quasi-fermi potentials), while P.SOL1 and P.SOL2 print the *previous* two solutions. CURRENT prints currents (electron, hole, conduction, displacement and total) at each node for the present solution; P.CURR1 and P.CURR2 print currents for previous solutions. QUE prints space charge, recombination and electric field for the present solution; P.QUE1 and P.QUE print the same quantities for the previous two solutions. MATERIAL prints material information (permittivity, band-gap, etc.), including the value of the concentration dependent mobility and lifetime (if specified) at each point.

4. Examples

The following prints the physical coordinates, doping and region/electrode information for points along the 10th x grid line, from the 1st to the 20th y grid lines.

```
PRINT  POINTS IX.LO=10 IX.HI=10 IY.LO=1 IY.HI=20
```

In the next example, solution information is printed for
$0 < x < 1\mu m$ and $0 < y < 2\mu m$.

```
PRINT  SOLUTION X.MIN=0 X.MAX=1 Y.MIN=0 Y.MAX=2
```

The QF card

1. Syntax

QF <charge> <location>

2. Description

The QF card introduces fixed charge at an oxide/semiconductor interface.

3. Parameters

<charge>

Concentration = <real>

This is the fixed charge concentration, measured in cm^{-2}.

<location>

X.Left = <real>
X.Right = <real>
Y.Top = <real>
Y.Bottom = <real>

This is the bounding box, measured in μm. Any oxide/semiconductor interfaces found within this region are charged. Note that if an output file is specified on a DOPING card preceding the QF cards, the information on the QF cards (only those which occur after the DOPING card) will also be saved for regrids.

4. Examples

A non-planar surface may be charged by using a box which contains the whole device, provided there is only one interface in the device.

```
QF   X.LEFT=-4 X.RIGHT=4 Y.TOP=-0.5 Y.BOT=4
```

The REGION card

1. Syntax

REGIon <number> <position> <material>

2. Description

The region card defines the location of materials in a rectangular mesh. Every triangle must be defined to be some material.

3. Parameters

<number>

NUmber = <integer>

This parameter selects the region in question. There is a maximum of eight regions in a PISCES structure.

<position>

IX.Low = <integer>
IX.High = <integer>
IY.Low = <integer>
IY.High = <integer>

These parameters are the indices of a box in the rectangular mesh.

<material>

One of:

SILicon
Gaas
SEmiconductor
Oxide or SIO2
NITride or SI3n4
SApphire
Insulator

4. Examples

The following defines a silicon region extending from nodes 1 to 25 in the x direction and nodes 4 to 20 in the y direction :

```
REGION NUM=1 IX.LO=1 IX.HI=25 IY.LO=4 IY.HI=20 SIL
```

Note that region cards are cumulative in effect :

```
REGION NUM=1 IX.LO=4  IX.HI=5  IY.LO=1 IY.HI=20 OXIDE
REGION NUM=1 IX.LO=36 IX.HI=37 IY.LO=1 IY.HI=40 OXIDE
```

defines one region comprised of two separate strips.

The REGRID card

1. Syntax

REGRid <variable> <files> <control>

2. Description

The REGRID card allows refinement of a crude mesh. Any triangle across which the chosen variable changes by more than a specified tolerance, or in which the chosen variable exceeds a given value, is refined.

3. Parameters

<variable>

One of :

Potential	=	<logical>	Mid-gap potential	(Volts)
QFN	=	<logical>	Electron quasi-fermi level	(Volts)
QFP	=	<logical>	Hole quasi-fermi level	(Volts)
DOPIng	=	<logical>	Net doping	(cm^{-3})
Electron	=	<logical>	Electron concentration	(cm^{-3})
Hole	=	<logical>	Hole concentration	(cm^{-3})
NET.CHrg	=	<logical>	Net charge	(cm^{-3})
NET.CArr	=	<logical>	Net carrier concentration	(cm^{-3})
MIn.Carr	=	<logical>	Minority carrier conc.	(cm^{-3})

This parameter selects the discriminatory variable.

<control>

STep	=	<real>	no default
RAtio	=	<real>	(same as step)
CHange	=	<logical>	default : true
Absolute		<logical	default: false
Logarithm	=	<logical>	default: false
Max.level	=	<integer>	default: dynamic
DOPFile	=	<character>	
SMooth.k	=	<integer>	default: 0
LOcaldop	=	<logical>	default: false

REgion	=	<integer>	default:all
IGnore	=	<integer>	default:none
X.MIn	=	<real>	
X.MAx	=	<real>	
Y.MIn	=	<real>	
Y.MAx	=	<real>	
COs.Ang	=	<real>	default:2.0

CHange determines whether to use the magnitude or the difference of a variable in a triangle as the criterion of refinement. It is normally set to true (difference). STEP is the numerical criterion for refining a triangle. RATIO is a synonym. If the variable ranges across many orders of magnitude, it is advisable to examine its logarithmic variation, using the LOG flag. In this case STEP will be interpreted as the step in the logarithm. ABSOLUTE specifies that the absolute value of the quantity is to be used. Since many of the quantities may become negative, PISCES-II actually uses

$$\log(x) = \text{sign}(x) \times \log_{10}(1+|x|)$$

to avoid overflow. To get the true logarithm of a quantity, specify ABSOLUTE and LOGARITHM - the absolute is taken first and there is no danger of negative arguments. LOCALDOP is used with minority carrier regrids, and specifies that if the minority carrier concentration exceeds the local doping, the grid is to be refined. MAX.LEVEL is the maximum level of any triangle relative to the original mesh. It defaults to one more than the maximum level of the grid, but can be set to a smaller value to limit refinement. DOPFILE is the name of a file (up to 20 characters) which contains the doping for the device (see DOPING card). Specifying DOPFILE avoids interpolating doping values at any newly created grid points (the default), by using the initial doping specification to redope the structure. SMOOTH.K has the same meaning as on the mesh card. The bounds X.MIn-Y.Max are used to limit the refinement; only triangles which have nodes which fall inside the box are considered for refinement. The REGION parameter has a similar use; only regions specified are refined according to the user criterion. (Others may be refined as a side effect, to maintain well-shaped triangles). The default is to refine all regions for potential and electric field regrids, and all semiconductor regions for regrids which depend on the other variables. The parameter IGNORE is similar to REGION, but opposite in effect. Ignored regions are not regridded either according to the user criterion or according to the "obtuse criterion" (see below); nor are they smoothed after regrid. The default is not to ignore any region.

The last parameter, COS.ANGLE, defines the "obtuse criterion" to limit the creation of obtuse angles in the mesh. If regrid would create a triangle with an angle whose cos is less than -COS.ANGLE, nodes are added so that this does not occur. The test can be turned off locally by using the ignore parameter; it can be turned off everywhere by using a value of COS.ANG greater than 1. The default is to turn it off everywhere.

<files>

OUTFile = <filename>

OUTFILE is the binary output mesh file, and is necessary if the mesh is to be used for subsequent runs. A history of the triangle tree is stored in a file of a similar name, to assist further regridding steps.

4. Examples

Starting with an initial grid, we refine twice, requesting that all triangles with large doping steps be refined:

```
REGRID  LOG DOPING STEP=6 OUTF=grid1 DOPF=dopxx1
REGRID  LOG DOPING STEP=6 OUTF=grid2 DOPF=dopxx1
```

A similar effect could be obtained with just one regrid statement:

```
REGRID  LOG DOPING STEP=6 OUTF=grid2 DOPF=dopxx1
+       MAX.LEVEL=2
```

In both cases two levels of refinement are done. The first choice is preferable however, because new doping information is introduced at each level of refinement. This gives a better criterion for refinement, and fewer triangles. Now we perform an initial solution and refine triangles which exhibit large potential steps:

```
SOLVE  INIT OUT=grid2.si
REGRID POTENTIAL STEP=0.2 OUTF=grid3
```

The SOLVE card

1. Syntax

SOlve <initial estimate> <bias voltages> <output file>

2. Description

The solve card instructs PISCES to perform a solution for one or more specified bias points.

3. Parameters

<initial estimate>

Initial	=	<logical>
PREvious	=	<logical>
PROject	=	<logical>
LOcal	=	<logical>

The above parameters are used to specify how the initial guess for the solution is to be obtained. The first bias point for a given structure must have the INITIAL parameter specified. From then PISCES will either use the previous solution (PREVIOUS), or if there are two previous solutions present and equivalent bias steps are taken on any electrodes that are changed (see Chapter 3 and example below), an extrapolation (PROJECT) from the preceding two solutions will be used to get an improved initial guess. After the initial bias point, PISCES will automatically use extrapolation wherever possible if no estimate parameter is supplied. A different type of previous guess is available by using LOCAL values of the quasi-fermi levels.

<bias voltages>

V1	=	<real>	
V2	=	<real>	
.			
.			
.			
V9	=	<real>	
V0	=	<real>	
VStep	=	<real>	(default is 0.0)
Nsteps	=	<integer>	(default is 0)
Electrode	=	<integer>	
TStep	=	<real>	(default is ∞)

The parameters V1, V2,...,V9, V0 represent the bias voltages applied at contacts 1, 2,...,9, 0. **The defaults for these parameters are the potentials from the previous bias point.** Note that for the INITIAL bias point, 0 volts will be assumed for any voltage that is not specified. VSTEP is a voltage increment to be added to a voltage at one or more electrodes, as specified by the integer assigned to ELECTRODE. If more than one electrode is to be stepped, ELECTRODE should then be an n-digit integer, where each of the n-digits is a separate electrode number (and if there are 10 electrodes, don't put electrode 0 first in the sequence!). NSTEPS is the number of bias increments (steps) to be taken; i.e., if VSTEP is specified, the specified electrode is incremented NSTEPS times. TSTEP is the time step between bias points (in seconds). The initial bias point at time 0 is assumed to be the previous bias point and NSTEPS time steps are taken. If both a VSTEP and a TSTEP are specified, a ramped voltage can be simulated.

<file i/o> (optional)

Outfile	=	<character string>	
Currents	=	<logical>	(default is true)

The OUTFILE parameter specifies the name of the binary output file for the solution of this bias point. The file names may contain up to 20 characters. If an electrode is stepped so that more than one solution is generated by this card, the last non-blank character of the supplied file name will have its ASCII code incremented by one for each bias point in succession, resulting in a unique file per bias point. If CURRENTS is specified, the electron, hole, and displacement currents, and the electric field, will be computed and stored with the solution.

4. Examples

The following performs an initial bias point, saving the solution to the data file OUT0:

```
SOLVE   INIT OUTF=OUT0
```

In the next example, bias stepping is illustrated. The two solve cards produce the following bias conditions:

Bias point #	V1	V2	V3
1	0.0	0.5	-0.5
2	1.0	0.5	0.0
3	2.0	0.5	0.0
4	3.0	0.5	0.0
5	4.0	0.5	0.0
6	5.0	0.5	0.0

The solutions for these bias points will be saved to the files OUT1, OUTA, OUTB, OUTC, OUTD and OUTE. Note that the initial guess for the first bias point is obtained directly from the preceding solution because the PREVIOUS option was specified. The initial guesses for bias points 2 and 3 will also be obtained as if PREVIOUS had been specified since two electrodes (numbers 1 and 3) had their biases changed on bias point 2. However, for bias points 4, 5 and 6, PISCES will use a projection to obtain an initial guess since starting with bias point 4, both of its preceding solutions (bias points 2 and 3) only had the same electrode bias (number 1) altered.

```
SOLVE   PREV V1=0 V2=.5 V3=-.5 OUTF=OUT1
SOLVE   V1=1 V2=.5 V3=0 VSTEP=1 NSTEPS=4
+       ELECT=1 OUTF=OUTA
```

Here is a case where two electrodes are stepped (2 and 3). The bias points solved for will be (0,0,1), (0,.5,1.5), (0,1,2) and (0,2,3). PISCES will use the PROJECT option to predict an initial guess for the third and fourth bias points since the bias voltages on both electrodes 2 and 3 have been altered by the same amount between each point.

```
SOLVE   V1=0 V2=0 V3=1 VSTEP=.5 NSTEPS=2
+           ELECT=23
SOLVE    V2=2 V3=3
```

If no new voltages are specified and a VSTEP is included, the first bias point solved for is the preceding one incremented appropriately by VSTEP. This is illustrated by repeating the above example as a three card sequence:

```
SOLVE   V1=0 V2=0 V3=1
SOLVE   VSTEP=.5 NSTEPS=2 ELECT=23
SOLVE   VSTEP=1   NSTEPS=1 ELECT=23
```

The following sequence is an example of a time-dependent solution. The first card computes the solution for a device with 1 volt on V1 and 0 on V2 in steady-state. The second SOLVE card specifies that V1 is to be changed instantaneously to 2 volts, and the solution is to be obtained every nanosecond out to 10 ns. The third SOLVE card produces a solution for V1=2, V2=0 using a time-step of 10ns and using the last bias point (V1=2, V2=0 at 10 ns) as an initial guess; i.e., we will obtain the solution at 20 ns, 30 ns, ..., 100 ns after changing V1 from 1 volt to 2 volts. Finally, the fourth SOLVE card performs the steady-state solution at V1=2, V2=0.

```
SOLVE   V1=1 V2=0
SOLVE   V1=2 TSTEP=1e-9 NSTEPS=10
SOLVE   TSTEP=1e-8 NSTEPS=9
SOLVE   V1=2 V2=0
```

The SPREAD card

1. Syntax

SPread <direction> <region> <specifics>

2. Description

The SPREAD card provides a way to distort rectangular grids in the vertical direction to follow surface and junction contours. SPREAD is very useful in reducing the amount of grid for some specific problems, most notably MOSFET's. The SPREAD card is somewhat complicated; it is suggested to follow the supplied examples very carefully (particularly the MOSFET example in Chapter 5).

3. Parameters

<direction>

One of:

LEFT	=	<logical>	(default is false)
RIGHT	=	<logical>	(default is false)

LEFT and RIGHT specify that the left and right-hand sides of the grid respectively be distorted.

<region>

Width	=	<real>
Upper	=	<integer>
LOwer	=	<integer>

WIDTH specifies the width from the left or right edge (depending on the LEFT and RIGHT parameters) of the distorted area. The actual x-coordinate specified by WIDTH (min[x] + WIDTH for LEFT, max[x] - WIDTH for RIGHT) will lie in the middle of the transition region between the distorted and undistorted grid regions. UPPER and LOWER specify the upper and lower y-grid lines between which the distortion will take place.

<specifics>

One of:

Y.Lower	=	<real>
Thickness	=	<real>

And:

Vol.ratio	=	<real>	(default is 0.44)
Encroach	=	<real>	(default is 1.0)
GRAding	=	<real>	(default is 1.0)
GR1	=	<real>	(default is 1.0)
GR2	=	<real>	(default is 1.0)
Middle	=	<real>	
Y.Middle	=	<real>	

The Y.LOWER and THICKNESS parameters define the distorted grid region; only one should be supplied. Y.LOWER is the physical location in the distorted region at which the line specified by LOWER will be moved. The line specified by UPPER is not moved. THICKNESS is the thickness of the distorted region; THICKNESS will usually move the positions of both the UPPER and LOWER grid lines (unless VOL.RATIO is set to 0 or 1). VOL.RATIO specifies the ratio of the downward displacement of the lower grid line to the net increase in thickness. The default is 0.44 so that oxide-silicon interfaces are correct. VOL.RATIO is ignored if Y.LOWER is specified. ENCROACH is a factor which defines the abruptness of the transition between distorted and non-distorted grid. The transition region becomes more abrupt with smaller ENROACH factors (the minimum is 0.1). An important note: depending on the characteristics of the undistorted grid, very bad triangles (long, thin and obtuse) may result if ENCROACH is set too low. GRADING specifies a grid ratio (identical to the RATIO parameter on the X.MESH and Y.MESH cards) to produce a non-uniform grid in the distorted region. As alternative to a single grading parameter, GR1 and GR2 can be specified along with the y grid line MIDDLE and location Y.MIDDLE so that GR1 is used as the grading in the spread region from UPPER to MIDDLE and GR2 is the grading from MIDDLE to LOWER.

4. Examples

The following spreads what was previously a uniform 400 Angstroms of oxide to 1000 Angstroms on the left side of the device. This will result in a net increase in thickness of 600 Angstroms of oxide. Because the default VOL.RATIO is used, 0.44*(600) = 264 Angstroms of the net increase will lie below the original 400 Angstroms and 0.56*(600) = 336 Angstroms of the net increase will lie above the original 400 Angstroms. The width of the spread region is 0.5 μm and the oxide taper is quite gradual because of the high encroachment factor. The grid is left uniform in the spread region.

```
$ *** Mesh definition ***
MESH      NX=30 NY=20 RECT
X.M       N=1   L=0
X.M       N=30  L=2
Y.M       N=1   L=-.04
Y.M       N=5   L=0
Y.M       N=20  L=1  R=1.4

$ *** Thin oxide ***
REGION    X.L=1 X.H=30 Y.L=1 Y.H=5

$ *** Silicon substrate ***
REGION    X.L=1 X.H=30 Y.L=5 Y.H=20

$ *** Spread ***
SPREAD    LEFT WIDTH=0.5 UP=1 LO=5 THICK=0.1 ENC=1.3
```

In the next example, the right side of the grid is distorted in order to follow a junction contour. Assume that the initial grid is defined as above. Y.LOWER is used so that there is no increase in the size of the device, just grid redistribution. With Y.LOWER set to the junction, the ENCROACH parameter should be chosen such that the lower grid line (LOWER=10) follows the junction as closely as possible. Note that the grid is graded so that the grid lines are spaced closer together as they approach the junction. Because the point specified by WIDTH on the spread card lies in the middle of the transition region, it should be chosen to be slightly larger than the width of the doping "box" (WIDTH < X.LEFT - X.RIGHT = 0.5 μm).

```
$ *** Doping ***
DOPING    UNIFORM N.TYPE CONC=1E15
DOPING    GAUSS P.TYPE X.LEFT=1.5 X.RIGHT=2
+         PEAK=0 CONC=1e19 RATIO=.75 JUNC=0.3

$ *** Spread ***
SPREAD    RIGHT WIDTH=0.7 UP=5 LO=10 Y.LO=0.3
+         ENC=1.2 GRAD=0.7
```

The SYMBOLIC card

1. Syntax

SYmbolic <solution method> <carriers> <options>

2. Description

The symbolic card performs a symbolic factorization in preparation for the LU decompositions in the solution phase of PISCES. Because each of the available numerical solution techniques used by PISCES may result in entirely different linear systems, the method used and the number of carriers to be simulated must be specified at this time. The symbolic factorization may be optionally read from or written to a file; if an input file is specified, the symbolic factorization information in that file must be consistent with the method specified on. the present card.

3. Parameters

<solution method>

One of:

Newton	=	<logical>
Gummel	=	<logical>

If the Newton method is chosen, one of the iterative methods can optionally be specified:

Block	=	<logical>
Knot.blocks	=	<logical>

The solution method must always be specified, except for the Poisson only case (carriers = 0). The different methods and their applications are discussed in Chapter 3.

<carriers>

CArriers	=	<integer>	(default is 1)
Electrons	=	<logical>	(default is true)
Holes	=	<logical>	(default is false)

The CARRIERS parameter specifies the number of carriers to be simulated. If only one carrier is to be simulated, the specific

carrier can be specified by including either HOLES or ELECTRONS.

<options>

Min.degree	=	<logical>	(default is true)
Strip	=	<logical>	(default is true)
Infile	=	<character string>	
Outfile	=	<character string>	
Print	=	<logical>	(default is false)

MIN.DEGREE uses a minimum degree ordering of the pivots for decomposition in order to reduce the size of the generated L and U and hence, to reduce the amount of cpu spent in solving linear systems. This parameter is definitely recommended. STRIP specifies that redundancy (zero couplings) be removed from the symbolic map, and is naturally on. The INFILE and OUTFILE parameters specify input/output files names for the symbolic factorization. The file names may contain up to 20 characters. Note that these binary files can be quite large, so it may be advisable not to use this feature. (In some computing environments it is also faster to compute the symbolic information than to read it from disk). The PRINT parameter indicates that information about the memory allocated for the run should be printed to the PISCES standard output file.

4. Examples

The following specifies a symbolic factorization for a simulation with only holes and using the full Newton method (the symbolic factorization is saved in a file called SYMB.OUT):

```
SYMBOLIC  NEWTON CARR=1 HOLES OUTF=SYMB.OUT
```

In the next example, a previously generated symbolic factorization is read in from a file called SYMB.IN. The method used is the knot-block Newton method and both carriers are included in the simulation:

```
SYMBOLIC  NEWTON CARR=2 INF=SYMB.IN KNOT
```

The TITLE card

1. Syntax

Title <character string>

2. Description

The TITLE card specifies a title (up to 60 characters) to be used in PISCES ASCII output.

3. Examples

```
TITLE  *** CMOS p-channel device ***
```

The VECTOR card

1. Syntax

VEctor <plotted quantity> <control>

2. Description

The VECTOR card plots vector quantities over an area of the device defined by the previous PLOT.2D card.

3. Parameters

<plotted quantity>

One of:

J.Conduc	=	<logical>	Conduction current
J.Electr	=	<logical>	Electron current
J.Hole	=	<logical>	Hole current
J.Displa	=	<logical>	Displacement current
J.Total	=	<logical>	Total current
E.Field	=	<logical>	Electric field

The above parameters specify the quantity to be plotted. There is no default.

<control>

LInear	=	<logical>	(default is false)
Scale	=	<real>	(default is 1)
LIne.type	=	<integer>	(default is 1)

LINEAR specifies linearly scaled magnitudes. SCALE a scale factor to be multiply all magnitudes by. LINE.TYPE specifies the vector line type for plotting.

4. Examples

Plot electron and hole currents over a device :

```
PLOT.2D  BOUN  NO.FILL
VECTOR  J.ELEC  LINE=2
VECTOR  J.HOLE  LINE=3
```

The X.MESH and Y.MESH cards

1. Syntax

X.Mesh <node> <location> <ratio>
Y.Mesh <node> <location> <ratio>

2. Description

The X.MESH and Y.MESH cards specify the location of lines of nodes in a rectangular mesh.

3. Parameters

<node>

Node = <integer>

This is the number of the line in the mesh. There can be at most 120 lines in either direction. Lines are assigned consecutively, beginning with the first and ending with the last.

<location>

Location = <real>

This is where to locate the line. The location is interpreted in microns.

<ratio>

Ratio = <real>

This gives the ratio to use when interpolating lines between the given lines. The spacing grows/shrinks by ratio in each subinterval, and the ratio should usually lie between 0.667 and 1.5.

4. Examples

Space lines closely around a junction in a 1-d diode with the junction at 0.85 microns :

```
Y.MESH  N=1  LOC=0.0
Y.MESH  N=20  LOC=0.85  RATIO=0.75
Y.MESH  N=40  LOC=2  RATIO=1.333
```

Appendix B

IGGI commands

1. Main command level.

IGGI is an interactive program which takes its input from a user at a graphics terminal. Each command consists of one letter, terminated by a carriage return. If several commands begin with the same letter, the program then prompts for the second letter of the command, which should be typed and followed by a carriage return. If a line has several letters before the carriage return, only the last is significant.

The top level menu has the following commands.

c)reate new mesh	d)raw mesh	e)dit mesh	g)enerate nodes
r)ead mesh	s)core mesh	s)mooth nodes	s)mooth triangles
t)riangulate	w)rite mesh	o)rder nodes	
q)uit			

C)reate new mesh

This choice enables the user to initiate a new device outline. It sets up a background grid to assist in aligning input. It then waits for the user to draw the outline of the structure, which should be entered counter-clockwise using the graphics cursor. (The counter-clockwise requirement is essential - all regions in the device are maintained in counter-clockwise order and entering the outline backwards causes some very peculiar problems which do not always appear until the grid is triangulated.) The device outline must be a polygon, with no sides and no nodes repeated. It can be subdivided into more complex shapes later. After the last point is entered, the cursor should be moved outside the window to indicate that the device is complete; IGGI then closes the connection beteen the first and last node and leaves the user at edit command level.

D)raw mesh

The draw command plots the device outline on the screen without entering the edit level.

E)dit mesh

The grid editor has a large variety of options and is treated separately in the next section.

G)enerate nodes

Once the region boundaries have been specified and suitable grid densities chosen, IGGI will generate nodes in a straightforward way in each region. Precautions are taken not to crowd or duplicate any preexisting nodes. If electrodes have been specified, any node added on the boundary between two existing electrode nodes is made part of the same electrode. The progress of node generation is displayed by updating the graphics display with the new nodes.

R)ead mesh

Two mesh formats are read, internal format binary files and user-prepared text files. Internal format files are used to save a particular mesh so it can be read back and modified. User-prepared files can be used to enter x-y data from some external source, for instance a digitized SEM photograph. If there is no graphics input available, files also provide the only means of using the non-planar geometrical facilities of IGGI. The format of user-prepared files is detailed at the end of this appendix.

S)core mesh

The score command prints mesh statistics : the number of nodes and elements, the percentage of obtuse triangles, the ratio of largest to smallest element in the mesh, and the largest and smallest angles. In general the aim of the mesh generator is to make the triangles as equilateral as possible, and in particular to avoid obtuse triangles.

S)mooth nodes

The smooth nodes command moves each node to the average of the nodes around it, and repeats this pass until all nodes are stationary.

S)mooth triangles

The smooth triangles command searches the mesh for triangle pairs to flip, as discussed in Section 4.6.

T)riangulate

The triangulation algorithm is the cornerstone of the irregular grid program. It takes nodes located in arbitrary positions and links them together to form triangles, while trying to optimize the shape of the triangles formed. As in node generation, the progress of the algorithm is shown by updating the graphics display with the new triangles as they are formed. After triangulation, a triangle smoothing step is always recommended.

W)rite mesh

Meshes are written in two formats, a binary internal format, and a text file which can be read by PISCES-II. The binary format is used to store meshes for later re-use within IGGI, while the text file provides the link to device simulation. When an output file is written for PISCES-II, the y coordinate is negated to satisfy the semiconductor convention of increasing y into the bulk.

Q)uit

The quit command exits the grid generator, after checking whether or not to save the current grid.

2. Editor level

The edit level menu has the following commands.

a)dd nodes	c)hange region	de)lete nodes	di)vide regions
j)oin regions	o)rigin change	m)ove nodes	p)rior mesh
r)edraw	sa)ve mesh	se)t options	sc)ale
w)indow	e)lectrode	q)uit	

In the following, several conventions are used.

the 'escape box' is a small box drawn at the lower right of the screen. Graphics input is terminated by moving the cursor to the escape box, or to any location outside of the main plotting area.

the 'reference grid' is a light square grid drawn in the background. When it is on, user input is latched to the nearest reference grid point, otherwise it is taken as given. **N.B.** Coarse reference grids should be treated with care, as the nearest

reference point may be quite far from the graphics cursor. This can cause surprising results.

A)dd nodes

The add node command allows direct specification of mesh nodes. Boundary nodes and internal nodes are treated differently. For adding internal nodes, the graphics cursor is displayed and input is entered by moving the cursor to various locations and pressing the space bar. Pressing any other key causes IGGI to prompt for the coordinates in x,y form. All graphics input is handled in this way, so that accurate values may be entered when necessary. Moving the cursor to the escape box terminates input. For adding boundary nodes, the program prompts first for the region and secondly for the boundary segment to be split in two. The user then draws the new boundary node. Any existing triangulation is destroyed by adding nodes.

C)hange region parameters

This command prompts for the grid spacing of a region and for its material type. Density may be specified as a number in microns, or as a fraction of some other region's density, or the graphics device may be used to draw a small box representing the average spacing between nodes. The number of nodes generated is a sensitive function of the density. The density should not vary too abruptly from one region to another, or the mesh will be very strained at the interface. It is recommended *not* to use a fine grid spacing in IGGI, allowing automatic refinement in PISCES-II to handle rapid changes in the solution. This procedure avoids the need to laboriously prepare graded meshes tailored to a single device.

De)lete nodes

Nodes may be removed by pointing to them with the graphics cursor. Any existing triangulation is destroyed by removing nodes.

Di)vide region

The division mechanism allows the user to divide the device into different material sections. The dividing line must begin at an existing boundary node, cross the region and finish at another boundary node. (Divides are frequently preceded by add-boundary sequences to make the necessary start and finish nodes.) The dividing line must lie entirely inside the region being divided, must contain no duplicate nodes and must not cross itself. The user is prompted for the region to be divided, which is indicated by positioning the cursor anywhere in that region. The program then prompts for the start of the split and

the user should select a boundary node of that region. Then the dividing line should be drawn; new nodes may be added en route to make a curved division. Finally the cursor is positioned **outside** the region being divided, close to the final node of the split, and the program closes the region definitions. **Do note** that the division is not complete until the cursor is positioned outside the region beside the final node; a very common error is to add a new node just inside the boundary, expecting it to finish the division. Both subregions are given the same density and material type as the parent, which may then be changed with the c)hange region command.

E)lectrode draw/create

The electrode draw command emphasizes the nodes of a selected electrode. The electrode create command allows the user to select given nodes to be part of an electrode. Electrodes are allocated consecutively, starting at number 1. Creating an electrode which already exists has the effect of removing the old one.

J)oin regions

This command undoes the effect of the divide command. IGGI prompts for the two regions. Provided the two selected regions share a common boundary, they are amalgamated into one region. If they are both triangulated, the new region retains their triangles, otherwise the new region is untriangulated. The user is prompted for the material type of the new region.

O)rigin change

Any node of the mesh can be selected as an origin, and the remaining nodes have their coordinates re-calculated relative to this node.

M)ove nodes

Mesh nodes may be moved at will. IGGI turns over the graphics cursor to the user, who may repeatedly select a node and then indicate its new position. The procedure is terminated by moving the cursor to the edge of the picture. Existing triangles are redrawn to reflect the new position of the node. This is useful for optimizing triangle shapes manually.

P)rior mesh

When entering the editor, a temporary copy of the working grid is stored to disk; this copy can be updated at any time by the save command. Prior mesh overwrites the working mesh with the stored copy, and is used to undo mistakes.

R)edraw

The redraw command resets the window stack to zero and redraws the mesh at full scale.

Se)t options

The set options command provides variations in how the mesh is displayed. Each option may be toggled. The fill option scales the mesh to use the entire viewscreen area, otherwise the mesh is drawn with equal x and y scales. (Without equal scales it is difficult to judge the quality of the triangulation). The triangles option causes triangles to be displayed. The obtuse option causes obtuse triangles to be shaded. If the region boundaries are turned on, boundary nodes are highlighted with a small diamond, otherwise they are drawn as single pixels (to speed up graphics output).

Sc)ale

A strip of the mesh may be stretched in the x or y direction. Iggi prompts for the direction and lower/upper bounds of the strip, and for the new length of the strip. Nodes within the strip are stretched/compressed to fit its new length, nodes with coordinates above the upper bound of the strip are moved accordingly, and nodes with coordinates below the lower bound are left unchanged.

W)indow

The graphics window may be used to zoom down on different parts of the mesh. To window in, the program prompts for the lower left and upper right corners of a window, then redraws everything inside that window to fill the screen. The new window gets pushed onto a stack. Window-out pops the stack and redisplays the mesh at the previous scale.

Q)uit

exits the editor and returns to main command level.

3. Input file format.

Iggi reads a boundary description of a mesh in a simple format. The various parts of a mesh appear in the order :

- List of node coordinates
- List of nodes on the outer perimeter
- List of nodes in the boundary of each region
- List of electrode nodes
- Terminating card.

The cards all have the same simple format, consisting of a one character identifier in column 1, and one or two free-format numeric fields.

Each **node** card is identified with a 'c', followed by the x and y coordinates of the node. Coordinates are taken as microns. The order of specification determines the index of the node in the mesh.

The **perimeter** section begins with a card carrying just a 'p', followed by cards specifying the nodes on the perimeter. Each of these begins with a 'b' , followed by the index of a node. The order of the cards determines the ordering of nodes in the boundary, and must be counter-clockwise.

After the perimeter is specified, the **regions** must be specified. Each region begins with a card carrying an 'r' and the material code of that region, followed by the desired node spacing in that region. The material codes are :

PISCES-II material codes	
Silicon	1
Gallium Arsenide	2
Other semiconductor	3
Silicon dioxide	-1
Silicon nitride	-2
Sapphire	-3
Other insulator	-4

This initial card is followed by cards specifying the nodes on the boundary of the region. Each is denoted by a 'b', followed by the index of a node. The order of the cards determines the ordering of nodes in the boundary, and must be counterclockwise.

The **electrodes** are specified by an initial card carrying an 'e', followed by a series of cards beginning with a 'b' and specifying the index of an electrode node.

The input must be terminated with a card starting 'd'.

3.1. Example

To clarify this, we illustrate the description of a square 2x2 mesh. There is only one region, and one electrode which covers the bottom surface.

```
c 0.0 0.0
c 1.0 0.0
```

```
c 0.0 1.0
c 1.0 1.0
p
b 1
b 2
b 4
b 3
r 1 0.2
b 1
b 2
b 4
b 3
e
b 1
b 2
d
```

Note the counterclockwise specification of the perimeter.

Appendix C

Portability considerations

The PISCES-II release tape contains three distinct programs :

- the simulation program itself (PISCES II-A)
- the grid editor (IGGI)
- the graphics postprocessor

The sections below deal with installing the code in a different system, changing the graphics to deal with different graphics devices, and changing the size of the program.

1. Installation.

The release tape is in unlabelled, fixed-block ASCII format. The first file on the tape is named FILES, and contains a list of all the other files on the tape. The subroutines of all programs are included in alphabetical order. The first step in installation should be to separate the different programs into different directories, provided the target system supports a hierarchically organized file system. (This is for the sake of convenience in reading a directory listing). It is suggested that the program be split along language boundaries. PISCES-II and the postprocessor are written in Fortran 77, and the file MakefileP lists the subroutines needed by both. The low-level graphics support for both programs is written in Ratfor (Rationalized Fortran) and Fortran translations are included. The file MakefileG describes the routines it uses. IGGI is in Berkeley Pascal, and the file MakefileI lists its subroutines. The particular language dependencies present in each program are now detailed.

1.1. Fortran 77

The simulator and the postprocessor are written directly in Fortran 77 and adhere to the ANSI standard [1] with one exception - the include facility. It is used throughout to interpolate common blocks into the code, and avoids repeating the definition of the blocks in each subroutine. If the local compiler does not support this feature, then each include statement should be replaced with the file it references. It is essential that the order of the blocks be maintained. A small number of system calls are made. Subroutine clock makes the

made here.

4. References

[1] ANSI publication X3.9-1978

[2] William N. Joy, Susan L. Graham, Charles B. Haley, *Berkeley Pascal User's Manual - Version 1.1*, April 1979.

[3] Kathleen Jensen and Niklaus Wirth, *Pascal - User Manual and Report*, Springer-Verlag, New York(1975)