PISCES-2ET and Its Application Subsystems

PART I: PISCES-2ET 2D Device Simulator

Z. Yu, D. Chen, L. So, and R. W. Dutton

PART II: Curve Tracer

S. Beebe, Z. Yu, R. J. G. Goossens, and R. W. Dutton

PART III: Mixed-Mode Device/Circuit Simulator

F. Rotella, Z. Yu, and R. W. Dutton

Integrated Circuits Laboratory
Stanford University
Stanford, California 94305

Table of Contents

PART I

PISCES-2ET 2D Device Simulator

for Silicon and Heterostructures

Zhiping Yu, Datong Chen, Lydia So
and Robert W. Dutton

CHAPTER 1 Introduction and Acknowledgment

PISCES-2ET is a dual energy transport (for carrier temperatures and lattice thermal diffusion) version of PISCES-II developed based on Stanford's 9009 version and Intel's enhanced 8830 version. This work is completed under the support of SRC (Semiconductor Research Corp.), ARO (Army Research Office), and ARPA (Advanced Research Projects Agency) and with close collaboration from Intel and HP. There are many new features available in the 2ET version, predominately, the capabilities to simulate the carrier and lattice temperatures and heterostructures in compound semiconductors. Hence, various non-stationary phenomena such as hot carrier effects and velocity overshoot can be analyzed using this program. The electrical behavior of optoelectronic devices can also be simulated with reasonable accuracy. Most of the material parameters have been calibrated and thoroughly surveyed with the help from industry. Other new features which can be found in PISCES-2ET include:

- High and zero frequency small signal AC analysis
- Improved initial guess using Newton projection method
- Carrier energy dependent mobility and impact ionization models
- About ten different local and non-local (electric) field dependent mobility models
- Parameterization of array sizes for expansion of maximum grid number
- ASCII and binary exchange data format with process simulation programs

This document is organized as follows. CHAPTER 2 discusses the theoretical background for DUET transport model, which is based on the moment approach to solving Boltzmann Transport Equation (BTE) and is applicable to heterostructure analysis also. In CHAPTER 3, physical models used in the program are described. The emphasis is on the various mobility models, many parameters of which have been calibrated by industrial users. Material parameters for four common compound semiconductors, namely Ge_xSi_{1-x}, $Al_xGa_{1-x}As$, $Al_xIn_{1-x}As$, and $Ga_xIn_{1-x}As_yP_{1-y}$, used in heterostructure and optoelectronic devices are discussed in CHAPTER 4. Discussion of carrier transport in heterostructures and issues related to Fermi-Dirac (FD) statistics are delayed until APPENDIX A to provide a complete picture of DUET model yet not to inundate the concise forms in the main text with complex coefficients required by using the FD distribution. In CHAPTER 5, relevant numerical techniques are presented, including the algorithms for small signal AC analyses. Besides, the details in parameterization of global arrays used in the device solver are provided. Finally four simulation examples are included in CHAPTER 6. The first two examples compare the simulated results to the experimental data regarding the carrier temperature effects in FET structures to establish the confidence level of the energy transport model used in the code. The other two examples are about the simulation of heterostructures including one application in a realistic, cylindrically symmetric LED (light emitting diode) structure. Two short appendices, one dealing with the mathematical properties of Fermi integral (APPENDIX B) and the other listing several expressions referred to in this document from the previous PISCES reports (APPENDIX C), are included for users' reference. Finally a user's manual provides the detailed syntax and usage of all commands available in the program.

Authors are grateful to many contributions from both academia and industry during the development of PISECE-2ET. The DUET transport model has been developed in collaboration with Prof. K. Hess' group in University of Illinois at Urbana. One of the key developers, Dr. E. Kan, who recently joined our group at Stanford, made numerous contributions to the code including helping parameterize the data structure and model calibration. Drs. T. Thurgate of Intel and M. Tan of HP have been working closely with us from the very beginning of this project and provided access to their codes and experimental data. Various industrial connections under the sponsorship of SRC are greatly appreciated, including, but by no means being complete, Drs. R. Lowther of Harris Semiconductor, I. Lim of Motorola, and K. L. Chen of OCD in HP. Dr. K. Wu of this group at Stanford developed the algorithm for high frequency small signal AC analysis, and Mr. A. Mujtaba helped review the section of mobility models. Their contributions are greatly appreciated. Finally, the continuous funding from

SRC through contract #SRC 93-SJ-116 is one of the key factors to the success of this program development and the grant from ARO through contract DAAL 03-91-G-0152 plays a critical role in extending the simulation capabilities from silicon to compound materials and heterostructure. The GaAs work was supported through ARPA contract DAAL 01-91-K-0145. Their support are gratefully acknowledged.

CHAPTER 2
DUET Carrier Transport Model

The DUET model, a carrier transport model in semiconductors, is developed based on the moment approach to solving Boltzmann Transport Equation (BTE). It uses six state variables to describe the status of a semiconductor device. These six variables are: electrostatic potential, ψ, carrier concentrations, n and p, carrier temperatures, T_n and T_p, and lattice temperature, T_L, and they are functions of space and time. All other device characteristics such as terminal *I-V* characteristics and circuit model parameters can be calculated from the knowledge of the distribution of these basic variables. To determine the distribution of these variables under applied bias, six independent equations are required together with proper boundary conditions. It is well established that with the drift-diffusion (DD) carrier transport model, Shockley semiconductor equations, i.e., Poisson's equation and carrier continuity equations, govern the distribution of ψ, n, and p. The carrier concentrations can also be replaced, equivalently, by their respective quasi-Fermi levels, ϕ_n and ϕ_p, in classical distribution (either Boltzmann or Fermi-Dirac) functions. With the temperatures for both carriers and lattice introduced as independent variables, three more equations are needed and they can be derived from the energy balance principle. In this chapter, we will first introduce two (kinetic) energy balance equations for carriers and one thermal diffusion equation for the lattice. Various macro quantities such as carrier and energy densities will be defined using the distribution function at the quasi-thermal equilibrium. The auxiliary expressions for transport are provided. Finally the proper boundary conditions for solving differential equations are discussed.

2.1 Boltzmann Transport Equation and Distribution Function

The key assumption in the DUET model is that the true distribution function can be constructed based on the quasi-thermal equilibrium function using the perturbation theory as follows [1]:

$$f(\boldsymbol{r}, \boldsymbol{k}) = f_0(\boldsymbol{r}, \boldsymbol{k}) - \tau(\boldsymbol{r}, \boldsymbol{k}) \left(\frac{h}{2\pi m^*} \boldsymbol{k} \cdot \nabla f_0 - \frac{h}{2\pi m^*} \frac{\partial f_0}{\partial \varepsilon} \boldsymbol{k} \cdot \boldsymbol{E} \right) \tag{2.1}$$

where $\boldsymbol{r}$ and $\boldsymbol{k}$ are spatial (position) and wavenumber vectors, respectively, f_0 is the distribution function at the quasi-thermal equilibrium in both the real $(\boldsymbol{r})$ and momentum $(\boldsymbol{k})$ space, τ is the relaxation time which depends on both $\boldsymbol{r}$ and $\boldsymbol{k}$, and so is defined in a microscopic sense, $\boldsymbol{E}$ is the electric field and ε is the carrier kinetic energy, e.g. for electrons, $\varepsilon = E - E_C$ where E_C is the conduction band edge. Note that in this manual, we use bold faced E $(\boldsymbol{E})$ and it components to represent the electric field, while the plain faced E is used for the total energy of carriers. All other symbols in the above expression have conventional meanings.

The quasi-thermal equilibrium function, f_0, is chosen from one of the classical distribution functions and for the present we will use the Boltzmann distribution function. In APPENDIX A, we will describe how to extend the distribution function to Fermi-Dirac statistics. The Boltzmann distribution function has the following form in $\boldsymbol{r}$ and $\boldsymbol{k}$ space:

$$f_0(\boldsymbol{r}, \boldsymbol{k}) = 2 \exp\left(-\frac{E - E_{Fn}}{k_B T_n} \right) = 2 \exp\left(-\frac{\varepsilon - (E_{Fn} - E_C)}{k_B T_n} \right) \tag{2.2}$$

where E_{Fn} and T_n are the quasi-Fermi energy and temperature for electrons, respectively. In our later analysis, it is often desirable to express f_0 in terms of the carrier concentration. We now proceed this transformation. From definition, it is easy to conclude that the carrier concentration is the zero-moment of the distribution function in $\boldsymbol{k}$ space. Thus taking electrons as an example, we have

$$n = N_C \exp\left(\frac{E_{Fn} - E_C}{k_B T_n} \right) \tag{2.3}$$

where N_C is the effective density of states for the conduction band and is found to be

$$N_C = 2\left(\frac{2\pi m_n^*(T_L)k_BT_n}{h^2}\right)^{3/2} \tag{2.4}$$

where T_L is the lattice temperature and because in our DUET model, T_L is not necessarily the same as T_n, we have explicitly indicated the lattice temperature dependence of the effective mass, m_n^*.

Expressing $E_{Fn} - E_C$ in terms of *n* in Eq. (2.3), we obtain an expression of f_0 as function of *n*, T_n, T_L, and ε as follows:

$$f_0(\boldsymbol{r}, \boldsymbol{k}) = f_0(n(\boldsymbol{r}), T_n(\boldsymbol{r}), T_L(\boldsymbol{r}), \varepsilon) = \frac{2n}{N_C(T_n, T_L)}\exp\left(-\frac{\varepsilon}{k_BT_n}\right) \tag{2.5}$$

This form of distribution function constitutes the basis for our model discussion.

2.2 Energy Balance and Thermal Diffusion Equations

Temperature is a measure of the kinetic energy for random motion and the equation governing the carrier/lattice temperature can be derived from the energy balance principle. Because of the free particle nature of carriers, Fick's second law is applied to describe the carrier kinetic energy balance (or continuity). For electrons, the continuity principle dictates

$$\frac{\partial w_n}{\partial t} = -\nabla\cdot\boldsymbol{s}_n + \boldsymbol{j}_n\cdot\boldsymbol{E}_n - u_{wn} \tag{2.6}$$

where *w* is the *kinetic* energy density and ***s*** is the energy flux, and both can be defined and evaluated from the carrier distribution function. The last two terms on the right hand side (RHS) of the above equation represent the energy conversion and net loss rates, respectively. The familiar Joule heat term, $\boldsymbol{j}_n\cdot\boldsymbol{E}_n$ where $\boldsymbol{E}_n$ is the electric field acting on electrons[1], actually represents the rate of conversion from the electrostatic potential to the kinetic energy, and u_w is the rate of net loss (loss minus

1. In heterostructures, the electric field acting on carriers is in general different for electrons and holes.

generation) for kinetic energy due to carrier recombination/generation and energy exchange with the lattice through phonon scattering. The first step in the development of carrier transport model is then to define and determine the expressions for various quantities used in the above balance equation. DUET model is based on the Stratton's description of distribution function at non-thermal equilibrium (Eq. (2.1) and refer to [1] for details). The concept of the carrier temperature can directly be deduced from the definition of the kinetic energy density in this approach, and it turns out that the parameter T_c, where *c* stands for either *n* or *p*, used in the distribution function, f_0,at quasi-thermal equilibrium (Eqs. (2.2) and (2.5)) is the carrier temperature following the ideal gas kinetics. The carrier concentration (*n*), kinetic energy density (w_n), current density ($\boldsymbol{j}_n$), and energy flux ($\boldsymbol{s}_n$) can all be evaluated from the distribution function, *f* in Eq. (2.1), from their respective definitions by integration in the momentum (i.e. $\boldsymbol{k}$) space:

$$n = \int f d^3 p = N_C \exp\left(\frac{E_{Fn} - E_C}{k_B T_n}\right) \tag{2.7}$$

$$w_n = \int \frac{p^2}{2m_n^*} f dp^3 = \frac{3}{2} n k_B T_n \tag{2.8}$$

$$\begin{aligned} \boldsymbol{j}_n &= -\frac{q}{m_n^*} \int \boldsymbol{p} f d^3 p \\ &= qD_n \nabla n - \frac{3}{2} qnD_n \nabla \ln m_n^* + n\mu_n \nabla E_C + qnD_n^T \nabla T_n \\ &= n\mu_n \nabla E_{Fn} + qn\mu_n Q_n \nabla T_n \end{aligned} \tag{2.9}$$

$$\boldsymbol{s}_n = \frac{1}{m_n^*} \int \boldsymbol{p} \frac{p^2}{2m_n^*} f d^3 p = -P_n T_n \boldsymbol{j}_n - \kappa_n \nabla T_n \tag{2.10}$$

where we have introduced the carrier momentum, $\boldsymbol{p} = (h/2\pi)\boldsymbol{k}$, and the integral element volume, $d^3 p = (dk_x dk_y dk_z)/(2\pi)^3$ and constant effective mass in (crystal) momentum space is assumed[1]. The transport coefficients *D* (diffusion constant), μ (carrier mobility), D^T (thermal diffusivity or called "Soret" coefficient in [1]), *Q* (thermopower), *P* (thermoelectric power), and κ (thermal conductivity)

1. This assumption is not required in the DUET model (see [2]) but is used for present implementation of the code.

are all defined and can be evaluated from the even part of the distribution function *f* in the momentum space (f_0 in Eq. (2.1) for DUET model), and are thus related to each other. For continuity of presentation, the expressions for these coefficients are not given here, rather they can be found in APPENDIX A. We now consider the rates for energy generation g_w and loss r_w, from which

$$u_w = r_w - g_w \tag{2.11}$$

r_w is related to both carrier recombination and energy transferring from carriers to the lattice. At present, three recombination mechanisms are considered in the model and they are Shockley-Reed-Hall (SRH), Auger, and radiative recombinations. For g_w, only the impact ionization, which is the inverse process of Auger recombination, is taken into account[1]. With all relevant mechanisms identified, we are able to list the complete set of equations for the DUET model and their auxiliary expressions as follows:

Poisson's equation:

$$\nabla \cdot (-\varepsilon \nabla \psi) = q\,(p - n + N_D^+ - N_A^-) \tag{2.12}$$

where dielectric constant, ε, is not to be confused with carrier kinetic energy, $\mathcal{E}$.

Carrier continuity equations:

$$\frac{\partial n}{\partial t} = \frac{1}{q}\nabla \cdot \boldsymbol{j}_n - u \tag{2.13}$$

$$\frac{\partial p}{\partial t} = -\frac{1}{q}\nabla \cdot \boldsymbol{j}_p - u \tag{2.14}$$

where *u* is the net recombination rate of electron-hole pairs.

Carrier (kinetic) energy balance equations:

$$\frac{\partial w_n}{\partial t} = -\nabla \cdot \boldsymbol{s}_n + \boldsymbol{j}_n \cdot \boldsymbol{E}_n - u_{wn} \tag{2.15}$$

1. Thus, photo-generation as discussed in Section 3.3.2 applies only to the conservation of carrier population.

$$\frac{\partial w_p}{\partial t} = -\nabla \cdot \boldsymbol{s}_p + \boldsymbol{j}_p \cdot \boldsymbol{E}_p - u_{wp} \tag{2.16}$$

Thermal diffusion equation for the lattice:

$$\begin{aligned} c_L \frac{\partial T_L}{\partial t} &= \nabla \cdot (\kappa_L \nabla T_L) + u_{SRH}\left[\frac{3}{2}k_B T_n + E_g(T_L) + \frac{3}{2}k_B T_p\right] + \\ &\frac{w_n(T_n) - w_n(T_L)}{\tau_{wn}} + \frac{w_p(T_p) - w_p(T_L)}{\tau_{wp}} \end{aligned} \tag{2.17}$$

where τ_{wn} and τ_{wp} are energy relaxation times for electrons and holes, respectively.

The following auxiliary expressions are needed for current and energy flux densities, and energy loss rate in Eqs. (2.15)-(2.17).

Current densities:

$$\begin{aligned} \boldsymbol{j}_n &= qD_n \nabla n - \frac{3}{2} qnD_n \nabla \ln m_n^* + n\mu_n \nabla E_C + qnD_n^T \nabla T_n \\ &= n\mu_n \nabla E_{Fn} + qn\mu_n Q_n \nabla T_n \end{aligned} \tag{2.18}$$

$$\begin{aligned} \boldsymbol{j}_p &= -qD_p \nabla p + \frac{3}{2} qpD_p \nabla \ln m_p^* + p\mu_p \nabla E_V - qpD_p^T \nabla T_p \\ &= p\mu_p \nabla E_{Fp} - qp\mu_p Q_p \nabla T_p \end{aligned} \tag{2.19}$$

Energy flux densities:

$$\boldsymbol{s}_n = -P_n T_n \boldsymbol{j}_n - \kappa_n \nabla T_n \tag{2.20}$$

$$\boldsymbol{s}_p = P_p T_p \boldsymbol{j}_p - \kappa_p \nabla T_p \tag{2.21}$$

where D and μ are related by Einstein relationship (see Eq. (A.23) for a general expression) and Q is also linked to μ in a more complex way (Eq. (A.43)), other coefficients have more simple expressions if only Boltzmann statistics is considered.

$$P_n = P_p = P = \frac{3}{2}\frac{k_B}{q} \tag{2.22}$$

$$\kappa_n = nk_B T_n \mu_n P \tag{2.23}$$

$$\kappa_p = pk_B T_p \mu_p P \tag{2.24}$$

It is apparent that the carrier mobility, μ, is the key parameter to other transport coefficients, and the mobility can be obtained either from the theoretical calculation or more often through empirical or semi-empirical formulation by fitting the analysis/simulation results to the experimental data. We will discuss in a great detail the mobility modeling in CHAPTER 3. Finally, the rate of net loss of carrier kinetic energy is modeled as follows:

$$\begin{aligned} u_{wn} = \frac{3}{2}(u_{SRH} + u_{rad})\, k_B T_n - (u_{n,Auger} - g_{n,imp}) \left[E_g(T_L) + \frac{3}{2} k_B T_p\right] - \\ \frac{3}{2} g_{p,imp} k_B T_n - \frac{w_n(T_n) - w_n(T_L)}{\tau_{wn}} \end{aligned} \tag{2.25}$$

$$\begin{aligned} u_{wp} = \frac{3}{2}(u_{SRH} + u_{rad})\, k_B T_p - (u_{p,Auger} - g_{p,imp}) \left[E_g(T_L) + \frac{3}{2} k_B T_n\right] - \\ \frac{3}{2} g_{n,imp} k_B T_p - \frac{w_p(T_p) - w_p(T_L)}{\tau_{wp}} \end{aligned} \tag{2.26}$$

where u_{SRH}, u_{rad}, and u_{Auger} are net carrier loss rates due to SRH, Auger, and radiative recombinations, respectively, g_{imp} is the generation rate due to the impact ionization (II). Note that for both Auger recombination and II, we need to distinguish if they are caused by electrons or holes. The last term on RHS of each above expression represents the energy exchange between the carriers and lattice.

A special case for the above general description (Eqs. (2.15)-(2.17)) is when only the lattice temperature is of concern, i.e., the carriers can be considered to have the same temperature as the lattice. Such a scenario is often encountered in, say, the simulation of power devices. We can then lump Eqs. (2.15)-(2.17) by requiring all temperatures being the same (designated *T*) and need to solve the following thermal diffusion equation only.

$$\frac{\partial}{\partial t}\left[\left(c_L + \frac{3}{2}(n+p)\,k_B\right) T\right] = \nabla \cdot (\kappa_L \nabla T - \boldsymbol{s}_n - \boldsymbol{s}_p) + \boldsymbol{j}_n \cdot \boldsymbol{E}_n + \boldsymbol{j}_p \cdot \boldsymbol{E}_p + u_{SRH} E_g(T) \tag{2.27}$$

Together with Poisson's (Eq. (2.12)) and carrier continuity equations (Eqs. (2.13)-(2.14)), these four equations are solved for four state variables: ψ*, n, p,* and *T* (for both lattice and carriers).

2.3 Boundary Conditions

Since the DUET model uses both carrier and lattice temperatures to specify the device state, we also need to establish thermal boundary conditions for temperatures in addition to the electrical boundary conditions (See [3] and [4] for detailed discussion). We discuss here only the thermal boundary conditions.

Normally the Dirichlet thermal boundary condition, i.e., the known carrier and lattice temperatures, is applied to both thermal and electrical contacts. In many cases, however, the heat conduction through the intimate contacts to the device is modeled by a thermal resistance. In this case, the lattice (and carrier) temperature at the contact becomes unknown. The relationship between the heat flux and temperatures at the contact ($T_{L,\,cont}$) and thermo-reservoir (the environment temperature, T_{env}) is specified as follows:

$$F_{th} = \frac{T_{env} - T_{L,\,cont}}{R_{th}} \tag{2.28}$$

where the direction of the heat flux, F_{th}, is defined as going into the device. In the present model, the carrier temperature is always assumed the same as the lattice temperature at the contact. There may be two types of contacts to the device: electrical and thermal ones. An electrical contact must be a thermal contact also, but the contrary is not necessarily true. For those boundaries of a device region where there is no electrical/thermal contact specified, a reflective boundary condition is assumed as is the case for electrical boundary conditions.

CHAPTER 3
Physical Models

3.1 Mobility Models

Carrier mobility is one of the most important parameters in the carrier transport model. In PISCES-2ET, the mobility is, for most cases, modeled as the function of the total doping density, *N*, lattice temperature, T_L, surface/interface scattering mechanisms which are generally modeled using the dependence on the transverse electric field to the surface/interface, and the electric field along the current path (longitudinal field). When the device size is in the submicron regime, however, the local, longitudinal field dependence may not be accurate enough and non-local effects, which is mainly characterized by the carrier temperature dependence, have to be taken into consideration. The following expression describes in a general way the carrier mobility dependence on various factors:

$$\mu(N, T_L, E_\perp, E_\parallel / T_c) = f(\mu_0(N, T_L, E_\perp), E_\parallel / T_c) \tag{3.1}$$

where $E_\parallel$ and $E_\perp$ are the longitudinal and transverse components of the electric field with respect to (w.r.t) the current direction, μ_0, which depends on *N*, T_L, and $E_\perp$, is called the low field mobility because when $E_\parallel \to 0$, $\mu \to \mu_0$, T_c is the carrier temperature with the subscript *c* representing either *n* or *p* for electrons and holes, respectively, and the symbol $E_\parallel / T_c$ indicates the dependency is either on $E_\parallel$ or on T_c but not on both. In some models such as ones developed by Intel, the above expression is simplified to the following form to separate mobility reductions due to different field components:

$$\mu\,(N, T_L, E_\perp, E_\parallel / T_c) \;=\; \mu_0\,(N, T_L)\,\cdot r_{perp}\,(E_\perp)\,\cdot r_{para}\,(E_\parallel) \qquad (3.2)$$

where *r*'s are reduction factors and the subscripts *perp* and *para* represent the perpendicular and parallel components of the electric field.

In the following, we will first consider the low field mobility models without taking into account the effect of $E_\perp$. We then consider the effect of the transverse field dependence and finally discuss the longitudinal field or carrier-temperature dependence.

3.1.1 Low field Mobility Models

Without considering the transverse field dependence, the low field mobility can be written as $\mu_0(N, T_L)$. There are six such models available in PISCES-2ET.

3.1.1.1 Constant Mobility

This mobility model is invoked when none of the following logical parameters is included in the `model` card: `conmob`, `ccsmob`, `analytic`, `arora`, and `user1`. The mobility is then the function of material only and does not dependent on the doping density. However, the field dependency can still be included in the simulation when other parameters such as `fldmob` (for longitudinal) and `tfldmob` (for non-local transverse field) are specified in the `model` card. The values for four group of semiconductor materials are listed in TABLE 3.1.

TABLE 3.1

	μ_n (cm^2 V^{-1} s^{-1})	μ_p (cm^2 V^{-1} s^{-1})
Silicon / Ge_xSi_{1-x}	1500	450
GaAs / Al_xGa_{1-x} As	8500	400
InAs / $Al_xIn_{1-x}As$	22600	250
InP / $Ga_xIn_{1-x}As_yP_{1-y}$	4500	150

3.1.1.2 Table Look-up Doping Dependent Mobility

When the parameter `conmob` is specified in the `model` card, the doping dependent mobility model is invoked. The default doping dependent model is based on a table look-up of data for silicon if none of parameters `ccsmob`, `analytic`, `arora`, and `user1` is specified together with `conmob` in the `model` card(s). Tables of data for other materials are not available at present and constant value will be used instead with this option. The range of the data for doping concentrations in silicon is from $N = 1\times10^{14}$ to $1\times10^{21}\text{cm}^{-3}$. If the doping concentration falls out of the above range then the closest boundary value for the mobility is used, otherwise the following interpolation scheme is applied.

$$\mu_0(N) = \mu_0(N_1) + \frac{\mu_0(N_2) - \mu_0(N_1)}{\log(N_2/N_1)}\log\frac{N}{N_1} \tag{3.3}$$

where $N_2 > N > N_1$. This model is valid for silicon only. Note that if any of the following parameters: `ccsmob`, `analytic`, `arora`, and `user1`, is used with or without `conmob` specified in the `model` card, that model takes precedence and the precedency of these models is `user1`, `arora`, `analytic`, and `ccsmob` with each preceding one overriding the trailing ones.

3.1.1.3 Analytical Doping Dependent Mobility Model

This model is invoked by parameter `analytic` in the `model` card. The formulation for this model is as follows:

$$\mu_0(N, T_L) = \gamma\mu_{surf}(N, T_L) + (1-\gamma)\mu_{bulk}(N, T_L) \tag{3.4}$$

where γ is a coefficient with value between 0 and 1 depending on the relative distance from the surface/ interface, μ_{surf} and μ_{bulk} are mobilities in the surface layer and bulk, respectively. γ is determined by an ERFC function as follows:

$$\gamma = 0.5\cdot\text{erfc}\left(\frac{y - y_{int} - d}{\lambda}\right) \tag{3.5}$$

where the silicon and SiO_2 interface is assumed to be parallel to the *x*-axis, y_{int} is the *y* coordinate of the interface, *d* and λ are two parameters: one for offset and the other for the characteristic length. Their default values are $d = 0.06\mu$ and $\lambda = 0.03\mu$, and both can be accessed by users via parameters `int.off` and `char.int` in `mobility` card, respectively.

The surface and bulk mobilities have the same form of dependence on *N* and T_L as follows:

$$\mu(N, T_L) = \frac{\bar{T}_L^d}{1+\alpha(N)}\mu_{max} + \left[\frac{\alpha(N)}{1+\alpha(N)}\right]\mu_{min} \tag{3.6}$$

where $\alpha(N) = (N/N_0)^c \bar{T}_L^b$ and hereinafter $\bar{T}_L = T_L \text{ (in K)}/300$, the normalized lattice temperature. But *N* has different meaning for bulk and surface. For bulk, *N* is simply the total doping concentration, while for surface *N* must include the effect of the interface charge in the following manner:

$$N = N + cQ_{int} \tag{3.7}$$

where Q_{int} is interface charge (in units of cm^{-2}) as specified by parameter `qf` in the `interface` card, and *c* is a scaling factor in units of cm^{-1} with default value of 10^7 and can be changed through parameter `qss.conc` in `mobility` card. Note that the effect of the surface mobility is taken into account only for those nodes which are directly (in the *y*-direction) under the Si/SiO$_2$ interface. There is one exception, however. That is, when the flag `intelmob.par` is set in `model` card, the surface mobility (μ_{srf} in Eq. (3.4)) uses a different expression from Eq. (3.6). The new expression has the form of

$$\mu_{srf} = \mu_0 \bar{T}_L^{-a} \tag{3.8}$$

where parameters μ_0 and *a* for silicon are listed in Table 3.2.

Table 3.2

	μ_0 (cm^2 V^{-1} s^{-1})	*a*
electrons	783.5	1.67
holes	247.4	1.13

There are six parameters in Eq. (3.6): μ_{max}, μ_{min}, N_0, *b*, *c*, and *d*, and all are dependent upon the carrier type (electrons or holes) and region (bulk or surface). The parameter values used in the code for silicon are listed in Table 3.3.

Table 3.3

		μ_{max} (cm^2 V^{-1}s^{-1})	μ_{min} (cm^2 V^{-1}s^{-1})	$\boldsymbol{N_0}$ (cm^{-3})	*b*	*c*	*d*
electrons	bulk	1400	55.24	1.1×10^{17}	-3.8	0.73	-2.3
	surface	1200	1200	1.1×10^{17}	-3.8	0.73	-2.3
holes	bulk	500	49.7	1.6×10^{17}	-3.7	0.70	-2.2
	surface	500	49.7	1.6×10^{17}	-3.7	0.70	-2.2

3.1.1.4 Arora’s Doping and Temperature Dependent Mobility Model

An empirical mobility model based on the fitting to the measurement data for silicon at different lattice temperature has originally been proposed by Arora and *et al.* [5] and can be invoked by using parameter `arora` in the `model` card. The model has a general form of the following:

$$\mu_0(N, T_L) = \mu_{min} + \frac{\mu_{dlt}}{1 + (N/N_0)^{\alpha}} \qquad (3.9)$$

where parameters μ_{min}, μ_{dlt}, N_0, and α are all functions of T_L in the form of $a\bar{T}_L^b$ where both a and b are constants. This model is later extended to apply to GaAs by Yu [6] based on the available measured data at 77 and 300 K. The parameter values for silicon and GaAs are listed in Table 3.4.

Table 3.4

		$\mu^{\dagger}_{min}$	$\mu^{\dagger}_{dlt}$	N_0 (cm^{-3})	α
silicon	electrons	$88\bar{T}_L^{-0.57}$	$1252\bar{T}_L^{-2.33}$	$1.25\times10^{17}\bar{T}_L^{2.4}$	$0.88\bar{T}_L^{-0.146}$
	holes	$4.3\bar{T}_L^{-0.57}$	$407\bar{T}_L^{-2.23}$	$2.35\times10^{17}\bar{T}_L^{2.4}$	$0.88\bar{T}_L^{-0.146}$
GaAs	electrons	$2136\bar{T}_L^{-0.7475}$	$6331\bar{T}_L^{-2.687}$	$7.345\times10^{16}\bar{T}_L^{3.535}$	$0.6273\bar{T}_L^{-0.1441}$
	holes	$21.48\bar{T}_L^{-1.124}$	$31.2\bar{T}_L^{-2.366}$	$5.136\times10^{17}\bar{T}_L^{3.690}$	0.8057

† units of cm^2 V^{-1} s^{-1}

3.1.1.5 User Definable Arora’s Mobility Model

This model uses exactly the same formulation for doping and (lattice) temperature dependent mobility as Arora’s model discussed above. The only difference is that all the coefficients are put in an initializable subroutine `initum1` in file `usrmob1.f`. If users want to change some of the coefficients in the model, they have to change this file and re-compile the program. This model can be invoked by parameter `user1` in `model` card and is applied to silicon only.

3.1.1.6 "Carrier-Carrier Scattering" Mobility Model

Invoked by parameter `ccsmob` in `model` card, the so-called "carrier-carrier scattering" mobility model follows the approach of Dorkel and Leturcq [7]. The model formulation is as follows and the mobility depends on the carrier concentrations also (the name of "carrier-carrier scattering").

$$\mu(N, T_L, n, p) = \mu_L(T_L)\left[\frac{1.025}{1+\{(X(N, T_L, n, p))/1.68\}^{1.43}} - 0.025\right] \tag{3.10}$$

$$\mu_L(T_L) = \mu_{L0}\bar{T}_L^{-\alpha} \tag{3.11}$$

$$X(N, T_L, n, p) = \sqrt{\frac{6\mu_L[\mu_I(N, T_L) + \mu_{ccs}(T_L, n, p)]}{\mu_I(N, T_L)\,\mu_{ccs}(T_L, n, p)}} \tag{3.12}$$

$$\mu_I(N, T_L) = \frac{AT_L^{3/2}}{N}\left[\ln\left(1+\frac{BT_L^2}{N}\right) - \frac{BT_L^2}{N+BT_L^2}\right]^{-1} \tag{3.13}$$

$$\mu_{ccs}(T_L, n, p) = \frac{2\times10^{17}T_L^{3/2}}{\sqrt{pn}}\left[\ln\{1+8.28\times10^8 T_L^2 (pn)^{-1/3}\}\right]^{-1} \tag{3.14}$$

where the parameters μ_{L0}, α, *A*, and *B* are listed in Table 3.5.

Table 3.5

	μ_{L0} (cm^2 V^{-1}s^{-1})	α	*A* (cm^{-1}s^{-1}V^{-1}K$^{-3/2}$)	*B* (cm^{-3}K^{-2})
electrons	1500	2.2	4.61×10^{17}	1.52×10^{15}
holes	450	2.2	1×10^{17}	6.25×10^{14}

Note that in the program, *N* is taken as the min (*N*, 10^{19}) where *N* is in units of cm^{-3}.

3.1.2 Mobility Reduction due to Transverse Field

We now consider the effects of the electric field on the mobility. There are two modeling approaches: a general one as in Eq. (3.1) in which the effects of the transverse and longitudinal field components can be counted for separately, and a more specific form of Eq. (3.2) in which the field effects can be factorized. We first discuss the second approach in this section for the reason of simplicity. There are two such models available in the program, both were developed by Intel (`intelmob` and `intelmob.par` in `model` card). We then consider the Lombardi model (`lombardi`) which shares the same feature as Intel's ones in that both transverse and longitudinal field effects are considered simultaneously in the code. In all these three models, the transverse field can be computed either based on the structure or by definition. We will elaborate this at the end of this section.

3.1.2.1 Intel's Local Field Models

There are two mobility models developed by Intel for all doping, lattice temperature, and field (including both transverse and longitudinal) dependence with one single logical flag. They are described in this section. Both models follow the same formulation of Eq. (3.2), and $\mu_0(N, T_L)$ has the same expression as Eq. (3.4), the analytical model.

For model invoked by parameter `intelmob` in `model` card,

$$r_{perp} = (1 + E_{\perp}/E_{crit})^{-\beta} \tag{3.15}$$

where E_{crit} is a parameter with value of 4.2×10^4 V/cm for electrons and 3×10^4 V/cm for holes in silicon, and $\beta = 0.5$. These parameters are all accessible to users through `mobility` card.

The second model specified by `intelmob.par` has different expression for μ_{srf} (Eq. (3.8)) and more user-accessible parameters for the reduction due to the transverse field:

$$r_{perp} = \frac{1}{1 + (E_{\perp}/E_{univ})^{\alpha}} \tag{3.16}$$

where the parameters α and E_{univ} are listed in Table 3.6.

Table 3.6

	α	E_{univ} (V/cm)
electrons	1.02	5.71×10^5
holes	0.95	2.57×10^5

Both Intel's mobility models use $\mu_0(N, T_L, E_{\perp}) = r_{perp}(E_{\perp})\,\mu_0(N, T_L)$ for the low-field mobility.

3.1.2.2 Lombardi’s Model

Different from the above approaches of simply applying a reduction factor to the low field mobility, PISCES-2ET also provides a more physics based model to account for the effect of the transverse field. This model is developed based on extensive experimental data by Lombardi *et al.* [8] and can be invoked by parameter `lombardi` in `model` card. This transverse field dependent mobility is expressed by using Mathiessen’s rule:

$$\frac{1}{\mu(N, T_L, E_\perp)} = \frac{1}{\mu_{ac}(N, T_L, E_\perp)} + \frac{1}{\mu_{srf}(E_\perp)} + \frac{1}{\mu_0(N, T_L)} \tag{3.17}$$

where μ_{ac} is the mobility due to the acoustic phonon scattering, which depends on both the transverse field and lattice temperature, μ_{srf} is the one due to the surface scattering and is the function of the transverse field only, and $\mu_0(N, T_L)$ can use any model in Section 3.1.1. The expressions for μ_{ac} and μ_{srf} are as follows

$$\mu_{ac}(N, T_L, E_\perp) = \frac{B}{E_\perp} + \frac{\alpha N^\beta}{T_L E_\perp^{1/3}} \tag{3.18}$$

$$\mu_{srf}(E_\perp) = \frac{\delta}{E_\perp^2} \tag{3.19}$$

The parameters used in the above expressions are listed in Table 3.7 for electrons and holes, respectively.

Table 3.7

	B	α	β	δ
electrons	4.75×10^7	1.74×10^5	0.125	5.82×10^{14}
holes	9.93×10^7	8.84×10^5	0.0317	2.05×10^{14}

Note that the units for N, T_L and $E_\perp$ in the above two equations and in conjunction with the use of parameter values in Table 3.7 are cm^{-3}, K, and V/cm, respectively.

There are now problems remain unspecified in the above Intel's and Lombardi models. That is the evaluation of the transverse field and the reduction due to the longitudinal field. We discuss these issues in the following subsections.

3.1.2.3 Evaluation of Transverse Electric Field

The transverse field is defined as the field component the direction of which is perpendicular to the current flow, so, mathematically

$$E_{\perp} = \frac{|\boldsymbol{E} \times \boldsymbol{j}|}{|\boldsymbol{j}|} \tag{3.20}$$

PISCES-2ET uses this formula to evaluate $E_{\perp}$ in the above three mobility models if the parameter `strfld` is not specified in `model` card. Such evaluation is computationally costly and might also cause numerically instability. Another alternative and simplification is to evaluate the normal field component based on the geometric proximity to the Si/SiO$_2$ interface. Thus when `strfld` is specified, the following formula is used:

$$E_{\perp} = E_y e^{-(y - y_{int})/L} \tag{3.21}$$

where y is the coordinate in the axis perpendicular to the interface, E_y is the transverse field at the interface (y_{int}), and L is the characteristic length which has a value of 0.5μ.

3.1.2.4 Watt's Surface Mobility Model

J. Watt of Stanford proposed a simple approach to modeling the surface mobility in the inversion layer in silicon near the Si-SiO$_2$ interface [9]. The inversion layer is assumed to be confined only in the first grid spacing in the silicon at the direction perpendicular to the interface. Thus the interface is at the top of the inversion layer whereas the first grid in the silicon lies at the bottom of the inversion layer. The requirement for the meshing when this model is to apply is that the top grid spacing should be reasonably large (a couple of hundred angstroms), quite contrary to the conventional thought that the meshing should be dense at the substrate surface. The carrier mobility in the inversion layer, $\mu_{0,\,inv}$, depends on the effective transverse field, $E_{\perp,\,eff}$, in the inversion layer. The model accuracy has been verified against the measurement and is in agreement with the universal mobility model requirement [10]. This model can be invoked by using parameter `srfmob` in the `model` card. The effective transverse field, $E_{\perp,\,eff}$, is defined and calculated as:

$$E_{\perp,\,eff} = \frac{1}{\varepsilon_{si}}(\zeta Q_d + \eta Q_i) \tag{3.22}$$

where Q_d is the charge density (per unit surface area) in the inversion layer due to the ionized impurities and Q_i is the charge density due to the inverted (minority) carriers, and both ζ and η are physics-based fitting parameters. It is usually taken that $\zeta = 1$ for both carriers and $\eta = 0.5$ for electrons and $\eta = 0.33$ for holes. The dependence of the mobility on $E_{\perp,\,eff}$ is modeled in the following way by using Mathiessen's rule:

$$\begin{aligned}\frac{1}{\mu_{0,\,inv}} &= \frac{1}{\mu_{ph}} + \frac{1}{\mu_{sr}} + \frac{1}{\mu_{im}} \\ &= \frac{1}{\mu_{ref1}}\left(\frac{1\times10^6}{E_{\perp,\,eff}}\right)^{\alpha1} + \frac{1}{\mu_{ref2}}\left(\frac{1\times10^6}{E_{\perp,\,eff}}\right)^{\alpha2} \\ &+ \frac{1}{\mu_{ref3}}\left(\frac{1\times10^{18}}{N_B}\right)^{-1}\left(\frac{1\times10^{12}}{N_i}\right)^{\alpha3}\end{aligned} \tag{3.23}$$

where μ_{ph} is the mobility due to the phonon scattering, μ_{sr} is the one due to the surface roughness scattering, and μ_{im} is due to the scattering of the charged impurities in the inversion layer. And N_B is the doping density in the substrate (units of cm^{-3}), and N_i is the inverted carrier density per unit surface area in the channel (units of cm^{-2}). The other parameters are listed in Table 3.8.

Table 3.8

	η	μ_{ref1}	$\alpha1$	μ_{ref2}	$\alpha2$	μ_{ref3}	$\alpha3$
electrons (295K)	0.50	481	-0.160	591	-2.17	1270	1.07
electrons (77K)	0.75	5260	0.334	1700	-2.32	680	1.03
holes (295K)	0.33	92.8	-0.296	124	-1.62	534	1.02
holes (77K)	0.45	280	-0.516	213	-1.86	210	1.03

The modeling of mobility dependence on the transverse field can generally be categorized as two classes: local or non-local field dependence. The local field dependence refers to the fact that the transverse electric field is computed locally, whereas non-local field dependence is to find an effective transverse field to be used in the mobility formulation and this effective field is computed based on the knowledge over the entire inversion (or accumulation) layer in the silicon substrate at the Si/SiO$_2$

interface. All the models discussed so far are of local field nature. In next section we will discuss two non-local models, both developed at University of Texas, Austin.

3.1.2.5 Shin's Non-local Transverse Field Models

A non-local mobility model is the one in which the mobility is not solely determined by the local electric field and there are some non-local quantities in the mobility formulation. For example, the mobility in the inversion layer at the surface of the substrate below the gate oxide may also be affected by the inversion layer conditions (layer thickness, etc.). In the following, we first provide the model formulation and then state briefly the requirement on the user input. The first model, invoked by parameter `oldtfld` in `model` card has the following dependence [11]:

$$\mu(N, T_L, E_\perp, E_\parallel) = \frac{\mu_0(N, T_L, E_\perp)}{\sqrt{1+\left(\frac{\mu_0 E_\parallel}{v_{sat}}\right)^2}} + \frac{E_\perp - E_0}{\left[1+\left(\frac{\mu_0 E_\parallel}{v_{sat}}\right)^2\right]^{3/2}} \frac{d\mu_0}{dE_\perp} \tag{3.24}$$

where the dependence of μ_0 is listed only once. Note that all four parameters, N, T_L, $E_\perp$, and $E_\parallel$ are functions of space, so are μ and μ_0. The non-locality is represented by the quantity of E_0 which is defined as the transverse electric field at the edge (i.e. bottom) of the inversion layer. The inversion layer edge is in turn determined by the criterion that the inverted carrier concentration starts to drop below the doping density. The expression for μ_0 is

$$\mu_0(N, T_L, E_\perp) = \left(\frac{1}{1150\bar{T}_L^{-5/2}} + 3.2\times10^{-9}\frac{p}{z}\bar{T}_L^{1/2}\right)^{-1} \tag{3.25}$$

where

$$p = 0.09\bar{T}_L^{-3/2} + 1.5\times10^{-8}\frac{N_f}{n^{1/4}\bar{T}_L} \tag{3.26}$$

$$z = \frac{0.039\bar{T}_L}{\frac{E_\perp + E_0}{2}} + \frac{1.24\times10^{-5}}{\left(\frac{E_\perp + E_0}{2}\right)^{1/3}} \tag{3.27}$$

where n is the inverted carrier density in the inversion layer and in this particular case is the electron concentration for the n-channel MOSFET, and N_f is the fixed interface charge per unit area at the Si/

SiO_2 interface. To use this model, users need to provide the location of the gate oxide through parameters `ox.left`, `ox.right`, and `ox.bottom` in the `model` card.

An extended version of Schwarz-Russek formulation [12] is also available in the code by specifying `tfldmob` in the `model` card. The formulation and its parameters are described below:

$$\mu_0(N, T_L, E_\perp) = \left(\frac{1}{\mu_{ph}} + \frac{1}{\mu_{sr}} + \frac{1}{\mu_C}\right)^{-1} \tag{3.28}$$

where μ_{ph} is the mobility due to the scattering by phonons in both bulk and surface and fixed interface charge, and μ_{sr} is the one due to the surface roughness scattering, and μ_C is to the screened Coulomb scattering. The detailed expressions for those three mobilities can be found in [13].

3.1.3 Mobility Dependency on Longitudinal Field/Carrier Temperature

When the electric field along the current flow (called longitudinal field) becomes large, the carrier mobility is reduced. This reduction is on top of the reduction due to the transverse field which usually happens only at the semiconductor and insulator interface. The longitudinal field reduction of the mobility can be modeled either as the function of the local field if the field intensity is not very large or the spatial change of either the field or the doping concentration is not very rapid, or as the function of the carrier temperature if the energy relaxation process lags apparently behind that of the momentum relaxation, a phenomenon termed as the non-local effect. In the following, we first give a general formulation for the mobility reduction due to the longitudinal field. Then two models for GaAs are provided. One exhibits the behavior of the negative differential mobility when the field exceeds a critical value and is the accurate model for electrons in the bulk material. The other has a saturation velocity and is suitable for the surface channel mobility modeling. Finally we present two carrier-temperature dependent mobility models.

3.1.3.1 General Formulation for Field Dependent Mobility in Silicon

A general approach to modeling the effect of the longitudinal field on the carrier mobility is to multiply a reduction factor with the low-field mobility which may have taken into consideration the effect of the transverse field. Thus,

$$\mu\,(N, T_L, E_\perp, E_\parallel) \;=\; r_{para}\,(E_\parallel)\,\mu_0\,(N, T_L, E_\perp) \tag{3.29}$$

and the reduction factor r_{para} is modeled as (refer to [14])

$$r_{para} \;=\; \left[1 + \left(\frac{\mu_0\,(N, T_L, E_\perp)\,E_\parallel}{v_{sat}}\right)^{\beta}\right]^{-1/\beta} \tag{3.30}$$

where in silicon β is 1.395 for electrons and 1.215 for holes, and the saturation velocity, v_{sat}, has the expression in Table 3.9. The parameter β can be changed by the user through two parameters, `b.electrons` and `b.holes`, in the `model` card.

3.1.3.2 GaAs Mobility Models

It can be deduced from Eqs. (3.29)-(3.30) that the carrier drift velocity, which is equal to $\mu E_\parallel$, increases monotonically as $E_\parallel$ increase and saturates at v_{sat}. However, for electrons in the bulk GaAs due to the existence of several valleys with different effective mass in the conduction band structure, the drift velocity reaches a peak as the electric field is increased to a critical value and then the velocity decreases as the field further increases. This phenomenon, if viewed from the mobility modeling point of view, amounts to a negative differential mobility (defined as $dv/dE_\parallel$). To model this field dependence for electrons in GaAs, a model proposed first by Thim [15] is used as follows:

$$\mu\,(N, T_L, E_\parallel) \;=\; \frac{\mu_0\,(N, T_L) + \dfrac{v_{sat}}{E}\left(\dfrac{E_\parallel}{E_0}\right)^4}{1 + \left(\dfrac{E_\parallel}{E_0}\right)^4} \tag{3.31}$$

where E_0 is the critical field and has a default value of 4kV/cm, and $v_{sat} \;=\; 1.13\times10^7 - 1.2\times10^4 T_L$. It can be shown that when $E > E_0$ Eq. (3.31) leads to a negative differential mobility (NDM). This is the default field dependent mobility model for electrons in GaAs and the value of E_0 can be altered by the user through the parameter `E0` in the `model` card.

One problem related to the above formulation is that when applied to the simulation of GaAs MESFET, the drain output characteristics (current vs. voltage) may exhibit an unrealistic zig-zag behavior. One possible reason is that the correct mobility model in the bulk may not be suitable to the carriers in the surface channel. In the program, we provide another mobility model for electrons in GaAs, in which the carrier velocity approaches the saturation velocity with increased field monotonically in a manner of the hyperbolic tangent function. The model formulation is as follows [16] and does not exhibit NDM behavior.

$$\mu(N, T_L, E_{\parallel}) = \frac{v_{sat}}{E_{\parallel}} \tanh\left(\frac{\mu_0(N, T_L) E_{\parallel}}{v_{sat}}\right) \tag{3.32}$$

This model can be invoked through parameter `hypertang` in the `model` card.

3.1.3.3 Carrier Temperature Dependent Mobility in Si (Haensch's Model)

A carrier temperature instead of local-field dependent mobility model as proposed Haensch [17] is available in the program. The normal mobility dependences on doping, lattice-temperature, and vertical electric field in the Si-SiO$_2$ interface are all preserved in this model. The complete formulation for this mobility model as applied to silicon is as follows:

$$\mu(N, T_L, E_{\perp}, T_c) = \frac{\mu_0(N, T_L, E_{\perp})}{1 + \frac{3}{2}\alpha(N, T_L, E_{\perp}) k_B (T_c - T_L)} \tag{3.33}$$

where the subscript *c* stands for carriers, α is defined as being proportional to μ_0 in the following way:

$$\alpha(N, T_L, E_{\perp}) = \frac{\mu_0(N, T_L, E_{\perp})}{q\tau_w v_{sat}^2(T_L)} \tag{3.34}$$

where τ_w is the energy relaxation time and v_{sat} is the saturation velocity for carriers, respectively. Their default values for silicon are listed in Table 3.9 where T_L is in units K.

Table 3.9

	τ_w (ps)	v_{sat} (cm/s)
electrons	0.42	$8.64\times10^6 - 2.68\times10^3 T_L$
holes	0.25	$10.65\times10^6 - 2.68\times10^3 T_L$

3.1.3.4 Local Field Dependent Mobility Transformed from Haensch's Model

The following mobility model preserves the local field dependency, but it is transformed from Haensch's formulation (Eqs. (3.33)-(3.34)). The model is valid for silicon only and can be invoked through parameter `fmob.new` in `model` card.

$$\mu(N, T_L, E_\perp, E_{\parallel}) = \mu_0(N, T_L, E_\perp) \frac{2}{1 + \sqrt{1 + 4\left[\frac{\mu_0(N, T_L, E_\perp) E_{\parallel}}{v_{sat}}\right]^2}} \tag{3.35}$$

3.2 Thermal Conductivity Formula

The thermal conductivity, κ, in the expression of carrier-energy/heat flux plays equally important role as mobility does in the current density expression. In CHAPTER 2 we have given expressions of κ for carriers in terms of the mobility. For thermal conductivity of the lattice, κ_L, it is assumed that the power dependence on the lattice temperature is observed, i.e.,

$$\kappa_L(T_L) = \kappa_0 \bar{T}_L^{-\alpha} \tag{3.36}$$

where κ_0 is the thermal conductivity at T_L= 300 K and varies from material to material while α is a constant (1.2) for all materials. Table 3.10 lists κ_L for different materials as implemented in the code.

Table 3.10

material	κ_0 (W K^{-1}cm^{-1})	material	κ_0 (W K^{-1}cm^{-1})
silicon	1.45	oxide	0.25
GaAs	0.44	nitride	0.25
AlAs	0.91	sapphire	0.25
InAs	0.29	insulator	0.25
InP	0.80	$Al_{0.5}Ga_{0.5}As$	0.0903

3.3 Recombination and Generation Mechanisms

3.3.1 Impact Ionization Models

The generation rate of electron-hole pairs due to the carrier impact ionization (II) is generally modeled as [18][1]

$$G = \alpha_n n v_n + \alpha_p p v_p \tag{3.37}$$

where v_n and v_p are the electron and hole velocities, respectively, and the impact ionization rates, α_n and α_p, which are defined as the number of electron-hole pairs generated by a carrier per unit distance traveled. The II rate thus defined is a strong function of the local electric field or more accurately (as most researchers now agree) of local carrier temperature. PISCES-2ET provides several II rate models having either local field or carrier temperature dependence. In the following we will first discuss the more classical local field dependent model and provide the relevant material parameters and then the carrier temperature dependent models.

3.3.1.1 Local Field Dependent II Rate

$$\alpha = Ae^{-(b/E_{\|})} \tag{3.38}$$

where $E_{\|}$ is the electric field component which is parallel to the current flow, i.e., $E_{\|} = \boldsymbol{j} \cdot \boldsymbol{E} / |\boldsymbol{j}|$[2]. The parameters used in the above expression for different materials are listed in Table 3.11

Table 3.11

Parameter	Si	GaAs/Al$_x$Ga$_{1-x}$As	Ga$_x$In$_{1-x}$As$_y$P$_{1-y}$	Al$_x$In$_{1-x}$As
A_n (cm^{-1})	3.8×10^6 [18]	4.0×10^6 [19]	3×10^4 [20]	8.6×10^6 [21]
A_p (cm^{-1})	2.25×10^6 [18]	1.34×10^6 [18]	3×10^4	2.3×10^7 [21]
b_n (V/cm)	1.75×10^6 [18]	2.3×10^6 [19]	6.4×10^5 [20]	3.5×10^6 [21]

1. Correction has been made to the original formulation (Eq. (80) on p. 59 of [18]).
2. We do not consider in Eq. (3.38) the situation of $\boldsymbol{j} \cdot \boldsymbol{E} < 0$, i.e. the current is against the field direction.

Table 3.11

Parameter	Si	GaAs/Al$_x$Ga$_{1-x}$As	Ga$_x$In$_{1-x}$As$_y$P$_{1-y}$	Al$_x$In$_{1-x}$As
b_p (V/cm)	3.26×10^6 [18]	2.03×10^6 [18]	6.4×10^5	4.5×10^6 [21]
m_n	1 [18]	1 [19]	1 [20]	1 [21]
m_p	1 [18]	1[a]	1	1 [21]

a. 2 in [18] but by comparison to the experimental data 1 is more proper.

3.3.1.2 Carrier Temperature Dependent II Models

There are several ways to model the carrier temperature dependent impact ionization. One way is to map the carrier temperature to an effective electric field, E_{eff}, in replacing $E_{\|}$ in Eq. (3.38) and all the material parameters in the expression are kept unchanged. The mapping between T_c and E_{eff} is done through the following formulation (refer to [22]):

$$E_{eff} = \frac{5}{2}\frac{k_B}{q}\frac{T_c - T_L}{\gamma L_w} \tag{3.39}$$

where $L_w = \tau_w v_{sat}$ is called the energy relaxation length and γ is an adjustable parameter (for fitting the experimental data) which can be accessed by the user using `impjt.ratio` in the `model` card (default: 1.0). The second way is to approximate the actual carrier (drift) velocity with their saturation velocity and to use carrier-temperature dependent ionization rate as follows:

$$G = \alpha(T_n) n v_{sat,n} + \alpha(T_p) p v_{sat,p} \tag{3.40}$$

$$\alpha(T_c) = A_c e^{-T_c/B_c} \tag{3.41}$$

where c stands for either n or p. The values of A and B are listed inTable 3.12. This model can be

Table 3.12

	A (cm^{-1})	B (K)
electrons	3.8×10^6	1.75×10^6
holes	2.25×10^7	3.26×10^6

invoked through parameter `imp.nvt` in the `model` card.

The third modeling approach in developing carrier temperature dependent II is to abandon the generation rate dependence on the carrier (average) velocity at all. The rationing behind this approach is that when the carrier energy is higher than some threshold value the generation rate should be determined by the number of high energy carriers only no matter which direction they travel. This model can be invoked by parameter `imp.nt` in the `model` card, and the formulation and its parameters are as follows:

$$G = nA_n T_n^{B_n} e^{-C_n/T_n} + pA_p T_p^{B_p} e^{-C_p/T_p} \tag{3.42}$$

where the parameters *A*, *B*, and *C* are listed Table 3.13

Table 3.13

	A	*B*	*C*
electrons	5.5×10^{11}	4.0	0.8
holes	2.0×10^{13}	4.0	1.0

3.3.2 Photo-Generation

This model is invoked by parameter `photogen` in the `model` card. The generation rate due to photons is modeled as a forced generation term which is proportional to the photon flux, F_{ph}, and the intensity decays exponentially deep into the device along the incident path. The decay length is characterized by the absorption coefficient, α. The current implementation assumes only *y*-dependence:

$$g(y) = \alpha F_{ph} e^{-\alpha y} \tag{3.43}$$

where F_{ph} has units of cm^{-2}s^{-1} and α units of cm^{-1}. Both F_{ph} and α have to be specified by the user through parameters `flux` and `abs.coef` in the `model` card.

3.3.3 SRH Recombination

Shockley-Read-Hall (SRH) recombination rate is determined by the following formula:

$$r_{SRH} = \frac{np - n_i^2}{\tau_n [p + n_i e^{(E_i - E_t)/(k_B T_L)}] + \tau_p [n + n_i e^{(E_t - E_i)/(k_B T_L)}]} \tag{3.44}$$

where E_t is the energy level for the recombination centers and E_i is the intrinsic Fermi Energy. Carrier lifetimes, τ_n and τ_p, can be either constant or dependent upon the total doping concentration (parameter `consrh` in `model` card). The formula for the doping dependent is as follows:

$$\tau = \frac{\tau_0}{1 + N_{tot}/N_{SRH}} \tag{3.45}$$

where N_{tot} is the total doping concentration, and the default values for parameters τ_0 and N_{srh} are the same for both types of carriers in silicon and $\tau_0 = 1\times10^{-7}$ s and $N_{srh} = 5\times10^{16}$ cm^{-3}. If the interfacial recombination is to be included, it is through the change of carrier lifetime in the following way:

$$\frac{1}{\tau_{eff}} = \frac{ds_{int}}{A} + \frac{1}{\tau} \tag{3.46}$$

where A and d are partial area and interface length associated with the node in the element, respectively, and s_{int} is the interface recombination velocity.

3.3.4 Auger Recombination

This is a three-carrier recombination process, involving either two electrons and one hole or two holes and one electron. The dependence on n and p is as follows:

$$u_{Aug}(n, p) = (c_n n + c_p p)(np - n_i^2) \tag{3.47}$$

where the default values for c_n and c_p in silicon are 2.8×10^{-31} and 9.9×10^{-32} cm^6 s^{-1}, respectively.

CHAPTER 4
Material Properties for Heterostructures

In CHAPTER 2 and APPENDIX A, expressions are introduced for carrier transport in heterostructures. In order to perform the simulation, essential material parameters must be known. In this chapter, we will mainly discuss various material properties needed for device simulation and their dependence on the composition.

4.1 Material Parameters for Device Simulation

Material parameters in compound semiconductors depend strongly on the composition as well as (to a lesser degree) on the doping level. We start with identifying which material properties (parameters) are mostly concerned in the device simulation and then discuss the composition dependence in terms of mole fraction. Beside, the lattice temperature effect on some parameters will also be mentioned.

The minimum set of material parameters which have to be known in order to proceed the device simulation include:

1. Electron affinity, χ, and conduction band edge offset due to the composition change.
2. Energy bandgap, E_g.
3. Effective masses for electrons and holes, m_n^* and m_p^*, from which the effective densities of states for the conduction and valence bands, N_C and N_V, can be derived (Eq. (2.4)).
4. Static and high-frequency dielectric constants, ε_0 and ε_∞.

5. Electron and hole mobilities, μ_n and μ_p, and their dependence on composition, doping density, electric field, and temperature.
6. Minority carrier lifetime and corresponding coefficients for various recombination mechanisms including Auger and radiative.
7. Coefficients for the impact ionization.
8. Saturation velocity, v_{sat}, which is used as a parameter in certain field-dependent mobility models.

Before we address the detailed values and dependence of these parameters, however, we will first discuss the interpolation scheme for determining the compound material parameters based on the knowledge of the base materials which constitute the compound.

4.2 Interpolation Scheme for Composition Dependence

The parameters for ternary materials can be determined using the linear interpolation scheme from those of the constituent binary materials. But often this simple scheme is not accurate enough and the quadratic term, which is available for some materials from experimental data, has to be included. For examples, refer to Eq. (4.5) and other formulation for energy bandgaps in Section 4.4.1. The material parameters for quaternaries can be determined from those of ternaries using the following interpolation scheme.

$$\begin{aligned} Q(x,y) = {} & \frac{1}{x(1-x)+y(1-y)} \cdot \{ \\ & x(1-x)\,[\,(1-y)\,T_{\mathrm{Ga}_x\mathrm{In}_{1-x}\mathrm{P}} + yT_{\mathrm{Ga}_x\mathrm{In}_{1-x}\mathrm{As}}] + \\ & y(1-y)\;(1-x)\;[\,(1-x)\,T_{\mathrm{InAs}_y\mathrm{P}_{1-y}} + xT_{\mathrm{GaAs}_y\mathrm{P}_{1-y}}]\ \} \end{aligned} \tag{4.1}$$

where $\mathrm{Ga}_x\mathrm{In}_{1\text{-}x}\mathrm{As}_y\mathrm{P}_{1\text{-}y}$ is used as an example for quaternary material. We now provide a number of material parameters in PISCES-2ET as discussed below.

4.3 Data for Material Parameters

4.3.1 Parameters for Base Materials

Currently PISCES-2ET can handle four material systems: Ge_xSi_{1-x}, $Al_xGa_{1-x}As$, $Al_xIn_{1-x}As$, and $Ga_xIn_{1-x}As_yP_{1-y}$. Those compound materials can be derived from four types of base materials: Si, GaAs, InAs, and InP. A partial list of their respective material parameters are given in th following.

Table 4.1

	Si	GaAs	InAs	InP	units
$E_g^{\dagger}$	1.08 [23]	1.424 [24]	0.359 [28]	1.347 [26]	eV
E_g	1.17 [24]	1.519 [24]	0.4105 [28]	1.4205	eV
α	4.73 [24]	5.405 [24]	3.35 [29]	4.1	10^{-4} eV/K
β	636 [24]	204 [24]	248 [29]	136	K
ε_0	11.8 [25]	13.1 [24]	14.55 [29]	12.4 [26]	
ε_∞		10.9	12.25 [28]	9.55 [26]	
χ	4.17 [23]	4.07 [24]	4.9 [29]	4.4 [29]	eV
m_n^*	1.08 [25]	0.067 [26]	0.023 [26]	0.08 [26]	m_0
m_{lh}^*	0.16 [24]	0.074 [26]	0.024 [29]	0.089 [26]	m_0
m_{hh}^*	0.49 [24]	0.62 [26]	0.41 [29]	0.85 [26]	m_0
$N_C^{\dagger}$	2.8×10^{19} [24]	4.42×10^{17}	8.72×10^{16}	5.66×10^{17}	cm^{-3}
$N_V^{\dagger}$	1.04×10^{19} [24]	8.47×10^{18}	6.66×10^{18}	2.03×10^{19}	cm^{-3}
$\mu_n^{\dagger}$	1500 [24]	8500 [24]	22600 [29]	4500 [29]	cm^2/V.s
$\mu_p^{\dagger}$	450 [24]	400 [24]	250 [27]	150 [29]	cm^2/V.s
$v_{sat}^{\dagger}$		1.5×10^7 [27]	1×10^7 [23]	2.5×10^7 [23]	cm/ s

† $T_L = 300$ K

The lattice temperature dependent energy bandgap is modeled as follows:

$$E_g(T_L) = E_{g0} - \frac{\alpha T_L^2}{\beta + T_L} \tag{4.2}$$

where T_L is in K. When the data for the light and heavy holes are available, the overall hole effective mass used in the program is determined by

$$m_p^{*\,3/2} = m_{lh}^{*\,3/2} + m_{hh}^{*\,3/2} \tag{4.3}$$

We have also defined three auxiliary materials: Ge, AlAs, and GaP, which are used to constitute the desired ternaries and quaternary. Their material parameters relevant to simulation are listed below:

Table 4.2

Parameters	Ge	AlAs	GaP	units
$E_g^{\dagger}$	0.67 [25]	2.14 [28]	2.24 [25]	eV
E_{g0}	0.744 [24]	2.23 [28]	2.4 [25]	eV
α	4.77 [24]	4.0 [28]	5.4 [25]	10^{-4} eV/K
β	235 [24]			K
ε_0	15.8 [25]	10.06 [28]	10.2 [25]	
ε_∞		8.16 [28]	8.90 [29]	
χ	4.0 [25]	3.62 [23]	4.0 [29]	eV
m_n^*	0.55 [25]	0.124 [28]	0.17 [26]	m_0
m_{lh}^*	0.044 [24]	0.26 [28]	0.14 [26]	m_0
m_{hh}^*	0.28 [24]	0.5 [28]	0.79 [26]	m_0
$N_C^{\dagger}$	1.04×10^{19} [24]			cm^{-3}
$N_V^{\dagger}$	6.0×10^{18} [24]			cm^{-3}

† $T_L = 300$ K

Now we discuss how the parameters for the compound materials are determined, starting with the bandgap.

4.4 Band Structure Parameters

4.4.1 Bandgap

We will discuss first those compound semiconductors which need only one mole fraction to determine the composition, that is, either Ge_xSi_{1-x} or ternaries. For Ge_xSi_{1-x}, currently only the data for the strained layer [30] is accessible in the program. The formulation is as follows (in units of eV):

$$\begin{aligned}
E_g(x) &= E_{g,Si} - \frac{1.16-0.945}{0.245}x && x \le 0.245 \\
&= 0.945 - \frac{0.945-0.87}{0.35-0.245}(x-0.245) && 0.245 < x \le 0.35 \\
&= 0.87 - \frac{0.87-0.78}{0.5-0.35}(x-0.35) && 0.35 < x \le 0.5 \\
&= 0.78 - \frac{0.78-0.72}{0.6-0.5}(x-0.5) && 0.5 < x \le 0.6 \\
&= 0.72 - \frac{0.72-0.69}{0.675-0.6}(x-0.6) && 0.6 < x \le 0.675 \\
&= 0.69 - \frac{0.69-0.66}{0.735-0.675}(x-0.675) && 0.675 < x \le 0.735 \\
&= 0.66 && x > 0.735
\end{aligned} \tag{4.4}$$

For $Al_xGa_{1-x}As$, simply use

$$\begin{aligned}
E_g(x) &= 1.424 + 1.247x && x \le 0.45 \quad \text{(direct band)} \\
&= 1.900 + 0.125x + 0.143x^2 && x > 0.45 \quad \text{(indirect band)}
\end{aligned} \tag{4.5}$$

For bandgap of $Al_xIn_{1-x}As$, as x is increased from 0 to 1 the band structure also undergoes a transition from the direct to indirect bandgap and the critical composition is $x = 0.56$. The expression for E_g at room temperature is as follows [31]:

$$\begin{aligned}
E_g &= 2.95x + 0.36(1-x) - 0.7x(1-x) && x \le 0.56 \quad \text{(direct)} \\
&= 2.16x + 1.37(1-x) - 0.7x(1-x) && x > 0.56 \quad \text{(indirect)}
\end{aligned} \tag{4.6}$$

where when $x < 0.56$ the (direct) bandgap is determined by the Γ valley of the conduction band ($E_g(\mathrm{AlAs}) = 2.95\mathrm{eV}$, $E_g(\mathrm{InAs}) = 0.36\mathrm{eV}$), and when $x > 0.56$ the (indirect) bandgap is determined by the *X* valley ($E_g(\mathrm{AlAs}) = 2.16\mathrm{eV}$, $E_g(\mathrm{InAs}) = 1.37\mathrm{eV}$.)

Before giving the expression for the bandgap in the quaternary $\mathrm{Ga}_x\mathrm{In}_{1\text{-}x}\mathrm{As}_y\mathrm{P}_{1\text{-}y}$. we first list the expressions for the bandgaps of four ternary materials, on which $\mathrm{Ga}_x\mathrm{In}_{1\text{-}x}\mathrm{As}_y\mathrm{P}_{1\text{-}y}$ is based.

$$E_g(\mathrm{Ga}_x\mathrm{In}_{1-x}\mathrm{P}) = 1.35 + 0.643x + 0.786x^2 \tag{4.7}$$

$$E_g(\mathrm{Ga}_x\mathrm{In}_{1-x}\mathrm{As}) = 0.36 + 0.505x + 0.555x^2 \tag{4.8}$$

$$E_g(\mathrm{In\,As}_y\mathrm{P}_{1-y}) = 1.35 - 1.083y + 0.091y^2 \tag{4.9}$$

$$_g(\mathrm{Ga\,As}_y\mathrm{P}_{1-y}) = 2.74 - 1.473y + 0.146y^2 \tag{4.10}$$

And for the strained $\mathrm{Ga}_x\mathrm{In}_{1\text{-}x}\mathrm{As}$ layer, the bandgap is evaluated according to the expression provided by Corzine *et al.* [32]:

$$E_g(\mathrm{Ga}_x\mathrm{In}_{1-x}\mathrm{As}) = 1.424 - (1-x)\{1.061 - (1-x)[0.07 + 0.03(1-x)]\} \tag{4.11}$$

If either *x* or *y* in $\mathrm{Ga}_x\mathrm{In}_{1\text{-}x}\mathrm{As}_y\mathrm{P}_{1\text{-}y}$ has the value of zero or one, then one of the expressions, Eqs. (4.7) - (4.10), is used to evaluate the bandgap, otherwise the following general formula is used to evaluate the (lowest-direct) bandgap in $\mathrm{Ga}_x\mathrm{In}_{1\text{-}x}\mathrm{As}_y\mathrm{P}_{1\text{-}y}$ [26]:

$$\begin{aligned} E_g(x,y) = \frac{1}{x(1-x) + y(1-y)} & \{x(1-x)[(1-y)E_g(\mathrm{Ga}_x\mathrm{In}_{1-x}\mathrm{P}) + yE_g(\mathrm{Ga}_x\mathrm{In}_{1-x}\mathrm{As})] \\ & + y(1-y)(1-x)[(1-x)E_g(\mathrm{InAs}_y\mathrm{P}_{1-y}) + xE_g(\mathrm{GaAs}_y\mathrm{P}_{1-y})]\} \end{aligned} \tag{4.12}$$

The values obtained from the above expression is for room temperature ($T_L = 300\mathrm{K}$).

4.4.2 Electron Affinity and Band-edge Offsets

The band edge offsets caused by the composition change are critical in determining the heterostructure device characteristics. In PISCES-2ET, the band edge offsets are actually accounted for by the change of the electron affinity and the bandgap. Even in the homogeneous material, if the bandgap narrowing occurs due to the heavy doping effect, the band edges shift towards the center of the bandgap. It is usually assumed in this case that the amount of the edge shift for both the conduction and valence bands are the same. For the current implementation, except of $Al_xGa_{1-x}As$, $Al_xIn_{1-x}As$, and $Ga_xIn_{1-x}As_yP_{1-y}$, it is assumed that all bandgap change due to the composition change occurs only at the valence edge due to the lack of sufficient experimental data. However, the bandgap narrowing due to the heavy doping effect, is accounted for in the aforementioned manner for all materials, i.e., evenly split. For $Al_xGa_{1-x}As$ / GaAs, $\Delta E_C : \Delta E_V$ is taken as 0.6 : 0.4 and for $Al_xIn_{1-x}As$ / InAs, 0.706 : 0.294, where ΔE is considered positive when the band edge is expanding, that is the bandgap becomes larger. For $Ga_xIn_{1-x}As_yP_{1-y}$ / InP the bandgap change is simply assumed evenly split among the conduction and valence bands edges[1]. By designating the percentage of the shift of conduction band edge with respect to the bandgap change as γ, we have the following relationship for the electron affinity as function of the mole fractions:

$$\chi(x, y) = \chi(0, 0) - \gamma[E_g(x, y) - E_g(0, 0)] \tag{4.13}$$

For ternaries, y does not appear in the formula. And

$$\Delta E_C = -\Delta\chi \tag{4.14}$$

$$\Delta E_V = \Delta\chi + \Delta E_g \tag{4.15}$$

4.4.3 Effective Mass and Density of States

The calculation of the effective density of states is based on the density-of-state effective mass, which in turn depends on the mole fraction. Moreover, one needs to consider for electrons the effective masses at different valleys in the conduction band and for holes the light and heavy masses in order to construct the overall effective masses used in computing the densities of states. We use a different formulation to compute effective masses for each different material system based on our best knowledge. For $Al_xGa_{1-x}As$ / GaAs material system, since we know composition dependence of electron effective mass and band gap at each valley (Γ, L, and X), we can use weighted scheme to compute the electron effective mass (for details, see [33]). And for holes, the following linear formula is used:

1. $\Delta E_C = (71 \pm 7)\,\%\,\Delta E_g$ is measured for $Al_{0.48}In_{0.52}As$ / $Ga_{0.47}In_{0.53}As$ in [34].

$$\frac{m_p^*(x)}{m_0} = 0.48 + 0.31x \tag{4.16}$$

For $Al_xIn_{1-x}As$ / InAs system, we simply use the linear interpolation scheme for both electrons and light and heavy holes. That is,

$$m^*(x) = (1-x)\, m^*_{\mathrm{InAs}} + x m^*_{\mathrm{AlAs}} \tag{4.17}$$

where the effective masses for InAs and AlAs can be found in Table 4.1 and Table 4.2, respectively.

For $Ga_xIn_{1-x}As_yP_{1-y}$ / InP, the interpolation scheme based on the constituent binaries is used for obtaining the effective masses for electrons and light and heavy holes. For electrons, we don't distinguish between the bulk and the strained layer. The interpolation scheme is as follows:

$$\begin{aligned} m_n^* = {} & (1-x)\, y m^*_{n,\,\mathrm{InAs}} + (1-x)\,(1-y)\, m^*_{n,\,\mathrm{InP}} + \\ & x y m^*_{n,\,\mathrm{GaAs}} + x\,(1-y)\, m^*_{n,\,\mathrm{GaP}} \end{aligned} \tag{4.18}$$

For holes, the effective masses of the light and heavy holes in the bulk material are calculated in the same manner. However, for the strained layer, they are taken as constants according to Corzine in [35][1].

$$m^*_{lh} = 0.078 m_0 \tag{4.19}$$

$$m^*_{hh} = 0.165 m_0 \tag{4.20}$$

4.5 Dielectric Constants

Different interpolation schemes are used for different compound materials. For ternaries $Al_xGa_{1-x}As$ and $Al_xIn_{1-x}As$, we use the same scheme as in SEDAN [33] for $Al_xGa_{1-x}As$. That is, by knowing the dielectric constants at high and low frequencies for the constituent binaries, the respective dielectric constant is evaluated using

1. We have intentionally flipped the data for the light and hole holes because of suspicion of the mistake incurred in the cited paper.

$$\varepsilon = \frac{1 + 2\left[x\frac{\varepsilon_1 - 1}{\varepsilon_1 + 2} + (1 - x)\frac{\varepsilon_2 - 1}{\varepsilon_2 + 2}\right]}{1 - x\frac{\varepsilon_1 - 1}{\varepsilon_1 + 2} - (1 - x)\frac{\varepsilon_2 - 1}{\varepsilon_2 + 2}} \tag{4.21}$$

where ε_1 is that of AlAs for both the materials, and ε_2 is that of GaAs for AlGaAs and of InAs for AlInAs. The formulation applies to both high and low frequencies.

The dielectric constants for Ge_xSi_{1-x} are obtained by using the linear interpolation for those of Ge and Si. For the quaternary $Ga_xIn_{1-x}As_yP_{1-y}$, the interpolation scheme in Eq. (4.18) is used based on the data for the constituent binaries.

4.6 Mobilities

Due to the lack of experimental data, the mobility dependences on composition, doping, lattice temperature, and field are not complete for most of the compound materials. Whenever data are not available constant mobility as listed in Section 3.1.1.1 is used. Readers are referred to [29] for data for many other materials, even though the collection of data is rather out-of-date (1971).

4.7 Impact Ionization

For relevant formulas and material parameters, see Section 3.3.1.1.

CHAPTER 5
Numerical Techniques

In this chapter various numerical techniques and data structure will be discussed. Since the development of PISCES has spanned over a decade, we only address here the improvement over the previous versions (II and II-B). Interested readers are referred to reports [3] and [4].

5.1 High and Zero Frequency AC Analyses

The algorithm for AC analysis in the device simulation mostly follows Laux's approach [36] and is based on the perturbation method applied to the DC solution. Before we discuss the improvement in the original implementation (PISCES IIB), the mathematical essence of the AC analysis is first described.

5.1.1 General Principle of AC Small Signal Analysis

Consider $\boldsymbol{x}$ as the column vector of all the basic variables to be solved such as ψ, n and p, and column vector $\boldsymbol{p}$ represents all the applied terminal bias, the task of DC analysis can be considered as the solution to the following nonlinear, algebraic equation:

$$\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{p}) = 0 \tag{5.1}$$

where both the equation vector (also a column vector) $\boldsymbol{F}$ and variable vector $\boldsymbol{x}$ have dimension of N, the number of all basic variables in the simulation region, i.e., $\boldsymbol{F},\ \boldsymbol{x} \in R^N$, while the parameter vector

$\boldsymbol{p}$ has the dimension of M, no bigger than the total number of the device terminals, or $\boldsymbol{p} \in R^M$. For analyses involving the time evolution such as transient and AC analyses, the above equation must be modified to include the time derivative part. Thus,

$$\boldsymbol{F} + \boldsymbol{f} = 0 \tag{5.2}$$

where

$$\boldsymbol{f} = D\frac{\partial \boldsymbol{x}}{\partial t} \tag{5.3}$$

where D is a diagonal matrix ($N \times N$) with rows corresponding to the Poisson's equation zero and others -1. We now apply bias of form

$$\boldsymbol{p} = \bar{\boldsymbol{p}} + \tilde{\boldsymbol{p}} e^{j\omega t} \tag{5.4}$$

That is, an AC signal of angular frequency ω and amplitude $\tilde{\boldsymbol{p}}$ (notice that the multi-dimensionality represents the bias at all terminals) which is generally a vector of complex numbers, superimposed on the DC bias, $\bar{\boldsymbol{p}}$. The response, that is the solution vector $\boldsymbol{x}$, should also have the form of

$$\boldsymbol{x} = \bar{\boldsymbol{x}} + \tilde{\boldsymbol{x}} e^{j\omega t} \tag{5.5}$$

where vector $\bar{\boldsymbol{x}}$ is the DC response and are the real numbers, and $\tilde{\boldsymbol{x}}$ is a phasor vector and are complex numbers due to the change of phase in response to the input signal. For the small signal analysis, $\|\tilde{\boldsymbol{p}}\| \to 0$ where $\| \quad \|$ represents the norm of the vector, hence $\tilde{\boldsymbol{x}} \to 0$. Substituting Eqs. (5.4) and (5.5) into Eq. (5.2) and applying the Taylor expansion to Eq. (5.2), by retaining only terms which have the first order of $e^{j\omega t}$ on obtains the following equation:

$$\boldsymbol{F}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{p}}) + [\boldsymbol{F}_{\boldsymbol{x}}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{p}})\tilde{\boldsymbol{x}} + \boldsymbol{F}_{\boldsymbol{p}}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{p}})\Delta\boldsymbol{p} + j\omega D\tilde{\boldsymbol{x}}]e^{j\omega t} = 0 \tag{5.6}$$

Note that in the above we have replaced $\tilde{\boldsymbol{p}}$ with a real number vector $\Delta\boldsymbol{p}$, i.e. we assume all the input signals are in the same phase. Furthermore, because both $\boldsymbol{x}$ and $\boldsymbol{p}$ are vectors the derivatives of $\boldsymbol{F}_{\boldsymbol{x}}$ and $\boldsymbol{F}_{\boldsymbol{p}}$ are actually matrices with $\boldsymbol{F}_{\boldsymbol{x}}$ being a square one while $\boldsymbol{F}_{\boldsymbol{p}}$ is generally not.

With the DC solution already known, i.e. $\boldsymbol{F}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{p}}) = 0$, Eq. (5.6) can be simplified to the following:

$$(\boldsymbol{F}_{\boldsymbol{x}} + j\omega D)\tilde{\boldsymbol{x}} = -\boldsymbol{F}_{\boldsymbol{p}}\Delta\boldsymbol{p} \tag{5.7}$$

and the only unknown variable is $\tilde{\boldsymbol{x}}$, the AC response. Because $\tilde{\boldsymbol{x}}$ is a complex number, we can write $\tilde{\boldsymbol{x}} = x_r + jx_i$ where x_r and x_i are real and imaginary parts, respectively, and $j = \sqrt{-1}$, Eq. (5.7) then becomes a set of two equations in real number mathematics:

$$\boldsymbol{F_x x_r} - \omega D\boldsymbol{x_i} = -\boldsymbol{F_p}\Delta\boldsymbol{p} \tag{5.8}$$

$$\omega D\boldsymbol{x_r} + \boldsymbol{F_x x_i} = 0 \tag{5.9}$$

These two equations are coupled to each other so to find $\boldsymbol{x}_r$ and $\boldsymbol{x}_i$, they have to be solved simultaneously in principle. Because each of vectors $\boldsymbol{x}_r$ and $\boldsymbol{x}_i$ has dimension of N, then we must solve a system of equations with dimension of $2N \times 2N$ which needs to quadruple the memory storage as is required for the DC analysis. Because the AC analysis is usually not performed as routinely as the DC analysis, to set aside a large trunk of memory space solely for the AC analysis is not an economic choice especially when the program is not written to take the advantage of dynamic memory allocation. It is preferable, therefore, to use an iterative method to solve the above equation. One popular method, also the one used in PISCES IIB is the so-called SOR (Successive Over-Relaxation) method [36]. The essence of this method can better be seen from the following matrix equation form:

$$A\boldsymbol{x} = \begin{pmatrix} J & -\omega D \\ \omega D & J \end{pmatrix} \begin{pmatrix} \boldsymbol{x_r} \\ \boldsymbol{x_i} \end{pmatrix} = \begin{pmatrix} B \\ 0 \end{pmatrix} \tag{5.10}$$

where we have altered the symbols used in Eqs. (5.8)-(5.9) with the Jacobian matrix $J = \boldsymbol{F_x}$ and the stimulus $B = -\boldsymbol{F_p}\Delta\boldsymbol{p}$ which has non-zero (real number) terms only for rows corresponding to the Poisson's equation. If the magnitude of the off-diagonal elements, i.e., ωD, in the above block matrix equation are small compared to the diagonal elements, then the iterative approach by repeating the loop of first solving

$$J\boldsymbol{x_r} = B + \omega D\boldsymbol{x_i} \tag{5.11}$$

for $\boldsymbol{x}_r$ while fixing $\boldsymbol{x}_i$ and then solving

$$J\boldsymbol{x_i} = -\omega D\boldsymbol{x_r} \tag{5.12}$$

for $\boldsymbol{x}_i$ using the recently updated $\boldsymbol{x}_r$, will converge easily. But as the ω increases, the convergence becomes poorer and eventually fails to converge when the frequency approaches the cutoff frequency, f_T, of the device under analysis [36]. A final note regarding SOR method is the use of relaxation factor.

Instead of employing the above simple iteration procedure (Eqs. (5.11) and (5.12)), the actual iteration takes form of

$$x_r^{k+1} = (1-\nu)\, x_r^k + \nu J^{-1}\,(B + \omega D x_i^k) \tag{5.13}$$

$$x_i^{k+1} = (1-\nu)\, x_i^k + \nu J^{-1}\,(-\omega D x_r^{k+1}) \tag{5.14}$$

where ν is the relaxation factor and its value ranges from 0 to 2, and k is the iteration count. When $\nu = 1$, Eqs. (5.13) and (5.14) are reduced to Eqs. (5.11) and (5.12). One of the most challenging tasks in SOR method is to determine the optimized value of ν and we will discuss later other iteration schemes which avoid using the relaxation factor at all.

5.1.2 Matrix Transformation to Improve Diagonal Dominance

The diagonal dominance of the coefficient matrix for the linear equation system at high frequency can be improved by performing a matrix transformation or called pre-conditioning. A properly chosen matrix T is left-multiplied to the both sides of (5.10) such that the resulted coefficient matrix, $A_T = TA$, will have a better or roughly the same diagonal dominance when the frequency increases. This transformation matrix can indeed be constructed as follows. Considering the coefficient matrix A in Eq. (5.10) as a 2×2 block matrix, the presence of the off-diagonal blocks with opposite signs actually enhances the determinant of the entire matrix. This observation suggests the use of the following transformation matrix [37]:

$$T = \begin{pmatrix} I & \omega D D_J^{-1} \\ -\omega D D_J^{-1} & I \end{pmatrix} \tag{5.15}$$

where I is the identity matrix and D_J is the diagonal matrix consisting of only the diagonal elements in J. The resulting equation is thus

$$A_T \boldsymbol{x} = \begin{pmatrix} J + \omega^2 D D_J^{-1} D & \omega D D_J^{-1} J_0 \\ -\omega D D_J^{-1} J_0 & J + \omega^2 D D_J^{-1} D \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_r \\ \boldsymbol{x}_i \end{pmatrix} = \begin{pmatrix} B \\ 0 \end{pmatrix} \tag{5.16}$$

where $J_0 = J - D_J$ and the property of $DB = 0$ has been used to arrive at the above equation[1]. Note that because D_J is a diagonal matrix so its inverse D_J^{-1} is trivial to compute. It becomes immediately clear from Eq. (5.16) that although the magnitude of the off-diagonal blocks still increases proportionally with ω, that of the diagonal entries may increase even faster due to the additional term which is proportional to ω^2, thus avoiding the problem of diagonal blocks being overwhelmed by the off-diagonals. There are three distinctive ranges of frequencies where the transformed matrix shows different characteristics. At low frequencies, which can be quantified by all the non-zero entries in $\omega D D_J^{-1}$ being much less than unity, off-diagonal blocks can be considered as small perturbations with respect to the diagonal blocks and iterative algorithms should do well with or without the transformation. At very high frequencies, where all the non-zero entries of $\omega D D_J^{-1}$ are much greater than unity, the enhancement to the diagonal blocks makes the entire matrix become increasingly diagonally dominant, which implies the improved performance for the iterative algorithm. At the intermediate frequencies between these extremes, however, the situation is not as clearly cut since the off-diagonal blocks can no longer be considered as small perturbations while the growth in the enhancements to the diagonal blocks has not caught up with the growth in the off-diagonals.

5.1.3 Pre-conditioned TFQMR Method for AC Analysis

As mentioned in Section 5.1.1, the choose of an optimized relaxation factor in SOR method is a difficult task. It is, therefore, much preferable to have an iterative method which does not require any relaxation factor, yet still keep the efficient usage of the memory space. In PISCES-2ET, we employ an iterative method called TFQMR (Transpose-Free Quasi-Minimal Residual) method [38]. The major difference from the conventional SOR method is, in addition to the elimination of the relaxation factor, that it updates the entire solution array, $(\boldsymbol{x}_r, \boldsymbol{x}_i)^T$, simultaneously. Before we describe the detailed procedure flow of this method, however, we will discuss the pre-conditioning technique further more. For any iterative method to converge fast in solving linear equation system, it is always desirable to have a near identity coefficient matrix. The near identity of a matrix can be achieved by pre-conditioning, that is, to multiply a certain matrix (called pre-conditioner) on the both sides of the equation. A simple and natural choice of the pre-conditioner would be the block diagonal matrix derived from the original coefficient matrix. With such a pre-conditioner, only back substitutions are

1. The right hand side of Eq. (5.10) has been multiplied by T but the result remains the same as the original.

needed during iteration. Designating the pre-conditioner as $P = M^{-1}$, then the matrix M for Eq. (5.16) will be

$$M = \begin{pmatrix} J + \omega^2 D D_J^{-1} D & 0 \\ 0 & J + \omega^2 D D_J^{-1} D \end{pmatrix} \tag{5.17}$$

and

$$M = \begin{pmatrix} J & 0 \\ 0 & J \end{pmatrix} \tag{5.18}$$

for Eq. (5.10). Readers can verify that by using these pre-conditioners, the resulting coefficient matrices have identity matrices as their diagonal blocks. At the low frequency, the off-diagonal blocks are very small in magnitude, leading to a quick convergence.

As an iterative algorithm, the SOR method has the advantage of simplicity and minimal requirement for the data storage. In practice, however, we have found that a proper relaxation coefficient is difficult to obtain. In contrast, there are several attractive advantages with the so-called QMR (Quasi-Minimal-Residual) iterative algorithm [39]. Until quite recently, all existing iterative algorithms either use non-minimization procedure in generating the update vectors, which results in a very volatile residual reduction process, or use a full minimization procedure which requires increasing data storage as the iteration proceeds. By applying a quasi-minimal residual approach, QMR is able to retain the advantage of requiring only the fixed amount of data storage. At the same time, the residual reduction process becomes very smooth with only occasional, insignificant increases. Furthermore, like all other Krylov subspace methods based on the Lanzos vectors, no external parameters are needed [39]. The particular QMR algorithm we have implemented in PISCES-2ET is the TFQMR (Transpose-Free Quasi-Minimal-Residual) [38] with the pre-conditioner M in either Eq. (5.17) or Eq. (5.18) applied before iteration. The detailed procedure of TFQMR with pre-conditioner M is summarized as follows:

1. Initialization:

   (a) $b = M^{-1}B$;

   (b) $y_1 = u_0 = r_0 = \tilde{r} = b,\ x_0 = d_0 = 0;$

```
        (c) $\rho_0 = \tau_0 = \|r_0\|, \theta_0 = \eta_0 = 0;$
    2. For $n = 1, 2, \dots$ do
        (a) $\nu_{n-1} = M^{-1}Ay_{2n-1};$
        (b) $\sigma_{n-1} = \tilde{r} \cdot v_{n-1}, \alpha_{n-1} = \rho_{n-1}/\sigma_{n-1};$
        (c) $y_{2n} = u_{n-1} - \alpha_{n-1}v_{n-1};$
        (d) $r_n = r_{n-1} - \alpha_{n-1}M^{-1}A(u_{n-1} + y_{2n-1});$
        (e) $\omega_{2n+1} = \|r_n\|, \omega_{2n} = \sqrt{\|r_n\| \cdot \|r_{n-1}\|};$
        (f) For $m = 2n-1, 2n$ do
            • $d_m = y_m + (\theta_{m-1}^2\eta_{m-1}/\alpha_{n-1})\,d_{m-1};$
            • $\theta_m = \omega_{m+1}/\tau_{m-1}, c_m = 1/\sqrt{1+\theta_m^2};$
            • $\tau_m = \omega_{m+1}c_m, \eta_m = c_m^2\alpha_{n-1};$
            • $x_m = x_{m-1} + \eta_m d_m;$
            • If $x$ has converged: stop.
        (g) $\rho_n = \tilde{r} \cdot r_n, \beta_n = \rho_n/\rho_{n-1};$
        (h) $u_n = r_n + \beta_n y_{2n};$
        (i) $y_{2n+1} = u_n + \beta_n(y_{2n} + \beta_n y_{2n-1});$
```

In [38], $\sqrt{1+m}\tau_m$ is given as an upper limit for the residual norm. In our implementation, the residual norm is initially assumed to be equal to τ_m (a coefficient of unity). When τ_m drops below the tolerance, the true residual norm is evaluated, which also gives out the real proportional coefficient between the residual norm and τ_m. If the residual norm is not below the tolerance, the iteration continues with the new coefficient.

The effectiveness of the TFQMR with pre-conditioning in AC analysis has been tested in an analysis of GaAs MESFET [40]. There is essentially no limitation as to how high the frequency can be applied to the device during simulation and the agreement between the measured and simulated results are very good.

5.1.4 Zero Frequency AC Analysis

As the conclusion of this section, we present a special case in the AC analysis where a direct method can be used and the results of this type of analysis have broad applications. This is the case when the frequency approaches zero. In this case, $\boldsymbol{x}_r$ can be found directly from Eq. (5.8) by setting $\omega = 0$. It would be tempted to conclude that $x_i = 0$ from Eq. (5.9) using the same reasoning. By doing so, however, valuable information will be lost. Consider the case where $\boldsymbol{x}_i$ is proportional to the frequency, which actually often occurs in the device characterization. For example, in capacitance $x_i = \omega C$. The linear coefficient can then be extracted from solving Eq. (5.9) even when $\omega \to 0$. A practical application would be the evaluation of the junction capacitance including both the depletion and diffusion components. Thus by assuming $\boldsymbol{x}_i = \omega \boldsymbol{C}$ where $\boldsymbol{C}$ is an array of trans-capacitances, one can find $\boldsymbol{C}$ by solving the following equation

$$\boldsymbol{F}_{\boldsymbol{x}} \boldsymbol{C} = -D\boldsymbol{x}_r \tag{5.19}$$

Information such as the change of the storage charge from the quasi steady state analysis can thus be obtained by using the zero-frequency AC analysis, which involves only two more solutions of a linear equation with the coefficient matrix the same as the Jacobian for the DC solution but different right hand side terms.

5.2 Discretization Scheme for Carrier and Energy Fluxes

In assembling the node equations for continuity equations, a key issue is to express the carrier/energy flux along the grid line using the variables on the two ends of the line. A classic discretization scheme for the current (or the carrier flux) as function of the carrier density and potential is the Scharfetter-Gummel discretization scheme. Now the carrier and lattice temperatures are also introduced as the basic variables, we need to re-examine the original scheme. The discretization form for the carrier flux from node 1 to node 2 along the grid line linking these two nodes is as follows when the temperature effects are taken into consideration:

$$F_{n,1\rightarrow 2} = \frac{1}{d_{21}}[\mu_{n2}B(u_{n,21})n_2T_{n2} - \mu_{n1}B(-u_{n,21})n_1T_{n1}] \tag{5.20}$$

$$F_{p,1\rightarrow 2} = \frac{1}{d_{21}}[\mu_{p1}B(u_{p,21})p_1T_{p1} - \mu_{p2}B(-u_{p,21})p_2T_{p2}] \tag{5.21}$$

where $F_n = -j_n/q$ and $F_p = j_p/q$, and B is the Bernoulli function defined as

$$B(x) = \frac{x}{e^x - 1} \tag{5.22}$$

$u_{c,21}$ is the function of the potential difference and carrier temperature defined as follows:

$$u_{c,21} = \frac{\psi_2 - \psi_1}{(T_{c1} + T_{c2})/2} \tag{5.23}$$

where c stands for either n or p and d_{21} is the distance between nodes 1 and 2. The assumptions made in the above discretization scheme are:

1. Mobility is constant along the grid line between nodes 1 and 2 (called as interval) and its value is the average of the mobilities on two nodes.
2. $E(x)/T_c(x) \approx \text{const}$ over the interval where x represents the position within the interval. In particular,

$$\frac{E}{T_c} = \frac{\psi_1 - \psi_2}{d_{21}}\frac{2}{T_{c1} + T_{c2}} \tag{5.24}$$

3. Flux, F, is constant over the interval.

The discretization for energy flux is as follows:

$$\begin{aligned} \mathbf{s}_{n,1\rightarrow 2} = \frac{1}{d_{21}}\Big[&\mu_{n2}B(u_{n,21})n_2T_{n2}\left(\frac{\psi_1+\psi_2}{2} - c_eT_{n2}\right) - \\ &\mu_{n1}B(-u_{n,21})n_1T_{n1}\left(\frac{\psi_1+\psi_2}{2} - c_eT_{n1}\right) \end{aligned} \tag{5.25}$$

$$\boldsymbol{s}_{p,\,1\to2} = \frac{1}{d_{21}}\Big[\mu_{p1}B\,(u_{p,\,21})\,p_1T_{p1}\,(\frac{\psi_1+\psi_2}{2}-c_eT_{p1}) - \mu_{p2}B\,(-u_{p,\,21})\,p_2T_{p2}\,(\frac{\psi_1+\psi_2}{2}-c_eT_{p2})\tag{5.26}$$

where $c_e = 5/2 - \nu$ [1] and is set to a default value of 1.8 in the code. The form for the heat flux:

$$F_{heat,\,1\to2} = \frac{\bar{\kappa}}{d_{21}}\,(T_{L1}-T_{L2})\tag{5.27}$$

where the average κ over the interval is evaluated as follows:

$$\bar{\kappa}_L = \kappa_0\left(\frac{T_{L1}+T_{L2}}{2\times300}\right)^{-\alpha}\tag{5.28}$$

5.3 Normalization

This section briefly describes the normalization scheme used in PISCES-2ET. Internal variables representing basic variables: ψ (as well as ϕ's for quasi Fermi potentials), n, p, T_n, T_p, and T_L, are all normalized before they are used in the computation. Potentials are normalized by $V_{th} = k_BT_0/q$ where T_0 is the environment temperature which can be accessed through parameter `temperature` in the `model` card. All temperatures are normalized using the environment temperature and carrier concentrations are normalized using $c_{scl} = \sqrt{N_CN_V}$ where N_C and N_V are for the base semiconductor material[2]. The current density is normalized using the quantity of $-qc_{scl}V_{th}$. The reason for choosing this normalization factor is related to the use of Scharfetter-Gummel scheme and minus sign is incurred because the electron current density is used as the reference. Other quantities such as distance, time, electric field, mobility are not normalized.

1. For meaning of ν see Eq. (A.27) in APPENDIX A.

2. There is only one base material for a simulated region even though there might be several material systems in the device.

5.4 Newton Projection Scheme

Newton projection scheme is a method to project the next solution based on the current solution when the applied bias is changed. It has also application in low-frequency, small signal analysis. The principle of this scheme is rather simple and is described in this section. Again taking Eq. (5.1), but now we only consider the DC case. When bias is advanced, i.e.

$$\boldsymbol{p} = \boldsymbol{p} + \Delta\boldsymbol{p} \tag{5.29}$$

one can obtain the following approximate equation:

$$\boldsymbol{F}(\boldsymbol{x} + \Delta\boldsymbol{x}, \boldsymbol{p} + \Delta\boldsymbol{p}) \approx \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{p}) + \boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{p})\Delta\boldsymbol{x} + \boldsymbol{F}_{\boldsymbol{p}}(\boldsymbol{x}, \boldsymbol{p})\Delta\boldsymbol{p} = 0 \tag{5.30}$$

Because the solution at $\boldsymbol{p}$ has been found, i.e., $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{p}) = 0$, the increment of $\boldsymbol{x}$ due to $\boldsymbol{p}$ can be found by solving the following system of equations:

$$\boldsymbol{F}_{\boldsymbol{x}}\Delta\boldsymbol{x} = -\boldsymbol{F}_{\boldsymbol{p}}\Delta\boldsymbol{p} \tag{5.31}$$

This is a linear system of equations with the same coefficient matrix as for the DC solution and $\boldsymbol{F}_{\boldsymbol{p}}$ is easy to find because $\boldsymbol{F}$ has few equations which are related to $\boldsymbol{p}$. The above formulation can also be applied to the low-frequency small signal analysis because when $\Delta\boldsymbol{p} \to 0$, the ratio of $\Delta\boldsymbol{x}/\Delta\boldsymbol{p}$ can be found exactly by solving the above equation for $\Delta\boldsymbol{x}$. Foe more detailed discussion of this method, readers are referred to [41].

5.5 Global Data Structure in PISCES-2ET

This section describes how one can change the maximum number of grid points in the simulator. Since PISCES-2ET is coded using Fortran 77, memory usage is static and knowledge of various array sizes is necessary during the compilation time. All of the basic dimensions are defined in the common file `p2conf.h` and are written in a fully scalable way. The first part of `p2conf.h` reads as follows:

```
-------------------------p2conf.h-----------------------------
c   There are four parameters that affect the memory usage of
c   PISCES-2ET:
c   AVGNB  - average neighbors of each node point including
c            itself.  For a quad-based mesh with control level
```

```
c            equal to 1, 6 neighbors are a very conservative
c            estimation (can be higher for pure Delaunay mesh.)
c   MAXPDE - max. no. of equations solved (1 for Poisson only,
c            6 for DUET)
c   MAXPT  - max. no. of grid points in the simulated device
c   MAX1D  - bandwidth of the LU-factored Jacobian. Since
c            minimum degree (fill-in) is employed, the worst
c            case will be sqrt(MAXPT). If the device is highly
c            1-D oriented, MAX1D can be much smaller to improve
c            memory efficiency.
c
      integer AVGNB, MAXPDE, MAXPT, MAX1D
      parameter (AVGNB = 6, MAXPDE = 3)
      parameter (MAXPT = 3000, MAX1D = 55)
c     parameter (MAXPT = 6000, MAX1D = 78)
c     parameter (MAXPT = 9000, MAX1D = 95)
---------------------------------------------------------------
```

The total memory usage during the run time of PISCES-2ET is approximately:

```
16 * MAX1D * MAXPDE * MAXPDE * MAXPT (in bytes)
```

Note that since the full Newton iteration is the default method owing to its robustness, the memory space required is proportional to the square of the number of equations and to the power 1.5 of the number of points in the worst case (when `MAX1D` is equal to the square root of the number of points, i.e., the mesh points are evenly distributed in X and Y directions). The present setup (`MAXPT = 3000`) will take about 24M-byte memory space. Since most of the memory allocation is shared in a common block called “`tmpco`” (in various `.h` files), users can still run DUET model with the present setup at least to about 1,200 points. Most of the modern workstations have more than 24M-byte main memory, so the swapping during execution should be minimal. If the application cannot be bounded by this setup, users need only to change the above four parameters and remake (recompile) the PISCES-2ET executable. PISCES-2ET will automatically report any insufficient memory allocation during the first encounter (usually at the symbolic factorization). Users can then modify the reported dimensions and make further trade-off accordingly. If the application is always much smaller than the present setup, we do not suggest to recompile. If MAXPT becomes too small and the factored LU matrix in full Newton iteration is no longer dominant in the memory usage, users probably will have some complaints from the compiler about array with negative dimensions. This is because the common "`tmpco`" space is shared in many routines and the patch array `TMPPAD` is used to make every common declaration to have the same length to avoid compiler errors. If users have to do this (probably due to an old 8MB main memory machine) and the compiler indeed complains the negative dimension, users need to change the last line in `p2conf.h`, where the size of the common block "`tmpco`" is defined symbolically. Other constraints will have minimal impact on the memory usage. They are listed below for reference.

```
---------------------p2conf.h--------------------------
c
c   OTHER GLOBAL CONSTRAINTS ON DIMENSION
c
      integer MAXCON, MAXCNT, MAXNB, MAXPAR
      integer MAXINF, MAXREG, MAXREC, MAXPTORI, MXBDEP
c.. maximum number of contact nodes
      parameter (MAXCON = 500)
c.. maximum number of contacts
      parameter (MAXCNT = 10)
c.. maximum number of triangles a point can belong to
      parameter (MAXNB = 20)
c.. maximum number material parameters
      parameter (MAXPAR = 50)
c.. maximum number of interfaces
      parameter (MAXINF = 10)
c.. maximum number of regions
      parameter (MAXREG = 1000)
c.. maximum elements in 1-D for temporary tensor-product
c   grid info
      parameter (MAXREC = 1000)
c.. maximum depth of bias partition
      parameter (MXBDEP = 10)
c.. maximum original tensor-product mesh points before
c   elimination
      parameter (MAXPTORI = 10*MAXPT)
---------------------------------------------------------
```

The remaining dimension declaration is defined symbolically.

CHAPTER 6
Simulation Examples

In this chapter, four simulation examples will be presented to demonstrate the new capabilities available in PISCES-2ET. The first two are for the application of carrier temperature analysis. These are simulations for the substrate current in MOSFET and output characteristics of SOI structures. Both simulated and measured data are compared. The last two are for the simulation of heterostructures. Among them the first one shows the electrical simulation of a GaAs/AlGaAs-based LED (for light emitting diode) with cylindrical symmetry. One unique feature of this example is the existence of the floating layer in the structure, which poses numerical difficulty. The second example for heterostructure analysis is the simulation of an AlInAs/GaInAs-based MODFET in which the surface Fermi-level pinning is shown.

6.1 Substrate Current of MOS Structure

This examples is to show the effect of the carrier-energy dependent impact ionization model on the substrate current of MOSFET. It has been known that the drift-diffusion (DD) model gives bigger-than-expected substrate current even for relatively long channel MOSFETs [22]. In this example, two MOS structures with effective channel length 2 and 0.8 µm, respectively, are simulated and the one with 0.8µm channel length is compared to the measured data. Following the approach in [42], the results are plotted in a format of log (I_{sub}/I_d) vs. $1/(V_{ds}-V_{d,sat})$ in which the experimental data are insensitive to the variations in the channel length, gate oxide thickness, and V_{gs}.

It can be clearly seen that the DUET model provides more reasonable results compared to the experimental data [42], [43] than does DD model. It should be noticed that during the simulation, one fitting parameter, which effectively changes the energy relaxation time (or equivalently the energy relaxation length) used in the expression for energy-dependent ionization rate, is used. The similar calibration approach has been reported in [22] too. The input deck to PISCES-2ET is listed below for this device structure. Special attention should be paid to the use of `symbolic` and `model` cards. In the `symbolic` card parameter `engy.ele` indicates that during the simulation the electron energy balance equation should also be solved together with the Poisson's and carrier continuity equations. In the `model` card, three parameters are relevant to the energy-dependent impact ionization (II) model: `imp.tn` (but not with `imp.tp`) specifies that only electron energy dependent II model is used and for holes the impact ionization rate is still determined by the local field (a default). `imp.jt` indicates for energy-dependent II model, the current density rather than the saturation velocity is used to determine the II rate. Finally, `impjt.rat` is used to specify the ratio of L_w to $\tau_w v_{sat}$ (refer to Eq. (3.39)). The simulation results are shown in Figure 6.2.

```
title N-MOS for Substrate Current Simulation
$
$  A 0.8 micron N-MOSFET example, Electron-ET model
$
options plotdev=xterm x.scr=3.8 y.scr=3.8
$  mesh
$===========================
mesh rect nx=40 ny=28
x.m n=1   l=0     r=1
x.m n=5   l=0.5   r=0.6
x.m n=35  l=1.5   r=1.0
x.m n=40  l=2.0   r=1.7
y.m n=1   l=-0.0152  r=1
y.m n=3   l=0.0      r=1
y.m n=5   l=0.002    r=1
y.m n=9   l=0.010    r=1
y.m n=16  l=0.1  r=1
y.m n=20  l=0.2  r=1
y.m n=22  l=0.3  r=1
y.m n=23  l=0.4  r=1
y.m n=25  l=1.1  r=1
y.m n=28  l=2.0  r=1
$  region and electrodes
$===========================
region num=1 ix.l=1 ix.h=40 iy.l=1 iy.h=3 oxide
region num=2 ix.l=1 ix.h=40 iy.l=3 iy.h=28 silicon
```

```
$ Electrode: 1-Source, 2-Gate, 3-Drain, 4-Substrate
elec num=1 ix.l=1  ix.h=4  iy.l=3  iy.h=3
elec num=2 ix.l=5  ix.h=35 iy.l=1  iy.h=1
elec num=3 ix.l=36 ix.h=40 iy.l=3  iy.h=3
elec num=4 ix.l=1  ix.h=40 iy.l=28 iy.h=28
$   doping
$============================
dop uniform p.type conc=2e17 reg=2
dop gauss n.type conc=5e19 char=0.08 ra=0.7 x.r=0.52  y.t=0 y.b=0.10 dir=y
dop gauss n.type conc=5e19 char=0.08 ra=0.7 x.l=1.48  y.t=0 y.b=0.10 dir=y

$   contact
$============================
contact all neutral
contact num=2 workfunction=4.17
$   initial solution
$============================
symbolic newton carrier=0
method itlim=50 p.tol=1e-5 c.tol=1e-5
model srh fldmob conmob
solve ini
$  ramping vg and vd
$============================
symbolic newton carrier=2 engy.ele
model srh fldmob conmob imp.tn imp.jt impjt.rat=0.69
method itlim=20 p.tol=1e-5 c.tol=1e-5 trap
log   ivfile=08_Tn.iv
solve v2=0.2 vstep=0.2 nstep=3 electr=2 proj
solve v2=1.0 outfile=08_Tn.g1 proj
solve v3=0.1 vstep=0.1 nstep=4  electr=3 proj
solve v3=0.6 vstep=0.2 nstep=12 electr=3 proj
solve v3=3.0 outfile=08_Tn.g3 proj
end
```

Figure 6.1 Input file for simulation of the substrate current for MOSFETs.

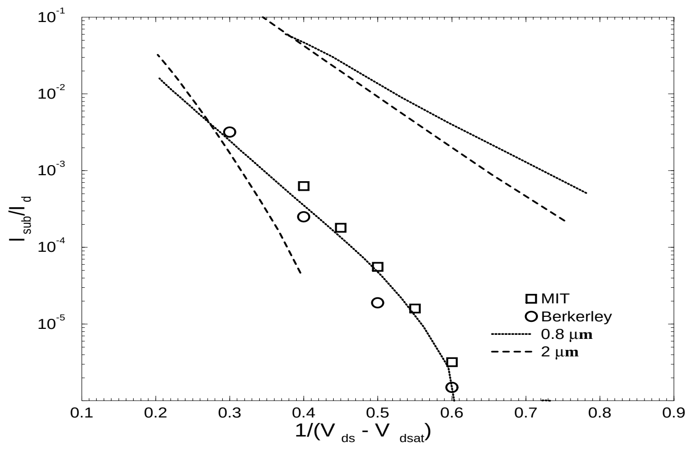


Figure 6.2 Simulation results for the substrate current in MOSFET with two different channel length (2 and 0.8 µm) and the comparison is made for 0.8 µm case between the ET-simulated and measured results [42], [43]. The upper curves are simulated using DD model while the lower curves are obtained

6.2 SOI Simulation

This example shows simulation for SOI structures with two different channel length: 0.47 and 0.12 µm [44]. Input deck for the simulation of SOI with 0.12µm channel length is as follows:

```
title SOI BERKELEY
$
$  Effective channel length =.12um
$  0.12um of silicon on 0.4um oxide substrate
$
$*************** define the rectangular grid ***********$
$
mesh     outf=s/msh12r width=9.5 rect nx=43 ny=35
$
x.m      n=1  l=0.0  r=1.00
x.m      n=7  l=0.5  r=0.9
X.m      n=12 l=0.6  r=.9
x.m      n=22 l=0.74 r=1.15
x.m      n=32 l=0.88 r=0.85
```

```
x.m      n=37 l=0.98 r=1.1
x.m      n=43 l=1.48 r=1.1
$
y.m      n=1  l=0.0  r=1.00
y.m      n=6  l=0.4  r=0.80
y.m      n=21 l=0.46 r=1.15
y.m      n=33 l=0.52 r=0.80
y.m      n=35 l=0.54 r=1.00
$
region   num=1 ix.l=1 ix.h=43 iy.l=1  iy.h=6  oxide
region   num=2 ix.l=1 ix.h=43 iy.l=6  iy.h=33 silicon
region   num=3 ix.l=1 ix.h=43 iy.l=33 iy.h=35 oxide
$
$*********** define the electrodes ************
$ #1-GATE #2-SOURCE #3-DRAIN #4-SUBSTRATE (below oxide)
$
electrod num=1 ix.l=12 ix.h=32 iy.l=35 iy.h=35
electrod num=2 ix.l=1  ix.h=12 iy.l=33 iy.h=33
electrod num=3 ix.l=32 ix.h=43 iy.l=33 iy.h=33
electrod num=4 ix.l=1  ix.h=43 iy.l=1  iy.h=1
$
$*********** define the doping concentrations *****
$
doping   uniform conc=6e16 p.type reg=2
doping   gauss conc=2e20 n.type characteristic=0.1 start=0.46
+        x.right=0.6 ratio.lateral=0.25 region=2
doping   gauss conc=2e20 n.type characteristic=0.1 start=0.46
+        x.left=0.88 ratio.lateral=0.25 region=2
$
interfac qf=20.e10
contact  num=1 n.poly
plot.1d  dop log abs a.x=0 a.y=0.4 b.x=1.48 b.y=0.4 min=14 max=21
+        outf=s/dd/dop.12r ascii
plot.1d  dop log abs a.x=0 a.y=0.52 b.x=1.48 b.y=0.52 min=14 max=21
$
models   conmob intelmob print
mobility ec.crit=8e4 char.int=0.04
material region=2 vsaturation=1e7
symb     newton carriers=0
$
method   xnorm itlimit=30
$
solve    ini
symb     newton carriers=1 engy.ele
```

```
method   trap itlimit=30
$
solve    v1=-3 v4=-15 v3=2 vstep=0.5 nstep=10 electr=4
solve    v1=-3 v4=64 v3=0.05
end
```

The partial input deck for SOI with 0.4µm channel length is listed below. Only structural part is included.

```
title SOI BERKELEY
$
$ Effective Channel length =.47um
$ 0.47um of silicon on 0.4um oxide substrate
$
mesh     width=9.5 rect nx=43 ny=35
$
x.m      n=1  l=0.00  r=1.00
x.m      n=7  l=0.5   r=0.9
X.m      n=12 l=0.6   r=.9
x.m      n=22 l=0.915 r=1.15
x.m      n=32 l=1.23  r=0.85
x.m      n=37 l=1.33  r=1.1
x.m      n=43 l=1.83  r=1.1
$
y.m      n=1  l=0.0   r=1.0
y.m      n=6  l=0.4   r=0.8
y.m      n=21 l=0.46  r=1.15
y.m      n=33 l=0.52  r=0.8
y.m      n=35 l=0.54  r=1.0
$
region   num=1 ix.l=1 ix.h=43 iy.l=1  iy.h=6  oxide
region   num=2 ix.l=1 ix.h=43 iy.l=6  iy.h=33 silicon
region   num=3 ix.l=1 ix.h=43 iy.l=33 iy.h=35 oxide
$
$*********** define the electrodes ************
$ #1-GATE #2-SOURCE #3-DRAIN #4-SUBSTRATE(below oxide)
$
electrod num=1 ix.l=12 ix.h=32 iy.l=35 iy.h=35
electrod num=2 ix.l=1  ix.h=12 iy.l=33 iy.h=33
electrod num=3 ix.l=32 ix.h=43 iy.l=33 iy.h=33
electrod num=4 ix.l=1  ix.h=43 iy.l=1  iy.h=1
$
$*********** define the doping concentrations *****
$
```

```
doping   uniform conc=6e16 p.type    reg=2
doping   gauss conc=2e20 n.type characteristic=0.1 start=0.46
+        x.right=0.6 ratio.lateral=0.25 region=2
doping   gauss conc=2e20 n.type characteristic=0.1 start=0.46
+        x.left=1.23 ratio.lateral=0.25 region=2
$
interfac qf=20e10
contact  num=1 n.poly
  ...
end
```

Figure 6.3 Input file for simulation of SOI structure.

The simulation results are compared with the measured data provided by Hu's group at UC Berkeley [44] in Figure 6.4. The agreement between ET model and experiment is good for devices with both channel lengths. While DD model gives good fit to the measured data for device with 0.47µ channel length, it grossly underestimates the drain current for device with 0.12µ, an indication that DD models fails to predict the velocity overshoot phenomenon.

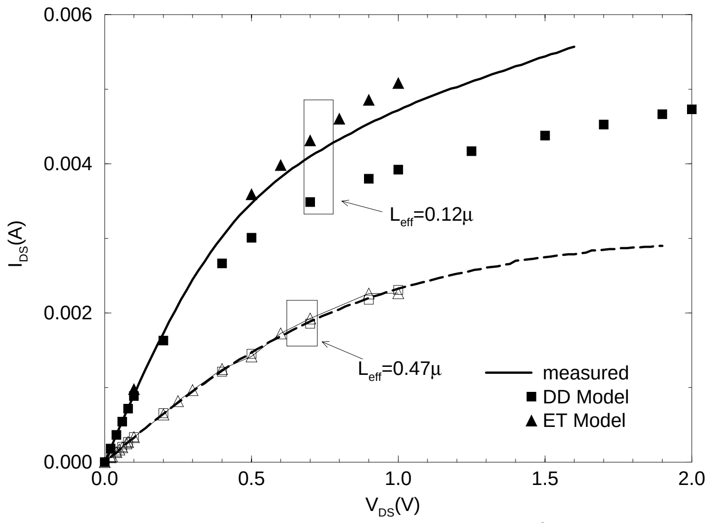


Figure 6.4 Simulated and measured data [44] for SOI structure with different channel length.

6.3 Cylindrically Symmetric LED

There are two unique features with the simulation of this light emitting diode (LED). The first is the cylindrical structure which requires the use of the cylindrical coordinate. PISCES-2ET has the capability of simulating cylindrically symmetric structure by specifying `cylin` in the `mesh` card. The second feature is the numerical difficulty caused by the existence of the current-blocking, floating *n*-layer in the *p*-region. To ensure the convergence of Newton iterations, a contact is put to the exterior perimeter of this floating layer and the zero-current boundary condition is applied. This technique guarantees the *n*-layer is indeed "floating" as far as the potential of the layer is concerned. The effect of the extra contact on the device characteristics is minor because the blocking layer is doped heavily. Following are two input files for the simulation. In order to put the contact to the floating layer from the exterior, the structure is shifted left in the lateral direction such that the rotation axis falls at $x = 0$. For the initial solution, the voltage boundary condition is used first and then the zero-current boundary condition is switched on. It is observed that before the diode is turned on, i.e., for the bias below 1V, the numerical (discretization) noise may exceed the magnitude of the simulated current and so the current conservation law (the sum of the signed terminal currents should be zero) is not well observed. Because the device behavior before turning on is not a primary concern, bigger bias steps are used and especially the diode bias is increased directly from 0.7 to 1V. Also note that the *n*-layer is on the top of the structure, so the negative cathode bias is incremented. Figure 6.5 lists the first input file for the simulation for obtaining the initial solution with the voltage boundary condition.

```
title Initial solution for LED starting from 0-carr
$
options plotdev=xterm x.scr=3.8 y.scr=3.8
$
$ ***** Define Rectangular Mesh *****
$
mesh rectangular nx=17 ny=29 cylin
$
x.mesh n=1  l=-62.0  r=1.0
x.mesh n=3  l=-37.0  r=1.0
x.mesh n=5  l=-30.0  r=1.0
x.mesh n=7  l=-13.0  r=1.0
x.mesh n=9  l=-12.0  r=1.0
x.mesh n=11 l=-11.0  r=1.0
x.mesh n=13 l=-10.0  r=1.0
x.mesh n=15 l=-9.0   r=1.0
```

```
x.mesh n=17 l=00.0   r=1.0
$
y.mesh n=1  l=0.000 r=1.0
y.mesh n=5  l=1.000 r=1.0
y.mesh n=9  l=3.600 r=1.0
y.mesh n=13 l=4.600 r=1.0
y.mesh n=17 l=5.600 r=1.0
y.mesh n=21 l=7.600 r=1.0
y.mesh n=25 l=8.900 r=1.0
y.mesh n=29 l=40.00 r=1.0
$
$ ***** Regions *****
$
region num=1 ix.l=1  ix.h=17 iy.l=1 iy.h=29 oxide
region num=2 ix.l=1  ix.h=5  iy.l=1 iy.h=5  algaas xmole=0.00
region num=3 ix.l=1  ix.h=15 iy.l=5 iy.h=9  algaas xmole=0.28
region num=3 ix.l=13 ix.h=17 iy.l=9 iy.h=13 algaas xmole=0.28
$
$ ***** Active Regions *****
$
region num=4 ix.l=1  ix.h=13 iy.l=9  iy.h=13 algaas xmole=0.08
region num=4 ix.l=11 ix.h=17 iy.l=13 iy.h=17 algaas xmole=0.08
region num=5 ix.l=1  ix.h=11 iy.l=13 iy.h=17 algaas xmole=0.28
region num=5 ix.l=9  ix.h=17 iy.l=17 iy.h=21 algaas xmole=0.28
region num=6 ix.l=1  ix.h=9  iy.l=17 iy.h=21 algaas xmole=0.00
region num=6 ix.l=7  ix.h=17 iy.l=21 iy.h=25 algaas xmole=0.00
region num=7 ix.l=1  ix.h=7  iy.l=21 iy.h=25 algaas xmole=0.00
region num=8 ix.l=1  ix.h=17 iy.l=25 iy.h=29 algaas xmole=0.00
$
$ ***** Electrodes *****
$
elec num=1 ix.l=1 ix.h=3  iy.l=1  iy.h=1
elec num=2 ix.l=1 ix.h=17 iy.l=29 iy.h=29
elec num=3 ix.l=1 ix.h=1  iy.l=22 iy.h=24
$
$ ***** Add Doping *****
$
dop unif conc=3e18 n.type x.l=-62.0 x.r=-30.0 y.t=0.0 y.b=1.0
dop unif conc=2e18 n.type x.l=-62.0 x.r=-9.0  y.t=1.0 y.b=3.6
dop unif conc=2e18 n.type x.l=-10.0 x.r=0.0   y.t=3.6 y.b=4.6
dop unif conc=3e18 n.type x.l=-62.0 x.r=-10.0 y.t=3.6 y.b=4.6
dop unif conc=3e18 n.type x.l=-11.0 x.r=0.0   y.t=4.6 y.b=5.6
dop unif conc=1e18 p.type x.l=-62.0 x.r=-11.0 y.t=4.6 y.b=5.6
dop unif conc=1e18 p.type x.l=-12.0 x.r=0.0   y.t=5.6 y.b=7.6
dop unif conc=2e18 p.type x.l=-62.0 x.r=-12.0 y.t=5.6 y.b=7.6
dop unif conc=2e18 p.type x.l=-13.0 x.r=0.0   y.t=7.6 y.b=8.9
dop unif conc=2e18 n.type x.l=-62.0 x.r=-13.0 y.t=7.6 y.b=8.9
dop unif conc=2e19 p.type x.l=-62.0 x.r=0.0   y.t=8.9 y.b=40.0
$
$ ****** Plots *****
$
plot.2d boundary pause
plot.2d grid pause
$
$ ***** Contacts *****
$
contact all neutral
symbol carrier=0 newton
method itlim=50 p.tol=5.e-5 c.tol=5.e-5
models temp=300 fldmob auger conmob rad consrh
$
solve ini
symbol carrier=2 newton
solve outfil=led.v0
$
end
```

Figure 6.5 First input deck for the simulation of LED, used to find the initial solution for voltage boundary condition.

The plots resulted from the running of the above file are shown in Figure 6.6 and Figure 6.7. Figure 6.6 shows the structure of the device and Figure 6.7 shows the mesh used in the simulation.

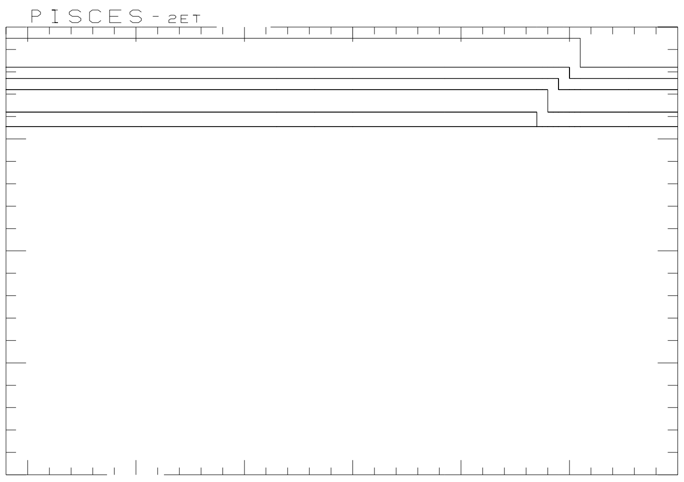


Figure 6.6 Device structure in terms of layers in LED.

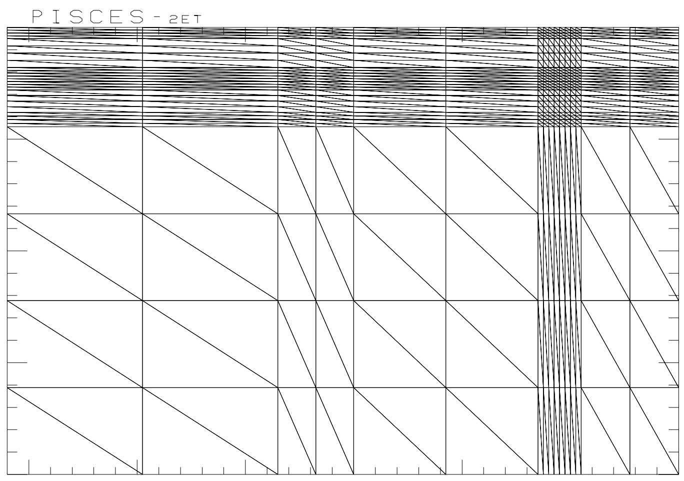


Figure 6.7 Mesh used in the LED simulation.

Once the initial solution is found with the voltage boundary condition, the current boundary condition is switched on for the contact to the floating layer in order to ensure it being “floating” (no current drawn at this terminal). This is implemented in the second file as listed in Figure 6.8.

```
title LED Simulation Using 0-carr Solution
$
options plotdev=xterm x.scr=3.8 y.scr=3.8
$
$ ***** Define Rectangular Mesh *****
$
mesh rectangular nx=17 ny=29 cylin
$
x.mesh n=1  l=-62.0  r=1.0
x.mesh n=3  l=-37.0  r=1.0
x.mesh n=5  l=-30.0  r=1.0
x.mesh n=7  l=-13.0  r=1.0
x.mesh n=9  l=-12.0  r=1.0
x.mesh n=11 l=-11.0  r=1.0
x.mesh n=13 l=-10.0  r=1.0
x.mesh n=15 l=-9.0   r=1.0
x.mesh n=17 l=00.0   r=1.0
$
y.mesh n=1  l=0.000 r=1.0
y.mesh n=5  l=1.000 r=1.0
y.mesh n=9  l=3.600 r=1.0
y.mesh n=13 l=4.600 r=1.0
y.mesh n=17 l=5.600 r=1.0
y.mesh n=21 l=7.600 r=1.0
y.mesh n=25 l=8.900 r=1.0
y.mesh n=29 l=40.00 r=1.0
$
$ ***** Regions *****
$
region num=1 ix.l=1  ix.h=17 iy.l=1 iy.h=29 oxide
region num=2 ix.l=1  ix.h=5  iy.l=1 iy.h=5  algaas xmole=0.00
region num=3 ix.l=1  ix.h=15 iy.l=5 iy.h=9  algaas xmole=0.28
region num=3 ix.l=13 ix.h=17 iy.l=9 iy.h=13 algaas xmole=0.28
$
$ ***** Active Regions *****
$
region num=4 ix.l=1  ix.h=13 iy.l=9  iy.h=13 algaas xmole=0.08
region num=4 ix.l=11 ix.h=17 iy.l=13 iy.h=17 algaas xmole=0.08
region num=5 ix.l=1  ix.h=11 iy.l=13 iy.h=17 algaas xmole=0.28
region num=5 ix.l=9  ix.h=17 iy.l=17 iy.h=21 algaas xmole=0.28
region num=6 ix.l=1  ix.h=9  iy.l=17 iy.h=21 algaas xmole=0.00
region num=6 ix.l=7  ix.h=17 iy.l=21 iy.h=25 algaas xmole=0.00
region num=7 ix.l=1  ix.h=7  iy.l=21 iy.h=25 algaas xmole=0.00
region num=8 ix.l=1  ix.h=17 iy.l=25 iy.h=29 algaas xmole=0.00
```

```
$ ***** Electrodes *****
$
elec num=1 ix.l=1 ix.h=3  iy.l=1  iy.h=1
elec num=2 ix.l=1 ix.h=17 iy.l=29 iy.h=29
elec num=3 ix.l=1 ix.h=1  iy.l=22 iy.h=24
$
$ ***** Add Doping *****
$
dop unif conc=3e18 n.type x.l=-62.0 x.r=-30.0 y.t=0.0 y.b=1.0
dop unif conc=2e18 n.type x.l=-62.0 x.r=-9.0  y.t=1.0 y.b=3.6
dop unif conc=2e18 n.type x.l=-10.0 x.r=0.0   y.t=3.6 y.b=4.6
dop unif conc=3e18 n.type x.l=-62.0 x.r=-10.0 y.t=3.6 y.b=4.6
dop unif conc=3e18 n.type x.l=-11.0 x.r=0.0   y.t=4.6 y.b=5.6
dop unif conc=1e18 p.type x.l=-62.0 x.r=-11.0 y.t=4.6 y.b=5.6
dop unif conc=1e18 p.type x.l=-12.0 x.r=0.0   y.t=5.6 y.b=7.6
dop unif conc=2e18 p.type x.l=-62.0 x.r=-12.0 y.t=5.6 y.b=7.6
dop unif conc=2e18 p.type x.l=-13.0 x.r=0.0   y.t=7.6 y.b=8.9
dop unif conc=2e18 n.type x.l=-62.0 x.r=-13.0 y.t=7.6 y.b=8.9
dop unif conc=2e19 p.type x.l=-62.0 x.r=0.0   y.t=8.9 y.b=40.0
$
$ ** Current boundary condition is now switched on for contact 3
$
contact all neutral
contact num=3 curr
symbol carrier=2 newton
method itlim=50 p.tol=5.e-5 c.tol=5.e-5 trap
models temp=300 fldmob auger conmob rad consrh
$
load infil=led.v0
log ivfil=led.iv
solve i3=0 outfil=led.i0
solve v1=-0.1 vstep=-0.1 nstep=5 elect=1
solve v1=-0.62 vstep=-0.02 nstep=3 elect=1
solve v1=-0.7
solve v1=-1.0 vstep=-0.1 nstep=10 elect=1 proj
$
plot.1d x.a=v1 y.a=i1 abs log
end
```

Figure 6.8 Second input file for LED simulation. The initial solution is obtained from the voltage boundary condition and is switched to the current boundary condition for the floating layer contact. The bias step jumps from -0.7 to $-1\,\text{V}$ to avoid the numerical instability.

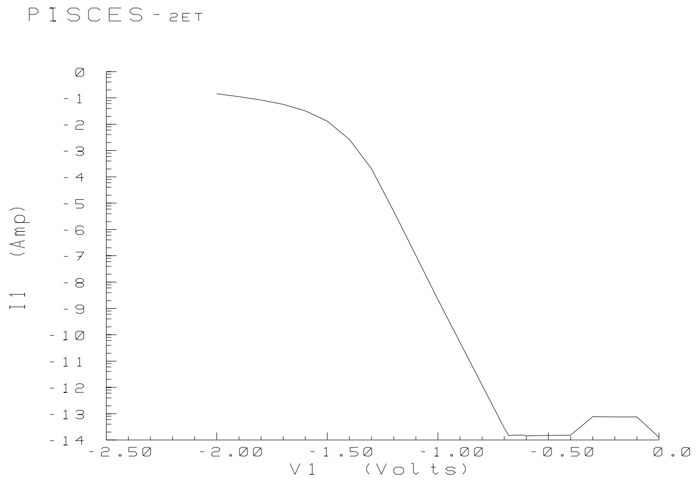


Figure 6.9 Simulated $I - V$ characteristics of LED. Notice that the bias is applied to the top of the structure (*n*-layer), so it is negative and the current computed below 0.7V is due to the numerical noise.

The numerical convergence behavior during simulation is excellent. The simulated *I-V* characteristics is shown in Figure 6.9. The bias ramp is applied to the cathode of the diode, so the magnitude of the bias increase is in the opposite direction to the *x*-axis.

6.4 AlInAs/GaInAs MODFET

This *n*-channel MODFET is constructed in a rectangular region with source, drain, and channel regions all formed in a narrow-bandgap (0.72eV) $Ga_{0.47}In_{0.53}As$ material. The doping in the channel region is light (5×10^{15} cm^{-3}) and the channel conduction is through the spill-over of electrons from a heavily-doped (4×10^{18} cm^{-3}), adjacent wide-bandgap (1.45eV) $Al_{0.48}In_{0.52}As$ region. Because in PISCES-2ET ternary compound $Ga_xIn_{1-x}As$ is represented by setting $y = 1$ in quaternary $Ga_xIn_{1-x}As_yP_{1-y}$, in the input deck which follows “`GaInAsP`” is used instead of “`GaInAs`”. The gate contact is formed in a recessed manner. But in order to maintain the planarity of the structure, an oxide is filled in the recessed region. The structure is described in the following input file (Figure 6.10) and different regions in the device are plotted in Figure 6.11.

```
title AlInAs/GaInAs based MODFET structure
$
options plotdev=xterm x.screen=3.8 y.screen=3.8
$
$ Rectangular mesh
$
mesh nx=55 ny=30 rect
$
x.mesh   n=1   l=0.0    r=1.0
x.mesh   n=4   l=1.0    r=0.8
x.mesh   n=9   l=1.625  r=0.9
x.mesh   n=20  l=1.925  r=1.0
x.mesh   n=32  l=2.075  r=1.0
x.mesh   n=46  l=2.375  r=1.0
x.mesh   n=52  l=3.0    r=1.1
x.mesh   n=55  l=4.0    r=1.2
$
y.mesh   n=1   l=0.0    r=1.0
y.mesh   n=4   l=0.03   r=1.0
y.mesh   n=12  l=0.06   r=0.9
y.mesh   n=14  l=0.066  r=1.0
y.mesh   n=15  l=0.069  r=1.0
y.mesh   n=25  l=0.099  r=1.1
y.mesh   n=30  l=0.2    r=1.5
$
$ Region and material specifications
$
region num=1 ix.l=9  ix.h=46 iy.l=1  iy.h=4  insulator
region num=2 ix.l=1  ix.h=9  iy.l=1  iy.h=4  gainasp   xmole=0.47 ymole=1.0
region num=3 ix.l=46 ix.h=55 iy.l=1  iy.h=4  gainasp   xmole=0.47 ymole=1.0
region num=4 ix.l=1  ix.h=55 iy.l=4  iy.h=12 alinas    xmole=0.48
region num=5 ix.l=1  ix.h=55 iy.l=12 iy.h=14 alinas    xmole=0.48
region num=6 ix.l=1  ix.h=55 iy.l=14 iy.h=15 alinas    xmole=0.48
region num=7 ix.l=1  ix.h=55 iy.l=15 iy.h=25 gainasp   xmole=0.47 ymole=1.0
region num=8 ix.l=1  ix.h=55 iy.l=25 iy.h=30 alinas    xmole=0.48

$
$ Electrodes: 1 source, 2 gate, and 3 drain
$
elec num=1 ix.l=1  ix.h=4  iy.l=1 iy.h=1
elec num=2 ix.l=20 ix.h=32 iy.l=4 iy.h=4
elec num=3 ix.l=52 ix.h=55 iy.l=1 iy.h=1
$
$ Doping specification
```

```
$
doping region=2 unif conc=2e19 n.type
doping region=3 unif conc=2e19 n.type
doping region=4 unif conc=1e15 n.type
doping region=5 unif conc=6e18 n.type
doping region=6 unif conc=1e15 n.type
doping region=7 unif conc=1e15 n.type
doping region=8 unif conc=1e15 n.type

$
$ Background doping
$
doping unif x.l=0.0 x.r=4.0 y.t=0.0 y.b=0.2 conc=5e14 p.type
$
$ Interface trapping for the pinning of Fermi-level
$
deepimp unif x.l=1.625 x.r=1.925 y.t=0.03 y.b=0.031 accep conc=1.0e19
+       eion=0.7
deepimp unif x.l=2.075 x.r=2.375 y.t=0.03 y.b=0.031 accep conc=1.0e19
+       eion=0.7
$
$ Show regions
$
plot.2d bound pause
$
$ Doping and x-mole profiles under the gate
$
plot.1d dop  x.s=2.0 x.e=2.0 y.s=0.0 y.e=0.2 log pause
plot.1d xmol x.s=2.0 x.e=2.0 y.s=0.0 y.e=0.2 min=0.45 max=0.49 pause
$
contact num=2 workf=4.8 surf.rec
models  temp=300 conmob consrh fldmob auger incompl
$
$ Solving Poisson's equation only for the equilibrium solution
$
symb newton carriers=0
method itlimit=50 biaspart
$
solve ini
$
$ Switch to full set of semiconductor equations
$
symb newton carr=2
$
```

```
$ Re-solve at the equilibrium and save solution
$
solve outfil=modfet.ini
log   ivfil=modfet.iv
$
$ Band diagram under the spacer between the gate and drain regions
$
plot.1d band.c neg x.s=1.8 x.e=1.8 y.s=0.0 y.e=0.2 min=-1.5 max=0.5
plot.1d band.v neg x.s=1.8 x.e=1.8 y.s=0.0 y.e=0.2 unch
plot.1d qfn    neg x.s=1.8 x.e=1.8 y.s=0.0 y.e=0.2 unch line=3 pause
$
$ Apply the negative gate voltage and sweep Vd upto 5.0
solve v2=-0.8 v3=0.0 vstep=0.2 nstep=4 elec=3 proj
solve v3=1.3 proj
solve v3=1.6 proj
solve v3=2.0 vstep=1.0 nstep=3 elec=3 proj
$
plot.1d x.a=v3 y.a=i3
$
end
```

Figure 6.10 Input file for the simulation of AlInAs/GaInAs MODFET. The surface Fermi-level pinning is modeled by specifying the high density of deep level impurities at the free surface.

The simulation results are shown in the following figures. Figure 6.12 shows the band diagram at the thermal equilibrium for the Fermi-level pinning due to the high density of the surface traps and the situation without surface pinning. Finally the simulated drain current vs. the drain voltage at $V_{GS} = -0.8\,\mathrm{V}$ is shown in Figure 6.13.

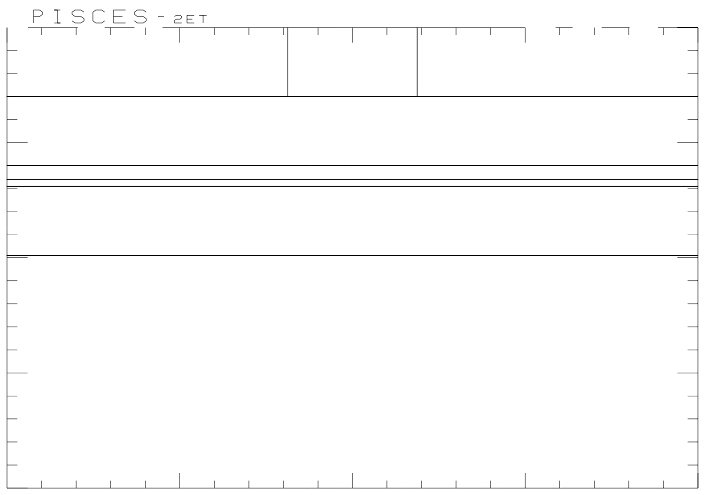


Figure 6.11 Regions in AlInAs/GaInAs MODFET structure.

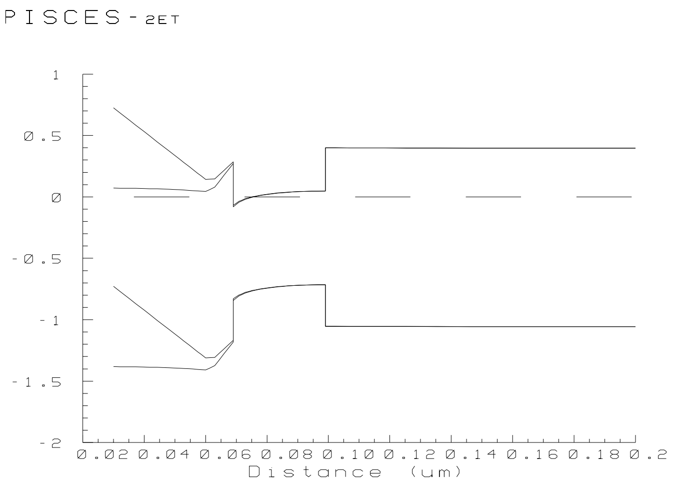


Figure 6.12 Band diagram along the line segment located at the spacer between the source and gate in the direction perpendicular to the surface of the MODFET. The effect of Fermi-level pinning due to the surface traps is shown by comparison to the free surface.

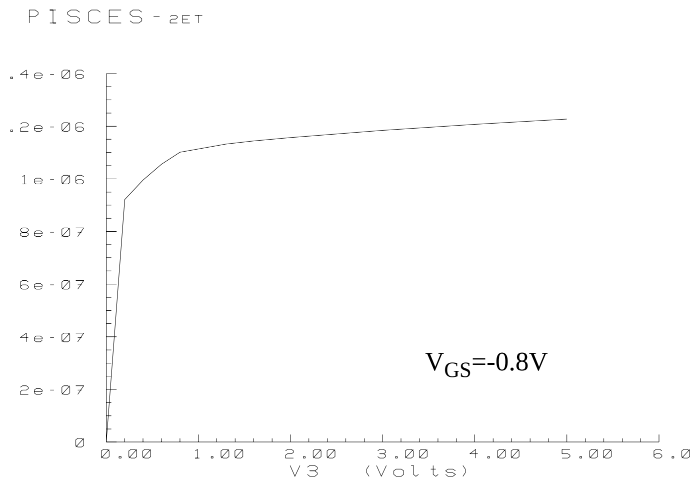


Figure 6.13 Simulated output $I - V$ characteristics at $V_{GS} = -0.8\,\text{V}$.

APPENDIX A Fermi-Dirac Distribution and Heterostructures

A.1 Effect of Fermi-Dirac Statistics

Now we consider a more general case in DUET model, i.e., the quasi-thermal equilibrium distribution function, f_0, used to form the complete distribution function in Eq. (2.1) is a Fermi-Dirac distribution. The Fermi-Dirac distribution function has the form of

$$f_0(\boldsymbol{r}, \boldsymbol{k}) = \frac{2}{1 + \exp\left(\frac{E(\boldsymbol{k}) - E_{Fn}(\boldsymbol{r})}{k_B T_n(\boldsymbol{r})}\right)} \tag{A.1}$$

We need to transform the function to have the dependence on n, T_n, and ε (the kinetic energy). This can be done in the following procedure. First with Eq. (A.1), one can show that from the definition of n (the carrier density) that

$$n = N_C(T_n, T_L) F_{1/2}\left(\frac{E_{Fn}(\boldsymbol{r}) - E_C(\boldsymbol{r})}{k_B T_n(\boldsymbol{r})}\right) \tag{A.2}$$

where N_C has the same expression as in section 2.1 and $F_{1/2}$ is the Fermi integral of order one half. The definition of Fermi integral as used in this manual will be given in APPENDIX B together with some other related properties. To have more concise notation, we define two symbols:

$$\eta_n = \frac{E_{Fn} - E_C}{k_B T_n} \tag{A.3}$$

which is a function of n and T_n as can be seen from Eq. (A.2), and

$$\gamma_n(\eta_n) = \frac{F_{1/2}(\eta_n)}{\exp(\eta_n)} \tag{A.4}$$

Note that γ_n is actually a measure of the carrier degeneracy, i.e., it is always smaller than unity but approaches the unity for non-degenerate case ($\eta_n \to -\infty$). There are similar expressions for holes, but η_p is expressed as

$$\eta_p = \frac{E_V - E_{Fp}}{k_B T_p} \tag{A.5}$$

We can now express the separation of the quasi-Fermi level and band edge using the carrier concentration as follows:

$$\exp\left(\frac{E_{Fn} - E_C}{k_B T_n}\right) = \frac{n}{N_C(T_n)\,\gamma_n(n, T_n)} \tag{A.6}$$

Note that throughout this document, we use n and T_n as the fundamental variables. Substituting this expression in Eq. (A.1), we obtain an equivalent form of the Fermi-Dirac distribution as follows:

$$f_0(n, T_n, \varepsilon) = \frac{2}{1 + \dfrac{N_C(T_n)\,\gamma_n(n, T_n)}{n}\exp\left(\dfrac{\varepsilon}{k_B T_n}\right)} \tag{A.7}$$

The complete distribution function can then be constructed as

$$f(\boldsymbol{r}, \boldsymbol{k}) = f_0(n(\boldsymbol{r}), T_n(\boldsymbol{r}), \varepsilon(\boldsymbol{k})) - \tau(\boldsymbol{r}, \boldsymbol{k})\left(\frac{h}{2\pi m_n^*}\boldsymbol{k}\cdot\nabla f_0 - q\frac{h}{2\pi m_n^*}\frac{\partial f_0}{\partial\varepsilon}\boldsymbol{k}\cdot\boldsymbol{E}_n\right) \tag{A.8}$$

We will further make several assumptions to simplify the above expression. First we assume the effective mass is constant, i.e. for parabolic band structure. Then the carrier velocity, $\boldsymbol{v}$, is related to the wave vector, $\boldsymbol{k}$, by the following expression:,

$$\boldsymbol{v} = \frac{h}{2\pi m^*}\boldsymbol{k} \tag{A.9}$$

and the kinetic energy can be written as

$$\varepsilon = \frac{m^* v^2}{2} = \frac{h^2}{2(2\pi)^2 m^*}k^2 \tag{A.10}$$

Furthermore, in heterostructures the electric field might be different for electrons and holes depending on the gradient of the respective band edge. For electrons, we have

$$\boldsymbol{E}_n = \frac{1}{q}\nabla E_C \tag{A.11}$$

We assume the relaxation time ($\boldsymbol{k}$-dependent τ) can be simplified to depend on the kinetic energy only, i.e., $\tau(\boldsymbol{r}, \boldsymbol{k}) = \tau(\boldsymbol{r}, \varepsilon)$. Eq. (A.8) can then be written as

$$f(\boldsymbol{r}, \boldsymbol{v}) = f_0(n, T_n, \varepsilon) - \tau(\boldsymbol{r}, \varepsilon)\left(\boldsymbol{v}\cdot\nabla f_0 - q\frac{\partial f_0}{\partial\varepsilon}\boldsymbol{v}\cdot\boldsymbol{E}_n\right) \tag{A.12}$$

Now we are ready to evaluate the carrier current and its relevant coefficients. By doing so, one may also find the exact definition for some commonly used physical parameters such as the mobility and diffusivity. The carrier current can be evaluated from the distribution function as follows:

$$\boldsymbol{j}_n = -q\int\boldsymbol{v}f(\boldsymbol{r}, \boldsymbol{k})\frac{1}{(2\pi)^3}dk_x dk_y dk_z = -q\int\boldsymbol{v}f(\boldsymbol{r}, \boldsymbol{v})\left(\frac{m_n^*}{h}\right)^3 dv_x dv_y dv_z = -q\int\boldsymbol{v}f(\boldsymbol{r}, \boldsymbol{v})\,d^3v \tag{A.13}$$

where we have used symbol $d^3v = (m^*/h)^3 dv_x dv_y dv_z$ and the introduction of $(2\pi)^{-3}$ in the integral in $\boldsymbol{k}$-space is due to the quantum mechanics consideration. It would be desirable to perform the above integral over ε if the integrand is the function of $\boldsymbol{r}$ and ε only. This can be done through the following transformation:

$$\left(\frac{m_n^*}{h}\right)^3\int dv_x dv_y dv_z = \frac{2\pi(2m_n^*)^{3/2}}{h^3}\int\varepsilon^{1/2}d\varepsilon = \frac{1}{\sqrt{\pi}}(k_B T_n)^{-3/2}N_C\int\varepsilon^{1/2}d\varepsilon \tag{A.14}$$

It should be noticed that the first part of the right-hand-side (RHS) in Eq. (A.12) is even in $\boldsymbol{k}$-space and the second part is odd in $\boldsymbol{k}$-space, thus in the integral of Eq. (A.13) the first part of the distribution function will vanish (by multiplying $\boldsymbol{v}$ the even part becomes odd), and only the second (odd) part is used in the integration. The result is as follows:

$$\boldsymbol{j}_n = q\int\tau(\boldsymbol{r},\varepsilon)\,\boldsymbol{vv}\cdot\nabla f_0 d^3v - q^2\left(\int\tau(\boldsymbol{r},\varepsilon)\,\frac{\partial f_0}{\partial\varepsilon}\boldsymbol{vv}d^3v\right)\cdot\boldsymbol{E}_n \tag{A.15}$$

Note that $\boldsymbol{vv}$ should be understood as the tensor. It is easy to identify the second term on the RHS represents the drift component because it is proportional to the electric field. The first term is, however, not clear at this stage and we need to further find out the form of ∇f_0. We now proceed to find its expression.

$$\nabla f_0 = \frac{\partial f_0}{\partial n}\nabla n + \frac{\partial f_0}{\partial T_n}\nabla T_n = k_B T_n\frac{\partial f_0}{\partial\varepsilon}\left(-\frac{\lambda_n}{n}\nabla n + \left(\frac{3}{2}\frac{\lambda_n}{T_n} - \frac{\varepsilon}{k_B T_n^2}\right)\nabla T_n\right) \tag{A.16}$$

Thus Eq. (A.15) becomes

$$\begin{aligned}\boldsymbol{j}_n = {} & q\left[-k_B T_n\lambda_n\frac{1}{n}\int d^3v\tau(\boldsymbol{r},\varepsilon)\,\frac{\partial f_0}{\partial\varepsilon}\boldsymbol{vv}\right]\cdot\nabla n + qn\left[-q\frac{1}{n}\int d^3v\tau(\boldsymbol{r},\varepsilon)\,\frac{\partial f_0}{\partial\varepsilon}\boldsymbol{vv}\right]\cdot\boldsymbol{E}_n \\ & + qn\left[-\frac{3}{2}\frac{1}{T_n}\left[-k_B T_n\lambda_n\frac{1}{n}\int d^3v\tau(\boldsymbol{r},\varepsilon)\,\frac{\partial f_0}{\partial\varepsilon}\boldsymbol{vv}\right] - \frac{1}{T_n}\frac{1}{n}\int d^3v\tau(\boldsymbol{r},\varepsilon)\,\varepsilon\frac{\partial f_0}{\partial\varepsilon}\boldsymbol{vv}\right]\cdot\nabla T_n\end{aligned} \tag{A.17}$$

This seemly formidably complex expression can greatly be simplified by introducing some familiar coefficients through identification of origin of each term.

A.2 General Relationship between Mobility, Diffusivity, and Thermal Diffusivity

Furthermore, the tensor representation can be reduced to scalar under the constant effective mass assumption we have made. That is,

$$\int d^3v\boldsymbol{vv} = \left[\frac{8\pi}{3h^3}\sqrt{2m_n^*}\int\varepsilon^{3/2}d\varepsilon\right]\hat{\boldsymbol{I}} \tag{A.18}$$

where $\hat{\boldsymbol{I}}$ is the identity matrix. By realizing the relationship of $\hat{\boldsymbol{I}}\cdot\nabla = \nabla$ and by comparing to the standard drift-diffusion expression of the carrier current, one can write the following compact form:

$$\boldsymbol{j}_n = qD_n\nabla n + qn\mu_n\boldsymbol{E}_n + qnD_n^T\nabla T_n \tag{A.19}$$

where coefficients D, μ, and D^T are named as diffusion constant (or diffusivity), mobility, and thermal diffusivity, respectively. Comparing Eq. (A.19) with Eq. (A.17), one obtains

$$\mu_n = -q\frac{1}{n}\frac{8\pi}{3h^3}\sqrt{2m_n^*}\int\tau(\boldsymbol{r},\varepsilon)\frac{\partial f_0}{\partial\varepsilon}\varepsilon^{3/2}d\varepsilon \tag{A.20}$$

$$D_n = \frac{k_BT_n}{q}\lambda_n\mu_n \tag{A.21}$$

$$D_n^T = -\frac{1}{T_n}\left[\frac{3}{2}D_n + \frac{8\pi}{3h^3}\sqrt{2m_n^*}\frac{1}{n}\int\tau(\boldsymbol{r},\varepsilon)\frac{\partial f_0}{\partial\varepsilon}\varepsilon^{5/2}d\varepsilon\right] \tag{A.22}$$

where $\hat{\boldsymbol{I}}$ is the identity matrix. It becomes immediately clear from Eq. (A.20) and Eq. (A.21) that we have arrived a generalized Einstein relationship between the mobility, μ, and diffusivity, D as follows:

$$\frac{D}{\mu} = \frac{k_BT}{q}\lambda = \frac{k_BT}{q}\frac{F_{1/2}(\eta)}{F_{-1/2}(\eta)} \tag{A.23}$$

where all T, λ, and η are for the same carriers as in D and μ. The relationship of D^T and μ is not so obvious at the first glance. But for the special case of Boltzmann statistics, i.e. for

$$f_0(n, T_n, \varepsilon) = \frac{n}{N_C}\exp\left(-\frac{\varepsilon}{k_BT_n}\right) \tag{A.24}$$

there exists a special relationship that

$$\frac{\partial f_0}{\partial T_n} = n\frac{\partial}{\partial T_n}\frac{\partial f_0}{\partial n} \tag{A.25}$$

Then we obtain

$$D_n^T = \frac{\partial D_n}{\partial T_n} \tag{A.26}$$

Note that this simple relationship does not hold for general Fermi-Dirac statistics.

Now we consider the specific form of the relaxation time, $\tau(\boldsymbol{r}, \varepsilon)$. For the current PISCES-2ET implementation, we assume the following power dependence:

$$\tau(\boldsymbol{r}, \varepsilon) = \alpha(\boldsymbol{r}, T_L)\,\varepsilon^{-\nu} \tag{A.27}$$

where $\nu \geq 0$ and α is the remaining part of τ which does not depend on the carrier kinetic energy. To be more specific, we have explicitly expressed the lattice temperature (T_L) dependence. Also note that α doesn't have the units of time.

With the above power dependence of the relaxation time on energy, we can find more direct link between D^T and μ. Notice that

$$\int_0^\infty \varepsilon^{-\nu}\frac{\partial f_0}{\partial \varepsilon}\varepsilon^{3/2}d\varepsilon = \int_0^\infty \varepsilon^{3/2-\nu}df_0 = (3-2\nu)\,\Gamma\left(\frac{3}{2}-\nu\right)F_{1/2-\nu}(\eta_n)\,(k_B T_n)^{3/2-\nu} \tag{A.28}$$

where integral by parts has been used during the derivation and ν must be no bigger than 3/2 to insure that no singularity occurs. For the same reasoning, one obtains

$$\int_0^\infty \varepsilon^{-\nu}\frac{\partial f_0}{\partial \varepsilon}\varepsilon^{5/2}d\varepsilon = (5-2\nu)\,\Gamma\left(\frac{5}{2}-\nu\right)F_{3/2-\nu}(\eta_n)\,(k_B T_n)^{5/2-\nu} \tag{A.29}$$

Thus it is clear that

$$\frac{D_n^T}{\mu_n} = -\frac{1}{T_n}\left[\frac{3}{2}\frac{k_B T_n}{q}\lambda_n - \left(\frac{5}{2}-\nu\right)\frac{k_B T_n}{q}\frac{F_{3/2-\nu}}{F_{1/2-\nu}}\right] \tag{A.30}$$

or

$$D_n^T = \frac{k}{q}\left[\left(\frac{5}{2}-\nu\right)\frac{F_{3/2-\nu}}{F_{1/2-\nu}} - \frac{3}{2}\lambda_n\right]\mu_n \tag{A.31}$$

We now consider a more specific case, that is for the case where the acoustic phonon scattering is dominant, then $p = 1$ according to Stratton [1]. Furthermore, for nondegenerate cases, using property Eq. (B.5), the ratio of Fermi integrals with different orders all become the unity, hence D_n^T even becomes zero. That is to say, the gradient of the carrier temperature doesn't contribute to the current flow.

A.3 Heterostructures

For heterostructures, the effective mass would be a function of the position also. In evaluating ∇f_0, therefore, one needs to take into consideration the variation of the effective mass. This can be done by adding an extra term in Eq. (A.16) as follows:

$$\nabla f_0 = \frac{\partial f_0}{\partial n}\nabla n + \frac{\partial f_0}{\partial T_n}\nabla T_n + \frac{\partial f_0}{\partial m_n^*}\nabla m_n^* = k_B T_n \frac{\partial f_0}{\partial \varepsilon}\left[-\frac{\lambda_n}{n}\nabla n + \left(\frac{3}{2}\frac{\lambda_n}{T_n} - \frac{\varepsilon}{k_B T_n^2}\right)\nabla T_n + \frac{3}{2}\frac{\lambda_n}{m_n^*}\nabla m_n^*\right] \quad \text{(A.32)}$$

The last term on RHS constitutes another driving force the carrier transport, but the difference from the previous two driving forces due to ∇n and ∇T_n is that the effective mass is a structure parameter and does not change during the solution process, i.e., it is not a basic variable. We will later rearrange the terms in such a way that only true driving forces appear in the current expression. Before we proceed further, we look for other structure parameters. As discussed previously, for heterostructures the electric field which is a direct consequence of the gradient of potential might be different for electrons and holes. Similar to Eq. (A.11), the electric field for holes can be express as the gradient of the valence band edge:

$$\boldsymbol{E}_p = \frac{1}{q}\nabla E_V \quad \text{(A.33)}$$

But for the entire semiconductor region, there is one variable for electrostatic potential, ψ. So it is desirable to link both E_C and E_V to ψ. We define for heterostructure the electrostatic potential as

$$\psi = -\frac{E_{vac}}{q} \quad \text{(A.34)}$$

where E_{vac} is the energy level for vacuum. The advantage of defining the potential this way is to warrant it is continuous and differentiable for the electric field has to be finite in magnitude. This is extremely important as in the heterostructure, the conventional definition of the potential as the intrinsic Fermi level

$$E_{Fi} = \frac{E_C + E_V}{2} - kT_L \ln\frac{N_C(T_L)}{N_V(T_L)} \quad \text{(A.35)}$$

where T_L is the lattice temperature and the densities of states for conduction and valence bands are evaluated at the lattice temperature, is not necessarily continuous. We can then relate the band edges to the potential using band structure parameters: electron affinity χ and bandgap E_g as follows:

$$E_C = -q\psi - \chi \tag{A.36}$$

$$E_V = -q\psi - \chi - E_g \tag{A.37}$$

We are now ready to rewrite the current expressions for electrons and holes. Based on Eq. (A.32) and Eq. (A.15), one obtains for electrons:

$$\begin{aligned} \boldsymbol{j}_n &= qD_n\nabla n + qn\mu_n\left(\boldsymbol{F}_n - \frac{3}{2}\lambda_n\frac{k_BT_n}{q}\nabla\ln m_n^*\right) + qnD_n^T\nabla T_n \\ &= qD_n\nabla n + qnD_n^T\nabla T_n - qn\mu_n\nabla\psi - n\mu_n\left(\nabla\chi + \frac{3}{2}\lambda_n k_B T_n\nabla\ln m_n^*\right) \end{aligned} \tag{A.38}$$

For holes,

$$\boldsymbol{j}_p = -qD_p\nabla p - qpD_p^T\nabla T_p - qp\mu_p\nabla\psi - p\mu_p\left[\nabla(\chi + E_g) - \frac{3}{2}\lambda_p k_B T_p\nabla\ln m_p^*\right] \tag{A.39}$$

The above expressions can greatly be simplified if the quasi-Fermi level instead of the carrier concentration is used as the driving force for the carrier transport. By introducing the kinetic energy $\varepsilon(\boldsymbol{k}) = E(\boldsymbol{k}) - E_C(\boldsymbol{r})$ in Eq. (A.2), the Fermi-Dirac distribution function can be written as

$$f_0(\boldsymbol{r},\boldsymbol{k}) = \frac{2}{1 + \exp\left(\dfrac{E(\boldsymbol{k}) - E_{Fn}(\boldsymbol{r})}{k_BT_n(\boldsymbol{r})}\right)} = \frac{2}{1 + \exp\left(\dfrac{\varepsilon(\boldsymbol{k}) + E_C(\boldsymbol{r}) - E_{Fn}(\boldsymbol{r})}{k_BT_n(\boldsymbol{r})}\right)} \tag{A.40}$$

Thus

$$\nabla f_0(\boldsymbol{r},\boldsymbol{k}) = \frac{\partial f_0}{\partial\varepsilon}\left(\nabla E_C - \nabla E_{Fn} - \frac{\varepsilon(\boldsymbol{k}) + E_C(\boldsymbol{r}) - E_{Fn}(\boldsymbol{r})}{T_n}\nabla T_n\right) \tag{A.41}$$

Substituting the above expression into Eq. (A.15) and using Eq. (A.11), one obtains

$$\boldsymbol{j}_n = n\mu_n\nabla E_{Fn} + qnQ_n\nabla T_n \tag{A.42}$$

where the thermopower Q is defined as

$$Q_n = D_n^T + \frac{k_B}{q}\left(\frac{3}{2}\lambda_n - \eta_n\right)\mu_n \tag{A.43}$$

Note that the coefficient for the current component explicitly due to the carrier temperature gradient in Eq. (A.42) is different from that in Eq. (A.19). But for the isothermal case ($T_n = \text{const}$), both expressions are reduced to the standard drift-diffusion (DD) form.

APPENDIX B
Mathematical Properties of Fermi Integral

We first define the Fermi integral of order ν as used in this document:

$$F_\nu(\eta) = \frac{1}{\Gamma(\nu+1)} \int \frac{x^\nu}{1+e^{x-\eta}} dx \tag{B.1}$$

where Γ is the Gamma function which is defined as

$$\Gamma(\nu) = \int_0^\infty x^{\nu-1} e^{-x} dx \tag{B.2}$$

and has the properties of

$$\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi} \tag{B.3}$$

and

$$\Gamma(\nu+1) = \nu\Gamma(\nu) \qquad \text{for } \nu > 0 \tag{B.4}$$

There are a few useful properties of Fermi integral, which are used in the program. We list two of them below:

$$\lim_{\eta \to -\infty} F_{\nu}(\eta) = e^{\eta} \qquad \text{regardless value of } \nu \tag{B.5}$$

and

$$\frac{d}{d\eta} F_{\nu}(\eta) = F_{\nu-1}(\eta) \tag{B.6}$$

From the above properties, one can derive two useful derivatives:

$$\frac{\partial}{\partial n} \eta_n(n, T_n) = \frac{\lambda_n}{n} \tag{B.7}$$

where λ_n is the symbol of $\lambda(\eta_n)$ with λ defined as

$$\lambda = \frac{F_{1/2}(\eta)}{F_{-1/2}(\eta)} \tag{B.8}$$

Another useful derivative is

$$\frac{\partial}{\partial n} \gamma_n(n, T_n) = \frac{\gamma_n}{n}(1 - \lambda_n) \tag{B.9}$$

For holes there are completely analogous formulas and are listed below:

$$\frac{\partial}{\partial p} \eta_p(p, T_p) = \frac{\lambda_p}{p} \tag{B.10}$$

$$\frac{\partial}{\partial p} \gamma_p(p, T_p) = \frac{\gamma_p}{p}(1 - \lambda_p) \tag{B.11}$$

Now we give the expressions for the derivative of η and γ w.r.t. the carrier temperature T.

$$\frac{\partial}{\partial T_n} \eta_n(n, T_n) = -\frac{3}{2} \frac{1}{T_n} \lambda_n \tag{B.12}$$

$$\frac{\partial}{\partial T_p} \eta_p(p, T_p) = -\frac{3}{2} \frac{1}{T_p} \lambda_p \tag{B.13}$$

$$\frac{\partial}{\partial T_n}\gamma_n(n, T_n) = -\frac{3}{2}(1-\lambda_n)\frac{\gamma_n}{T_n} \tag{B.14}$$

$$\frac{\partial}{\partial T_p}\gamma_p(p, T_p) = -\frac{3}{2}(1-\lambda_p)\frac{\gamma_p}{T_p} \tag{B.15}$$

APPENDIX C
Formulations in Previous PISCES-II Versions

Considering the limited access to the manuals and reports for the previous versions of PISCES II (i.e. PISCES-II User's Manual (1984) [3] and PISCES-IIB Supplementary report (1985) [4]), we list in this Appendix those equations and expressions referred to in the User's Manual but not discussed in this report. Users may find the similar ones in the previous PISCES II reports.

Surface recombination velocities for electrons and holes, v_{sn} and v_{sp}, are given by

$$\begin{aligned} v_{sn} &= \frac{A_n^{**}T^2}{qN_C} \\ v_{sp} &= \frac{A_p^{**}T^2}{qN_V} \end{aligned} \tag{C.1}$$

where A_n^{**} and A_p^{**} are the effective Richardson constants (p. 389 in [18]) for electrons and holes.

Incomplete ionization for shallow doping impurities:

$$\begin{aligned} N_D^+ &= \frac{N_D}{1 + g_D e^{(E_{Fn} - E_D)/kT}} \\ N_A^- &= \frac{N_A}{1 + g_A e^{(E_A - E_{Fp})/kT}} \end{aligned} \tag{C.2}$$

where E_D and E_A are impurity energies, g_A and g_D are degeneracy factors for donors and acceptors, respectively.

Surface mobility reduction factor, g_{surf}. To model the low-field mobility along the oxide-semiconductor interface in a simple way, the following formula is used:

$$\mu_{0,\,surface} = g_{surf}\mu_{0,\,bulk} \tag{C.3}$$

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

Field-dependent barrier lowering model for Schottky contacts:

$$\Delta\phi_b = \sqrt{\frac{q}{4\pi\varepsilon_s}}E^{1/2} + \alpha E \tag{C.4}$$

where E is the magnitude of the electric field at the interface. Both lowering mechanisms, image force (the first term on RHS of Eq. (C.4)) and static dipole layer (second term), are considered in the above model.

References

[1] R. Stratton, “Semiconductor current-flow equations (diffusion and degeneracy),” *IEEE Trans. Electron Devices,* vol. ED-19, pp. 1288-1292, Dec. 1972.

[2] E. C. Kan, D. Chen, U. Ravaioli, Z. Yu, and R. W. Dutton, “Formulation of macroscopic transport models for numerical simulation of semiconductor devices,” to be published in *VLSI Design,* Aug. 1994.

[3] M. R. Pinto, C. S. Rafferty and R. W. Dutton, “PISCES-II - Poisson and Continuity Equation Solver," Stanford Electronics Laboratory Technical Report, Stanford University, September 1984.

[4] M. R. Pinto, C. S. Rafferty, H. R. Yeager, and R. W. Dutton, “PISCES-II - Supplementary report," Technical Report, Integrated Circuits Laboratory, Stanford University, 1985.

[5] N. D. Arora, J. R. Hauser, and D. J. Roulston, “Electron and hole mobilities in silicon as a function of concentration and temperature,” *IEEE Trans. Ele. Dev.,* Vol. ED-29, pp. 292-295, Feb. 1982.

[6] Z. Yu *et al.*, “Development of doping and temperature dependent mobility model in GaAs using a robust optimizer,” *Private communication,* July 1993.

[7] J. M. Dorkel and Ph. Leturcq, “Carrier mobility in silicon semi-empirically related to temperature, doping and injection level,” *Solid-State Elec.,* Vol. 24, no. 9, pp. 821-825, 1981.

[8] C. Lombardi, S. Manzini, A. Saporito, and M. Vanzi, “A physically based mobility model for numerical simulation of non-planar devices,” *IEEE CAD*, Vol. 7, no. 11, pp. 1164-1171, Nov. 1988.

[9] J. T. Watt, *Modeling The Performance of Liquid-Nitrogen Cooled CMOS VLSI,* Ph.D. dissertation, Stanford University, May 1989.

[10] A. G. Sabnis and J. T. Clemens, “Characterization of the electron mobility in the inverted <100> Si surface,” *IEDM Technical Digest,* pp. 18-21, 1979.

[11] H. Shin, A. F. Tasch, Jr., C. M. Maziar, and S. K. Banerjee, “A new approach to verify and derive a transverse field-dependent mobility model for electrons in MOS inversion layers,” *IEEE Trans. Elec. Dev.,* Vol. 36, no. 6, pp. 1117-1124, June 1989.

[12] S. Schwarz and S. Russek, “Semi-empirical equations for electron velocity in silicon: Part II--MOS inversion layer,” IEEE Trans. Electron Devices, vol. ED-30, pp. 1634-1639, Dec. 1983.

[13] H. Shin, G. M. Yeric, A. F. Tasch and C. M. Maziar, “Physically-based models for effective mobility and local-field mobility of electrons in MOS inversion layers,” *Solid State Elec.*, Vol. 34, no. 6, pp. 545-552, 1991.

[14] D. M. Caughey and R. E. Thomas, “Carrier mobilities in silicon empirically related to doping and field,” Proc. IEEE, pp. 2192-2193, Dec. 1967.

[15] J. J. Barnes, R. J. Lomax, and G. I. Haddad, “Finite-element simulation of GaAs MESFET’s with lateral doping profiles and submicron gates,” *IEEE Trans. Electron Devices,* Vol. ED-23, Sept. 1976.

[16] H. R. Yeager, “Circuit-simulation models for the high electron-mobility transistor,” Ph. D. Thesis, Stanford University, April 1989.

[17] W. Haensch and M. Miura-Mattausch, “The hot-electron problem in small semiconductor devices,” *J. Appl. Phys.,* Vol. 60, p. 650, 1986.

[18] S. M. Sze, *Physics of Semiconductor Devices,* 1st ed., p. 62, John Wiley & Sons, New York, 1969.

[19] K. Hui, C. Hu, P. George, and P. K. Ko, “Impact ionization in GaAs MESFET’s”, *IEEE Ele. Dev. Lett.*, vol. 11, no. 3, p. 113, Feb. 1991.

[20] D. Ritter, R. A. Hamm, A. Feygenson, and M. B. Panish, “Anomalous electric field and temperature dependence of collector multiplication in InP/$Ga_{0.47}In_{0.53}As$ heterojunction bipolar transistors,” *Appl. Phys. Lett.* **60** (25), p. 3150, 22 June 1992.

[21] I. Watanabe, T. Torikai, K. Makita, K. Fukushima, and T. Uji, “Impact ionization rates in (100) $Al_{0.48}In_{0.52}As$,” *IEEE Ele. Dev. Lett.*, vol. 11, no. 10, p. 437, Oct. 1990.

[22] J. W. Slotboom, G. Streutker, M. J. v. Dort, P. H. Woerlee, A. Pruijmboom, and D. G. Gravestejn, “Non-local impact ionization in silicon devices,” IEDM Technical Digest, p. 127, 1991.

[23] PISCES-II code, version 1985.

[24] S. M. Sze, *Physics of Semiconductor Devices,* 2nd ed., John Wiley & Sons, New York, 1981.

[25] R. S. Muller and T. I. Kamins, *Device Electronics for Integrated Circuits,* John Wiley & Sons, New York, 1977.

[26] Sadao Adachi, “Material parameters of $In_{1-x}Ga_xAs_yP_{1-y}$ and related binaries,” *J. Appl. Phys.* **53** (12), Dec. 1982, pp. 8775-8792.

[27] S. M. Sze, ed. *High-Speed Semiconductor Devices,* John Wiley & Sons, New York, 1990.

[28] *Landolt-Börnstein,* New York: Springer-Verlag, Vol. 17a, 1982.

[29] M. Neuberger, *III-V Semiconducting Compounds,* Handbook of Electronic Materials Vol. 2, IFI/PLENUM: New York, 1971.

[30] D. V. Lang, *et al.*, "Measurement of the band gap of Ge_xSi_{1-x}/Si strained-layer heterostructures," *Appl. Phys. Lett.* 47 (12), pp.1333-1335, Dec. 1985.

[31] M P C M Krijn, "Heterojunction band offsets and effective masses in III-V quaternary alloys," *Semicond. Sci. Technol.* **6** pp. 27-31, 1991.

[32] S. W. Corzine, *et al.*, "Optical gain in III-V bulk and quantum well semiconductors," in *Quantum Well Lasers,* ed. P. S. Zory, Jr., Academic Press, 1993.

[33] Z. Yu and R. W. Dutton, "SEDAN III – A generalized electronic material device analysis program," *Tech. Rep.* Stanford Uni., 1985.

[34] R. People, K. W. Wecht, K. Alavi, and A. Y. Cho, "Measurement of the conduction-band discontinuity of molecular beam epitaxial grown $In_{0.52}Al_{0.48}As/In_{0.53}Ga_{0.47}As$, *N-n* heterojunction by *C-V* profiling," *Appl. Phys. lett.* **43** (1), pp. 118-120, July 1983.

[35] S. W. Corzine, R. H. Yan, and L. A. Coldren, "Theoretical gain in strained InGaAs/AlGaAs quantum wells including valence-band mixing effects," *Appl. Phys. Lett.* 57 (26), pp. 2835-2837, 24 Dec. 1990.

[36] S. E. Laux, "Techniques for small-signal analysis of semiconductor devices," *IEEE Trans. Electron Devices,* Vol. ED-32, No. 10, pp. 2028-2037, 1985.

[37] Z.-Y. Wang, K.-C. Wu, and R. W. Dutton, "An approach to construct pre-conditioning matrices for block iteration of linear equations," *IEEE Trans. CAD,* Vol. 11, No. 11, pp. 1334-1343, 1992.

[38] R. W. Freund, RIACS Tech. Rep. 91.18, NASA Ames Res. Ctr., Sept. 1991.

[39] R. W. Freund and N. M. Nachtigal, RIACS Tech. Rep. 90.51, NASA Ames Res. Ctr., Dec. 1990.

[40] K. Wu, Z. Yu, L. So, R.W. Dutton, and J. Sato-Iwanaga, "Robust and Efficient AC Analysis of High-speed Devices," *Digest IEDM '92,* pp. 935-938, San Francisco, Dec. 1992.

[41] Z. Yu, R.W. Dutton, and M. Vanzi, “An extension to Newton method in device simulators -- on an efficient algorithm to evaluate small signal parameters and to predict initial guess,” *IEEE Trans. CAD*, Vol. CAD-6, no. 1, pp.41-45, Jan. 1987.

[42] T. Y. Chan, P. K. Ko, and C. Hu, “A simple method to characterize substrate current in MOSFET’s,” *IEEE Electron Device Lett.,* Vol. 5, no. 12, pp. 505-507, Dec. 1984.

[43] G. G. Shahidi, D. A. Antoniadis, and H. I. Smith, “Reduction of channel hot-electron-generated substrate current in sub-150-nm channel length Si MOSFET’s,” *IEEE Electron Device Lett.*, Vol. 9, no. 10, pp. 497-499, Oct. 1988.

[44] Chenming Hu, *Private communication*, 1993.

[45] C. R. Crowell and S. M. Sze, “Temperature dependence of avalanche multiplication in semiconductors,” Applied Physics Letters, **9**, pp. 242-244, 1966.

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

[47] R. Bank, W. M. Coughran, W. Fichtner, E. H. Grosse, D. J. Rose, and R. K. Smith, “Transient simulation of silicon devices and circuits,” *IEEE Trans. Elect. Dev.,* pp. 1992-2007, Oct. 1985.

User’s Manual

CARD FORMAT

PISCES-2ET takes its input from a user provided ASCII file. Each line is a particular statement (or card), identified by the first word on the card (i.e., card name). 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. Card and 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 (string). Numerical parameters are assigned values by following the name of the parameter with an equal sign (=) and the value. Character parameters are assigned values by following the name of the parameter by an equal sign (=) and a string of characters. The first blank or tab delimits the string. The presence of a logical flag, which is represented by a character string, indicates TRUE while a logical flag preceded by a caret (^) indicates FALSE.

In the card descriptions which follow, the letters required to identify a parameter (i.e. a word) are printed in upper case, and the remainder of the word in lower case. For each card, parameters are first presented in a SYNTAX section and are then described in a PARAMETER section.

CARD SEQUENCE

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

- The MESH card must precede all other cards, except TITLE, COMMENT, and OPTIONS cards.
- When defining a rectangular mesh, the order of specification is as follows:

  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 cards must follow directly after the mesh definition.
- Before a solution, a symbolic factorization is necessary. Unless solving for the equilibrium condition (initial solution), a previous solution must also be loaded to provide an initial guess.
- Any CONTACT cards must precede the SYMBOLIC card.
- 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 (ψ, n, p, ϕ_n, ϕ_p, T_n, T_p, T_L, current (density), recombination rate, etc.) must be preceded somewhere in the input deck by a LOAD or SOLVE card to provide those quantities.

The execution of the program is line based, i.e. whenever a line is read in it is processed immediately. This feature allows users to run PISCES interactively.

In the following, cards are described in an alphabetic order.

CONVENTIONS

Multiple choices of parameters are denoted by putting the parameters in the braces and separated them by vertical bars. Thus

{ **par1 | par2** }

states that either **par1** or **par2** may be specified but not both.

All file names are currently set to no more than 20 characters in length.

Table M.1 lists the abbreviation of symbols for units in this manual.

Table M.1 Abbreviation for units used in this manual.

Abbr.	Units
K	Kelvin
J	Joule
s	second
μ	micrometer

Table M.1 Abbreviation for units used in this manual.

A	ampere
W	watt
C	coulomb

COMMAND CHECK

The **CHECK** card compares a specified solution (input) 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. Information for both solution and mesh is needed for check operation. All input files are assumed binary.

SYNTAX

CHeck `<file specification>`

`file specification:`

Infile = <filename> **Mesh** = <filename>
Samemesh = <logical>

PARAMETERS

Infile

A character string specifying the name of the solution file to compare. (Default: null)

Meshfile

A character string as the name of the file containing the mesh for the solution specified by **Infile**. (Default: null)

Samemesh

A logical flag indicating that the solution in **Infile** used the same mesh as the current solution. (Default: false)

EXAMPLES

```
  CHECK INFILE=file.sol MESH=file.msh
```

Compare solution in “file.sol” obtained using mesh “file.msh” against the current solution.

COMMAND COMMENT

The **COMMENT** card allows comments to be placed in the `PISCES-2ET` input file. The program ignores the information on the **COMMENT** card.

SYNTAX

COMment (or **$**) `<character strings>`

PARAMETERS

None.

EXAMPLES

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

COMMAND CONTACT

The **CONTACT** card defines the physical parameters of an electrode. If no **CONTACT** card is supplied for an electrode, it is assumed to be ohmic (charge-neutral and at the thermal equilibrium). Lumped elements, e.g., electrical (or thermal) resistor and capacitor, attached to the electrode are also specified here. See Section 2.3 for details.

SYNTAX

CONTAct `<number> <workfunction>`
+ `<special conditions>`

`number:`

{ **NUmber** = <integer> **|** **ALL** = <logical> }

`workfunction:`

{ **NEutral** = <logical> **|** **ALUminum** = <logical> **|**
P.polysilicon = <logical> **|** **N.polysilicon** =<logical> **|**
MOLybdenum = <logical> **|** **TUNgsten** = <logical> **|**
MO.disilicide = <logical> **|** **TU.disilicide** = <logical> **|**
Workfunction = <real> }

`special conditions:`

Surf.rec = <logical> **VSURFN** = <real>
VSURFP = <real> **BArrierlow** =<logical>
{ **ALPha** = <real> **|** **CUrrent** = <real> **|**
Resistance = <real> **|** **CApacitance** = <real> **|**
COn.resist = <real> **|** **TH.resist** = <real> }

PARAMETERS

ALL

A logical flag to indicate the same properties defined in this card are to be applied to all electrodes. (Default: false)

ALPha

A real number for the linear, dipole barrier lowering coefficient (α in Eq. (C.4)) in units of cm. (Default: 0.0)

ALUminum

A logical flag for using the work function of aluminum (4.17 eV). (Default: false)

Barrierlow

A logical flag to turn on the barrier lowering mechanism (Eq. (C.4)). (Default: false)

CApacitance

A real number for the capacitance per unit width (in *z*-direction) of a lumped capacitor to be attached to the contact, in units of F μ^{-1}. (Default: 0.0)

COn.resist

A real number for the distributed contact resistance, ρ_c, in units of Ω cm^2. The actual resistance associated with a certain area, *A*, of the contact is computed as $R = \rho_c / A$. (Default: 0.0)

CUrrent

A logical flag to impose the current boundary condition. (Default: false)

MO.disilicide

A logical flag for using the work function of molybdenum silicide (4.80 eV). (Default: false)

MOLybdenum

A logical flag for using the work function of molybdenum (4.53 eV). (Default: false)

NEutral

A logical flag to indicate that the work function is to be calculated from the doping level in the semiconductor at the contact assuming charge neutrality and thermal equilibrium. Note that neutral stands for charge neutral. (Default: true)

N.polysilicon

A logical flag for using the work function of n^+ silicon material, which is assumed to be the electron affinity for silicon (4.17 eV). (Default: false)

NUmber

An integer for the electrode sequence number which must previously be defined in the **ELEctrode** card. (No default)

Resistance

A real number for the resistance per unit width (in the *z*-direction) of a lumped resistor to be attached to the contact, in units of $\Omega\ \mu$. (Default: 0.0)

Surf.rec

A logical flag to indicate that finite surface recombination velocities are to be used at the respective contact. (Default: false)

P.polysilicon

A logical flag for using the work function of p^+ silicon material, which is assumed to be the sum of the electron affinity and bandgap of silicon (4.17 + E_{gap} eV). (Default: false)

TH.resist

A real number for the thermal resistance, as defined in Eq. (2.28), per unit width (in z-direction) attached to the contact, in units of K s J^{-1} μ. (Default: 0.0)

TU.disilicide

A logical flag for using the work function of tungsten silicide (4.80 eV). (Default: false)

TUNgsten

A logical flag for using the work function of tungsten (4.63 eV). (Default: false)

VSURFN, VSURFP

Real number parameters for electron and hole surface recombination velocities, respectively, in units of cm/s. (Defaults: as calculated according to Eq. (C.1))

Workfunction

A real number for the work function to be used, in units of eV. (Default: using flag **NEutral**)

If none of the special conditions above is specified, this contact is a normal ohmic (Dirichlet) boundary condition.

EXAMPLES

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

Define all electrodes except number 2 to be neutral, and number 2 is aluminum. Besides a workfunction, electrode number 2 also includes finite surface recombination velocities and barrier lowering. Note that the definition on the second card overrides that of the first.

```
CONTACT NUM=2 RESIS=1E5
CONTACT NUM=4 CON.RES=1E-6
```

Attach a lumped resistor to contact number 2 with a value of 10^5 Ω-μ. Include distributed contact resistance (10^{-6} Ω-cm^{-2} ~ Aluminum) on contact 4.

COMMAND CONTOUR

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

If a quantity specified is a vector, the contour of its magnitude is plotted unless flag of either **X.compon** or **Y.compon** is specified in the same card. In that case, the corresponding component of the vector quantity is plotted.

When a solution is loaded, model dependent quantities (e.g., current density and recombination) to be plotted 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 value of the current density, the models used in the solution should be specified. The quantity to be plotted has no default.

SYNTAX

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

`plotted quantity:`

{ **POtential** = <logical> **|** **QFN** = <logical> **|**
QFP = <logical> **|** **VAlenc.b** = <logical> **|**
CONduc.b = <logical> **|** **DOping** = <logical> **|**
ELectrons = <logical> **|** **Holes** = <logical> **|**
NET.CHarge = <logical> **|** **NET.CArrier** = <logical> **|**
J.Conduc = <logical> **|** **J.Electr** = <logical> **|**

J.Hole = <logical> **|** **J.Displa** = <logical> **|**
J.Total = <logical> **|** **E.field** = <logical> **|**
Recomb = <logical> **|** **Flowlines** = <logical> **|**
TEMP.Elec = <logical> **|** **TEMP.Hole** = <logical> **|**
TEMP.Lat = <logical> **|** **VELO.Elec** = <logical> **|**
VELO.Hole = <logical> **|** **Impact** = <logical> **|**
GEN.Elec = <logical> **|** **GEN.Hole** = <logical> **|**
ALPHAN = <logical> **|** **ALPHAP** = <logical> **|**
E.field }

`range definition:`

MIn.value = <real> **MAx.value** = <real>
DEl.value = <real> **NContours** = <integer>

`control:`

LIne.type = <integer>
ABsolute = <logical> **LOgarithm** = <logical>
{ **X.compon** = <logical> **|** **Y.compon** = <logical> }
PAuse = <logical> **COLor** = <logical>
C1.color = <integer> **C2.color** = <integer>
C3.color = <integer> **C4.color** = <integer>
C5.color = <integer> **C6.color** = <integer>
C7.color = <integer> **C8.color** = <integer>
C9.color = <integer> **C0.color** = <integer>

PARAMETERS

ABsolute

A logical flag to specify that the absolute value of the plotted quantity be taken. (Default: false)

ALPHAN, **ALPHAP**

Logical flags for impact ionization rates due to electrons and holes (α_n and α_p in Eq. (3.37)), respectively. (Defaults: false)

COLor

A logical flag to specify that color fill, as opposed to simple lines, should be used to delineate contours. (Default: false)

CONduc.b

A logical flag for potential corresponding to the conduction band edge. (Default: false)

C1.color, **C2.color**, **...** , **C9.color**, **C0.color**

Integer numbers to specify the color types for the contours. (Defaults: **C1.color**, 6; **C2.color**, 7; **C3.color**, 8; **C4.color**, 9; **C5.color**, 10; **C6.color**, 11; **C7.color**, 12; **C8.color**, 13; **C9.color**, 14; **C0.color**, 15)

DEl.value

A real number parameter to specify the interval between values of adjacent contours. (No default)

DOping

A logical flag for net doping concentration. (Default: false)

E.field

A logical flag for electric field. (Default: false)

ELectrons

A logical flag for electron concentration. (Default: false)

Flowlines

A logical flag for current flow lines. (Default: false)

GEN.Elec, GEN.Hole

Logical flags for impact ionization generation rates due to electrons and holes (first and second terms on the right hand side of Eq. (3.37)), respectively. (Defaults: false)

Holes

A logical flag for hole concentration. (Default: false)

Impact

A logical flag for generation rate due to impact ionization (*G* in Eq. (3.37)). (Default: false)

J.Conduc, J.Displa, J.Electr, J.Hole, J.Total

Logical flags for the conduction, displacement, electron, hole, and total current densities, respectively. (Default: false)

LIne.type

An integer parameter to define the type of plot line. (Default: 1)

LOgarithm

A logical flag for using logarithmic instead of original value of the quantity to plot. For rapidly varying quantities, the logarithmic value is often more revealing. Since many of the quantities may be negative, the program actually uses

$$\log(x) = \operatorname{sign}(x) \cdot \log(1 + |x|) \qquad \textbf{(M.1)}$$

to avoid overflow. To get the true logarithmic value of a quantity, use **ABsolute** and **LOgarithm**. The absolute value is then taken first and there is no danger of negative arguments. (Default: false)

MIn.value, MAx.value

Real number parameters to specify the minimum and maximum values of the quantity to be plotted, respectively. If the value is logarithmic, the minimum and maximum should be given as the logarithmic bounds. (Defaults: the actual minimum and maximum values of the quantity to be plotted over the device)

NContours

An integer parameter to specify the number of contours to be plotted. This is an alternative to the above specification. (No default)

NET.CArrier

A logical flag for net carrier concentration. (Default: false)

NET.CHarge

A logical flag for net charge density. (Default: false)

PAuse

A logical flag to cause the program to stop at the end of the plot so that a hard copy may be made before continuing. Execution can be resumed by hitting a carriage return. (Default: false)

POtential

A logical flag for electrostatic potential which is usually defined as the intrinsic Fermi level. (Default: false)

QFN, QFP

Logical flags for electron and hole quasi-Fermi levels, respectively. (Defaults: false)

Recomb

A logical flag for net recombination rate. (Default: false)

TEMP.Elec, TEMP.Hole, TEMP.Lat

Logical flags for electron, hole, and lattice temperatures, respectively. (Defaults: false)

VAlenc.b

A logical flag for potential corresponding to the valence band edge. (Default: false)

VELO.Elec, VELO.Hole

Logical flags for electron and hole velocities, which are derived from the carrier concentration and current density, respectively. (Defaults: false)

X.compon, Y.compon

Logical flags for taking *x* and *y* components of a vector quantity, respectively. (Default: false)

EXAMPLES

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

Plots the contours of potential from -1 volts to 3 volts in steps of 0.25 volts.

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

In this example, the log of the doping concentration is plotted from 1.0×10^{10} to 1.0×10^{20} in steps of 10. By specifying **ABsolute**, both the *n*-type and *p*-type contours are shown.

```
CONTOUR FLOW NCONT=11
```

The current flow lines are plotted. The number of flow lines is 11 so that 10% of the current flows between adjacent lines.

COMMAND DEEPIMPURITY

The **DEEPIMPURITY** card specifies the impurity doping with variable ionization energy.

This card has essentially the same features as the **DOping** card except it allows users to specify the impurity ionization energy as well. Its main application is for trap DC analysis as shown in the example for MODFET simulation.

SYNTAX

DEepimpurity `<DOping card parameters>`

```
+     <ionization energy>
```

```
Doping card parameters:
```

All parameters available in **DOping** card (see **DOping**)

```
ionization energy
```

Eioniz = <real>

PARAMETERS

Eioniz

Real number parameter to specify the ionization energy for (deep) impurity level, in units of eV. It is E_C - E_D for donors and E_A - E_V for holes. (Default: same as **EDb** and **EAb** in **MAterial** card)

EXAMPLES

```
DEEPIMP UNIF X.L=1.625 X.R=1.925 Y.T=0.03
+     Y.B=0.031 ACCEP CONC=1.0e19 EION=0.7
DEEPIMP UNIF X.L=2.075 X.R=2.375 Y.T=0.03
+     Y.B=0.031 ACCEP CONC=1.0e19 EION=0.7
```

This example shows how the **DEEPIMPURITY** card can be used to specify the surface traps. The traps have type of acceptors with ionization energy 0.7 eV (i.e., E_A - E_V = 0.7 eV, almost in the mid bandgap for GaAs). The trap density if viewed from per unit area would be $1\times10^{19}\times(0.031-0.03)\times1\times10^{-4} = 1\times10^{12}$ cm^{-2}. And the traps span in the *x*-direction from 1.625 to 1.925 μ and from 2.075 to 2.375 μ.

COMMAND DOPING

The **DOPING** card specifies the impurity doping in selected regions of the device.

SYNTAX

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

`profile type:`

{ **Erfc** = <logical> **|** **Gaussian** = <logical> **|**
Uniform = <logical> **|** **SUprem3** = <logical> **|**
OLD.Suprem3 = <logical> **|** **SImpldop** = <logical> **|**
S4geom = <logical> **|** **AScii** = <logical> }

`location:`

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

`region:`

REgion = <integer>

`profile specification:`

If `<profile type>` = **Erfc** or **Gaussian**:

`<profile>`
COncentrat = <real> **JUnction** = <real>
J.Conc = <real> **SLice.lat** = <real>
or

DOSe = <real>   **CHaracter** = <real>

or

COncentrat = <real>   **CHaracter** = <real>

and

{ **N.type** or **DONor** = <logical> **|**

P.type or **ACceptor** = <logical>

and any combination of

RAtio.lat = <real>   **Erfc.lat** = <logical>

Lat.char = <real>   **PEak** = <real>

DIrection = <character>

If `<profile type>` = **Uniform**:

COncentrat = <real>

{ **N.type** or **DONor** = <logical> **|**

P.type or **ACceptor** = <logical> }

If `<profile type>` **SUprem3** or **OLD.Suprem3**:

`<Input file>`

Infile =<filename>

`<dopant>`

Boron = <logical>   **PHosphor** = <logical>

ARsenic = <logical>   **ANtimony** = <logical>

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

`<Two-dimensional spread>`

DIrection = <x or y>   **STart** = <real>

RAtio.lat = <real>

`If  <profile type> =` **AScii**:

`<Input file>`
Infile =<filename>

`<dopant type>`
N.type or **DONor** = <logical>
P.type or **ACceptor** = <logical>
The ASCII concentrations in **Infile** are read and added to (**N.type**) or subtracted from (**P.type**) the impurity profiles.

`<Two-dimensional spread>`
DIrection = <x or y>   **STart** = <real>
RAtio.lat = <real>

If `<profile type>` = **S4geom** or **SImpldop**:

Infile = <filename>

`save:`

Outfile = <filename>

PARAMETERS

ACceptor (P.type)

A logical flag to specify *p*-type doping impurity (i.e. acceptors). (Default: false)

AScii

A logical flag for input doping file being in ASCII. If **AScii** is specified with **SUprem3**, then an ASCII `SUPREM-III` export file is expected.

AScii without **SUprem3** allows for input a simple ASCII data file containing information of concentration versus depth. The format of the ASCII input file is a depth in µ followed by a concentration in cm^{-3}-- one pair per line. By convention, positive concentrations refer to donors (*n*-type), while negative concentration values refer to acceptors (*p*-type). (Default: false)

Boron, **PHosphor**, **ARsenic**, **ANtimony**

Logical flags to indicate that the selected dopant profile will be extracted from the `SUPREM-III` save file specified by **Infile**. (Defaults: false)

CHaracter

A real number parameter for the principal characteristic length used in computing profiles specified by either **Erfc** or **Gaussian**, in units of µ. (No default)

COncentrat

A real number parameter for the peak doping concentration, in units of cm^{-3}. For a **Uniform** profile, it is the value of the doping level. (No default)

DIrection

A character of either x or y. Along with **STart** and **RAtio.lat**, it specifies where to locate a one-dimensional profile in the two-dimensional device and how to extend it to the second dimension. **DIrection** is the axis along which the profile is to be directed. (Default: y)

DONor (**N.type**)

A logical flag to specify *n*-type doping impurity (i.e. donors). (Default: false)

DOSe

A real number parameter for the total dose, in units of cm^{-2}. (No default)

ERFC, Gaussian, Uniform

Logical flags used to analytically describe profile shapes in either complementary error function (**Erfc**), or Gaussian (**Gaussian**), or constant (**Uniform**) form. Doping is introduced in the intersection of the location box and the region selected. The default box is set up to include the entire region. (Defaults: false)

ERFC.lat

A logical flag to specify ERFC doping distribution in the lateral direction. (Default: false)

Infile

A character string for the name of the input file from which the doping is either directly obtained or interpolated onto an existing mesh. If **AScii** is also specified, the doping profile specified in **Infile** is added to the previous, if any, impurity profile. (Default: null)Sc

J.conc

A real number parameter for the concentration at the junction, in units of cm^{-3}. (Default: **COncentrat** / 100)

JUnction

A real number parameter for the location of the junction, in units of μ. it must be located in semiconductor region, outside the constant doping box. When **JUnction** is used, the program 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.lat**. (No default)

Lat.char

A real number for the characteristic length of the distribution in the lateral direction as opposed to that in the principal direction (**CHaracter**). (No default)

OLd.Suprem3

A logical flag to read the doping profile from the earlier version of `SUPREM-III` process simulation program. A binary structure file specified by **Infile** is read in. The defaults for the constant doping box are set up as a line, parallel to the surface, and located at **STart**. (Default: false)

OUtfile

A character string for the name of a binary file in which all the **DOping** cards in the present file will be saved. 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 re-read after regridding to calculate the doping profile on a new mesh. (Default: null)

PEak

A real number parameter specifying the peak position of a doping profile, in units of μ. (Default: 0.)

RAtio.lat

A real number parameter governing the profile outside the constant doping box. The lateral profile is assumed to have the same form as the principal one, but is shrunk/expanded by the factor **RAtio.lat**. (Default: 0.8)

REgion

An integer parameter for the sequence number of the region where doping profile is to be added. This is an optional parameter. Multiple regions may be included by concatenating their region numbers into a single integer. (Default: all semiconductor region numbers concatenated)

SImpldop

A logical flag specifying use of the doping profile from a `SIMPL-2` file (rectangular grid) to be interpolated onto an existing `PISCES` mesh. (default: false)

SLice.lat

A real number parameter, in units of μ, to specify the location in the lateral direction (as opposed to the principal one), at which the junction depth specified by **JUnction** is to be searched along the principal direction. (No default)

STart

A real number parameter to specify where a one-dimensional doping profile is to be located along the direction specified by **DIrection**, used together with **RAtio.lat** also. (Default: 0.0)

SUprem3

A logical flag to read the doping profile from the "*export*" file saved during a process simulation using the late release of `SUPREM-III`. The default is to read binary export files. If the **AScii** parameter is also specified, then ASCII `SUPREM-III` export files will be expected. The defaults for the constant doping box are set up as a line, parallel to the surface, and located at **STart**. (Default: false)

S4geom

A logical flag to read the doping profile from the "*geometry*" file saved during `SUPREM-IV` 2D process simulation. (Default: false)

X.Left, **X.Right**, **Y.Top**, **Y.Bottom**

Real number parameters to specify the *x* and *y* bounds of the constant doping box within the simulation region. Inside the box the doping level is constant. Outside this area it falls off along the principal axis according to the profile specifications and along the lateral axis according to the lateral parameters. (The default bounds of the box depend on the type and principal direction of the profile as shown in Table M.1. In the case of Erfc or Gaussian or Suprem3, the bounds are defaulted to a line perpendicular to the principal axis and located at the start (S) / peak (P) of the profile, respectively. This is denoted by the entry SP in the table.)

Table M.1 Default bounding parameters for constant doping box. Uniform refers to the **Uniform** profile while *x*- and y-directions apply to one of **Erfc**, **Gaussian**, **SUprem3**, **OLD.Suprem3**, or **AScii** profiles when the principal direction is specified as such.

Parameter	Default		
	Uniform	All other profile type	
		x - direction	*y* - direction
X.Left	–∞	SP	–∞
X.Right	∞	X.Left	∞
Y.Top	–∞	–∞	Y.Bot
Y.Bottom	∞	∞	SP

EXAMPLES

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

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

```
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
```

An n-channel MOSFET with Gaussian source and drain. Because the default **X.Right** is +∞, 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 µm.

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

Reads a SUPREM bipolar profile and add it to a uniform substrate concentration. Adds doping only to those points lying in region 1.

```
COM  *** SUBSTRATE ***
DOP REGION=1 UNIFORM CONC=9.999463e+14 p.type
COM *** EMITTER ***
DOP REGION=1 ERFC N.TYPE CON=1e20 CHAR=0.1
+    X.LEF=-1 X.RIG=0 R.LAT=0.8
COM *** BASE ***
DOP REGION=1 SUPREM3 INFILE=base.exp BORON
+    X.LEF=-4 X.RIG=0 R.LAT=0.8
```

```
COM *** COLLECTOR ***
DOP REGION=1 GAUSS PHOS CON=1e17 CHAR=0.8
+    X.LEF=-7 X.RIG=0 R.LAT=0.8
```

Simulates a triple-diffused bipolar by using a mixture of analytic and SUPREM-III profiles. Uses an erfc for the emitter, a SUPREM-III profile for the base, a gaussian for the collector, and adding it to a uniform substrate concentration. Adds doping only to those points lying in region 1.

COMMAND ELECTRODE

The **ELECTRODE** card specifies the location of electrodes including thermal contacts in a rectangular mesh.

SYNTAX

ELEctrode `<number> <position> <type>`

`number:`

> **Number** = <integer>

`position:`

> { **IX.Low** = <integer> **IX.High** = <integer>
> **IY.Low** = <integer> **IY.High** = <integer> **|**
> **X.Low** = <real> **X.High** = <real>
> **Y.Low** = <real> **Y.High** = <real> }
> **Contact** = <logical> **|** **SURface** = <logical>
> **SUBstrate** = <logical> **|** **SYmmetri** = <logical>

`type:`

> **Thermal** = <logical>

PARAMETERS

Contact

A logical flag for placing an electrode on the top of the surface of the semiconductor region (such as at the insulator-semiconductor interface) in the device structure. If **Y.Low** is specified, the electrode has thickness of **Y.Low** extending to the semiconductor region.

IX.Low, **IX.High**, **IY.Low**, **IY.High**

Integer numbers to indicate that grids having x and y indices between **IX.Low** and **IX.High** and between **IY.Low** and **IY.High**, respectively, are designated electrode nodes.

Number

An integer number for the electrode. There may be up to ten electrodes, numbered 1, 2, 3, ... , 9, 0. They may be assigned in any order, but if there are *N* electrodes, none can have an electrode number above *N*.

SUBstrate

A logical flag for placing an electrode at the bottom of the semiconductor region in the device structure. If **Y.Low** is specified, then the electrode has thickness of **Y.Low** extending to the semiconductor region. (Default: false)

SURface

A logical flag for placing an electrode on the top (i.e., surface) of the device structure.(Default: false)

SYmmetri

A logical flag to place an electrode which is symmetric with the one specified in this card. The device must have a symmetric structure (Default: false)

Thermal

A logical flag to specify that the electrode is a thermal contact and there is no equal-potential requirement for those nodes on the same “electrode” rather lattice temperature is required to be the same for those nodes. (Default: false, meaning normal electrode)

X.Low, **X.High**, **Y.Low**, **Y.High**

Real number parameters, in units of μ, to indicate that grids having x and y coordinates between **X.Low** and **X.High** and between **Y.Low** and **Y.High**, respectively, are designated electrode nodes. The function of these parameters is essentially the same as that of **IX.Low, IX.High, IY.Low,** and **IY.High** except that the grid location is identified by its real coordinates. (No defaults)

EXAMPLES

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

Define a typical back-side contact.

COMMAND ELIMINATE

The **ELIMINATE** card terminates grids along lines in a rectangular mesh.

SYNTAX

ELIminate `<direction> <range>`

`direction:`

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

`range:`

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

PARAMETERS

IX.Low, **IX.High**, **IY.Low**, **IY.High**

Integer parameters for indices of grids used to define a rectangular region where grids along every second line are removed. Successive eliminations of the same region remove grids following the same principle but are based on the most recent mesh. For horizontal elimination, the vertical bounds should be decreased by one at each re-elimination of the same region, and conversely for vertical eliminations.

X.direction, **Y.direction**

Logical parameters to determine whether to eliminate grids along vertical (**Y.direction**) or horizontal (**X.direction**) lines. One must be chosen.

EXAMPLES

```
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
```

Points along vertical lines between 10 and 20 are removed except of those along line **IX** = 5.

COMMAND END (QUIT)

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.

SYNTAX

ENd or **Quit**

PARAMETERS

None.

EXAMPLES

```
END
```

COMMAND EXTRACT

The **EXTRACT** card extracts integrated electrical quantities, such as net charge, over a selected region based on the solution.

SYNTAX

EXtract `<variable> <bounds> <file i/o>`

`variable:`

> **NET.CHar** = <logical>   **NET.CArr** = <logical>
> **Electron** = <logical>   **Hole** = <logical>
> **Metal.charge** = <logical>   **N.Resist** = <logical>
> **P.Resist** = <logical>   **N.Current** = <logical>
> **P.Current** = <logical>

`bounds:`

> **X.MIn** = <real>   **X.MAx** = <real>
> **Y.MIn** = <real>   **Y.MAx** = <real>
> **Contact** = <integer>   **Regions** = <integer>

`file i/o:`

> **Outfile** = <filename>

PARAMETERS

Contact

An integer as the number of the contact for which electrode quantities, e.g. current and (metal) charge, are integrated in its portion intersecting the bounded region designated by **X.MIn**, **X.MAx**, **Y.MIn**, **Y.MAx**. (No default)

Metal.charge

A logical flag to indicate that integrated charge on a contact or its portion is to be calculated. This flag is useful for studies such as capacitance. (Default: false)

N.Current, P.Current

Logical flags for computing electron and hole currents, respectively, through an electrode or its portion. (Defaults: false)

NET.CHar, NET.CArr, Electron, Hole

Logical flags for integrated net charge, net carriers, electron and hole concentrations, respectively, over a selected region in the device. (Defaults: false)

N.Resist, P.Resist

Logical flags for the resistance of a cross section due to electrons and holes, respectively. The resistance of a cross section is defined as the one for the resistor with unit thickness in the third dimension and using the cross section as contacts. That is (for electrons),

$$R_n = \frac{1}{q\int n\mu_n ds} \tag{M.2}$$

where $\int ds$ for integration over the cross section, in units of $\Omega \ \mu^{-1}$. (Defaults: false)

Outfile

Character string for the name of an optional ASCII output file to which the result and bias information are to be written. (Default: null)

Regions

An integer to identify region(s) as defined in **Region** card. The integration will be conducted over only those nodes which fall within both the bounds specified by **X.MIn**, **X.MAx**, **Y.MIn**, **Y.MAx** and the particular set of regions specified by **Regions**. (No default)

X.MIn, X.MAx, Y.MIn, Y.MAx

Real numbers for the coordinates (in units of μ) of bounds which define the rectangular region for the integration. (Default: the entire device)

EXAMPLES

```
EXTRACT P.RESIST
```

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 METAL.CH CONT=1 X.MIN=-2.0 X.MAX=2.0
+    Y.MAX=-0.0499 Y.MIN=-0.0501
```

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 = 0.0.

COMMAND IMPACT

The **IMPACT** card specifies the impact ionization model used for (local) electric-field dependence. For many devices, the impact ionization model for continuity equations allows the accurate prediction of avalanche breakdown. The current models are for Si only. See also, relevant parameters in the **MODELS** and **CONTOUR** cards for both field and (carrier) energy dependent impact ionization models.

The Newton method with 2-carrier must be specified on the **METHOD** card since impact ionization is a 2-carrier process.

SYNTAX

IMpact `<model> <mode>`

`model:`

> **CRowell** = <logical>   **MOnte** =<logical>
> **LAMDAE** = <real>   **LAMDAH** = <real>

`mode:`

> **Break** = <logical>   **Ionpath** =<logical>

PARAMETERS

Break

A logical flag to calculate the breakdown voltage using the ionization integral. (Default: false)

Crowell

A logical flag to use the analytical expression proposed by Crowell and Sze for temperature dependent impact ionization rates [45]. There are four parameters in this expression: ionization energy, $\varepsilon_i = 1.5E_g$, Raman optical phonon energy, ε_r, carrier mean free path for optical phonon generation, λ, and the low temperature limit of the mean free path, λ_0. λ for electrons and holes can be altered by users through **LAMDAE** and **LAMDAH** in this card. The other parameters have values as used in [45]. If **Crowell** is not specified, the model as specified in Eq. (3.38) and Table 3.11 is used. (Default: false)

Ionpath

A logical flag to print the nodes, electric field, carrier ionization rates along the integration path for calculation of breakdown voltage. (Default: false)

LAMDAE, LAMDAH

Real number parameters for electron and hole mean free paths (λ in [45]), respectively, in units of cm. (Default: **LAMDAE** = 6.2×10^{-7}, **LAMDAH** = 3.8×10^{-7} at 300 K [45])

Monte

Character string for the name of the output solution file from a Monte Carlo (MC) solver. A non-null string serves also as the flag to indicate that the impact ionization rates (α's) are to be calculated from this MC solution file. (Default: null)

EXAMPLES

```
IMPACT  CROWELL LAMDAE=6.2e-7  LAMDAH=3.8e-7
```

Use the Crowell and Sze formulae with the default mean free paths.

COMMAND INCLUDE

The **INCLUDE** statement provides a shorthand way to include information from other files in the `PISCES` input file. The statements in the **INCLUDE**d file will be inserted into the `PISCES` input file in place of the **INCLUDE** statement when the input file is processed. The statements in the **INCLUDE**d file must use correct `PISCES` input syntax, and they must be in correct order with respect to the other statements in the present input file when the **INCLUDE**d file is expanded by the input parser. This is most useful for libraries of material and model parameters.

SYNTAX

INClude (**SOUrce**) `<filename>`

`filename`:

A character string

PARAMETERS

None.

EXAMPLES

```
INCLUDE MAT.ini
```

COMMAND INTERFACE

The **INTERFACE** card allows the specification of interface parameters (recombination velocities and fixed charges) at semiconductor-insulator boundaries.

SYNTAX

INTerface `<number> <parameters> <location>`

`number:`

Number = <integer>

`parameters:`

S.N = <real> **S.P** = <real>
Qf = <real>

`location:`

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

PARAMETERS

Number

An integer for the sequence number of the interface. The maximum number is currently set to 10. (No default)

Qf

Real number for the interfacial fixed charge density, in units of cm^{-2}. (Default: 0.0)

S.N, **S.P**

Real number parameters for electron and hole surface recombination velocities, respectively, at the semiconductor surface, in units of cm/s. (Defaults: 0.0)

X.MIn, **X.MAx**, **Y.MIn**, **Y.MAx**

Real number parameters for the coordinates of bounds to define a rectangular region, in units of µ. Any oxide/semiconductor interfaces found within this region are applied with parameters specified in this card. (Default: entire device)

EXAMPLES

```
INTERFACE X.MIN=-4 X.MAX=4 Y.MIN=-0.5 Y.MAX=4
+    QF=1E10 S.N=1E4 S.P=1E4
```

Define an interface with both fixed charge and recombination velocities.

COMMAND LOAD

The **LOAD** card loads previous solutions from files for plotting or as initial guess for present solution at other bias point.

SYNTAX

LOAd `<solution files> <actions>`

`solution files`:

> **INFile** or **IN1file** = <filename>   **IN2file** = <filename>
> **Outdiff** = <filename>   **Ascii** = <logical>

`actions`:

> **Difference** = <logical>   **No.check** = <logical>
> **Restore** = <logical>

PARAMETERS

Ascii

A logical flag to specify that any files read in or written to by **Load** should be ASCII rather than binary. (Default: false)

Difference

A logical flag to indicate that the difference between two solutions (**IN1file** and **IN2file**) is to be analyzed. The difference may be stored by specifying **Outdiff** and can only be used for the purpose of data plotting or extraction. (Default: false)

INFile or **IN1file**, **IN2file**

Character strings for file names. The **INFile** (or **IN1file**) and **IN2file** specify input files for solutions. **INFile** (or **IN1file**) and **IN2file** represent present and previous solutions, respectively. If only one solution is to be loaded (for plotting or initial guess requiring a single solution), **INFile** should be used. If two input files are needed to perform an extrapolation to obtain an initial guess (i.e., option **Extrapolate** on the **SOLVE** card), **IN1file** and **IN2file** should be used. The solution in **IN2file** is the first to be lost when new solution is obtained. (Defaults: null)

No.check

A logical flag to prevent `PISCES` from checking material parameter difference between the loaded file(s) and those specified in the current input file. Checking will never be done for loading of ASCII solution file(s). (Default: false)

Outdiff

Character string for name of file to which the difference from comparison of loaded solution files (flag **Difference**) is written to. (Default: null)

Restore

A logical flag to restore the material parameters and models in the loaded file as the current ones. (Default: false)

EXAMPLES

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

Specifies that a single solution file called SOL.IN should be loaded.

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

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 DIFF OUTD=SOL1-2
```

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

COMMAND LOG

The **LOG** card allows the I-V and/or AC characteristics of a run to be logged to ASCII file(s) specified by the user. Any I-V or AC data obtained subsequent to the card are saved. If a log file is already open, it is closed and a new file opened.

The format of the data saved in AC log file is as follows. The first line in the file records the number of terminals for the device under simulation. The lines starting with "*" record the number of the activation terminal, the magnitude of the AC input signal (sinusoidal), frequency, and DC bias (terminal voltages if lumped external resistors exist). The lines without "*" record, in sequence, the number of the activation terminal, AC conductances, capacitances, and magnitude of admittances.

SYNTAX

LOG `<file specification>`

`file specification`:

Ivfile or **Outfile** = <filename>   **Acfile** = <filename>

PARAMETERS

Acfile

Character string for the name of file to which the AC simulation results are to be written. (Default: null, meaning no AC data saved)

Ivfile or **Outfile**

Character string for the name of file to which simulated I-V information is to written. (Default: null, meaning no DC data saved)

EXAMPLES

```
LOG IVFIL=out.iv ACFIL=out.ac
```

Save the simulated I-V data in a file called out.iv and AC data in out.ac.

COMMAND MATERIAL

The **MATERIAL** card allows users to change default material parameters for the base semiconductor in the device (see Section 4.3.1 for explanation of base material). For the following three parameters: (static) dielectric constant, ε, thermal conductivity at room temperature, κ, and its temperature coefficient, α (in Eq. (3.36)), they can be changed from region to region using region number assigned in **Region** card. For default parameter value see Table 4.1. Some parameters which are not included in Table 4.1 are listed here for Si and GaAs (Table M.2 and Table M.3).

SYNTAX

MAterial `<region> <material parameters>`

`region:`

{ **NUmber** = <integer> **|** **Region** = <integer> }

`material parameters:`

EG300 = <real> **EGAlpha** = <real>
EGBeta = <real> **AFfinity** = <real>
Permittivity = <real> **Vsat** = <real>
MUN = <real> **MUP** = <real>
G.surface = <real> **TAUN0** = <real>
TAUP0 = <real> **NSRHN** = <real>
NSRHP = <real> **ETrap** = <real>
AUGN = <real> **AUGP** = <real>
NC300 = <real> **NV300** = <real>
ARICHN = <real> **ARICHP** = <real>

GCb = <real>   **GVb** = <real>
EDb = <real>   **EAb** = <real>
KP.300 = <real>   **KP.Alpha** = <real>

PARAMETERS

AFfinity

Electron affinity, in eV.

ARICHN, ARICHP

Richardson constants for electrons and holes (A_n^{**} and A_p^{**} in Eq. (C.1)), respectively, in units of A K^{-1} cm^{-2}. (Default: Table M.2).

AUGN, AUGP

Auger coefficients for electrons and holes (c_n and c_p in Eq. (3.47)), respectively, in units of cm^6 s^{-1}. (Default: Table M.2)

EAb, EDb

Acceptor and donor ionization energies ($E_A - E_V$ and $E_C - E_D$ in Eq. (C.2)), respectively, in units of eV. (Default: Table M.2)

EGAlpha

α in Eq. (4.2) of units eV K^{-1}.

EGBeta

β in Eq. (4.2) of units K.

EG300

Energy bandgap at 300 K (Eq. (4.2)), in eV.

ETrap

Trap level = E_t -E_i, in units of eV. (Default: Table M.2)

GCb, **GVb**

Degeneracy factors for the donor and acceptor levels (g_D and g_A in Eq. (C.2)), respectively, unitless. (Default: Table M.2)

G.surface

Surface mobility reduction factor (g_{surf} in Eq. (C.3)), unitless. (Default: Table M.2)

KP.300

Thermal conductivity at 300 K (κ_0 in Eq. (3.36)), in units of W cm^{-1} K^{-1}. (Default: Table 3.10)

KP.Alpha

Temperature coefficient for the thermal conductivity (α in Eq. (3.36)), unitless. (Default: 1.2)

MUN, **MUP**

Low-field mobilities for electrons and holes, respectively, in cm^2 V^{-1} s^{-1}.

NC300, **NV300**

Effective densities of states for conduction and valence bands, respectively, at 300 K, in cm^{-3}.

NSRHN, **NSRHP**

Reference concentration in SRH recombination formula (N_{SRH} in Eq. (3.45)) for electrons and holes, respectively, in cm^{-3}. (Default: Table M.2)

NUmber or **Region**

An integer for the sequence number of the region where any of three material parameters: static dielectric constant (i.e. **Permittivity**), thermal conductivity at room temperature (**KP.300**) and its temperature coefficient (**KP.Alpha**) can be changed from region to region. Note that all other material parameters can only be changed for the entire device (base material). (No default)

Permittivity

Static dielectric constant, unitless.

TAUN0, **TAUP0**

Minority carrier lifetimes for electrons and holes as used in SRH formula (τ_0 in Eq. (3.45)), in s. (Default: Table M.2)

Vsat

Carrier saturation velocity, in cm/s. (Default: see Table 3.9 and Table 4.1)

Table M.2 material parameters for Si and GaAs.

Parameter	Constant (units)	Silicon	GaAs
G.surface	Surface mob. reduction	1.0	1.0
TAUN0	Electron lifetime (s)	$1.0{\times}10^{-7}$	$1.0{\times}10^{-7}$
TAUP0	Hole lifetime (s)	$1.0{\times}10^{-7}$	$1.0{\times}10^{-7}$
NSRHN	SRH ref. conc. in Eq. (3.45) for *n* (cm^{-3})	$5.0{\times}10^{16}$	$5.0{\times}10^{16}$
NSRHP	SRH ref. conc. in Eq. (3.45) for *p* (cm^{-3})	$5.0{\times}10^{16}$	$5.0{\times}10^{16}$
ETrap	Trap level (eV)	0.0	0.0
AUGN	Auger coeff. (*n*) (cm^6/s)	$2.8{\times}10^{-31}$	$2.8{\times}10^{-31}$
AUGP	Auger coeff. (*p*) (cm^6/s)	$9.9{\times}10^{-32}$	$9.9{\times}10^{-32}$

Table M.2 material parameters for Si and GaAs.

Parameter	Constant (units)	Silicon	GaAs
ARICHN	Eff Richardson const (*n*) (A K^{-1} cm^{-2})	110	6.2857
ARICHP	Eff Richardson const (*p*) (A K^{-1} cm^{-2})	30	105
GCb	Cond band degen factor	2.0	2.0
GVb	Val band degen factor	4.0	2.0
EDb	Donor energy level (eV)	0.044	0.005
EAb	Acceptor energy level (eV)	0.045	0.005

Table M.3 Insulator permittivity

Insulator	Permittivity
Silicon dioxide	3.9
Silicon nitride	7.5
Sapphire	12.0

EXAMPLES

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

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 defaults, consistent with the semiconductor type chosen).

COMMAND MESH

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

SYNTAX

MESh `<type> <output files> <grid editor>`
+ `<smoothing key> <dimension> <electrodes>`

`type:` one of

`<Previous>`
Infile = <filename> **ASCII.In** = <logical>

`<Rectangular>`
Rectangular = <logical>
NX = <integer> **NY** = <integer>
Diag.flip = <logical>

`<Geometry>`
Geometry = <logical> **Infile** = <filename>
Flip.y = <logical> **SCale** = <integer>

`output files:`

OUTFile = <filename> **ASCII.Out** = <logical>

`grid editor:`

OUT.Asc = <filename> **Flip.y**=<logical>
SCale = <integer>

```
smoothing key:
```

SMooth.key = <integer>

```
dimension:
```

CYLindrical = <logical> **Width** = <real>

```
electrodes:
```

Elec.bot = <logical> **Poly.elec**

PARAMETERS

ASCII.In

A logical flag to indicate that the input mesh file is an ASCII rather than a binary file. (Default: false)

ASCII.Out

A logical flag to indicate the mesh file, as specified by **OUTFile**, to be saved is an ASCII one. Otherwise it will be binary. (Default: false)

Cylindrical

This logical flag specifies that the mesh, whether generated in the current run or read in from a file, is to be rotated about the right edge of the rectangular simulation region to permit the simulation of cylindrically symmetrical devices. This information will not be written to the mesh file; it must be specified in the input deck for each PISCES run. (Default: false)

Diag.flip

A logical flag to flip the diagonals in a rectangular mesh about the center line of the mesh. If not set, all the diagonals will be in the same direction. (Default: false)

Elec.bot

A logical flag to add an electrode to the bottom of the simulation region when the mesh file read in is a geometry file generated by a 2D process simulator such as `SUPREM-IV`. One of the possible applications of this flag is to add a substrate contact for a MOSFET structure. Currently, the electrode number assigned to this contact is hard-wired to 4 in the code. (Default: false)

Flip.y

A logical flag which reverses the sign of the y-coordinate. (Default: false)

Geometry

A logical flag to indicate that the input ASCII mesh file as specified by **Infile** was generated by either a 2D process simulator or by an external mesh generator. (Default: false)

Infile

The name of the previously generated mesh file. If flag **ASCII.In** is set, this file is an ASCII file. Otherwise, it is binary. If flag **Geometry** is set, this mesh file, which must be an ASCII one, was generated by either a 2D process simulator such as `SUPREM-IV` or by an external mesh generator. (Default: null)

NX, NY

Integer parameters for numbers of grids in the *x*- and *y*-directions for a rectangular mesh, respectively. (No defaults)

OUT.ascii

This logical flag indicates that an ASCII mesh file will be saved for later editing by an external grid editor. See Appendix B of [3] for details of the format. (Default: false)

OUTFile

Specifies the name of the mesh file to be saved. The saved mesh file is either ASCII or binary depending on the flag **ASCII.Out** and can be read in by a later `PISCES` run. (Default: null)

Poly.elec

A logical flag to convert the region with material type of polysilicon to an electrode. The condition and application scope of this flag are the same as those of **Elec.bot**. Because of the potential difference in integer representation of the polysilicon region for different process simulators, this flag is not guaranteed to work properly but it has been tested using one of the commercial versions of SUPREM-IV. Currently the electrode number assigned to this contact is hard-wired to 3 in the code. (Default: false)

Rectangular

A logical flag which initiates the generation of a rectangular mesh. (Default: true)

SCale

An integer factor by which all the coordinates read in are multiplied. (Default: 1)

SMooth.key

This integer parameter causes mesh smoothing as described in Section 4.6 of [3]. 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. (No default)

Width

This real number parameter specifies the width of the simulation region, i.e., the dimension in the z-direction, in units of microns. When this parameter is used, all units for currents become amperes instead of A/µ, otherwise simulated terminal currents are considered as current densities per micron in width. (Default: 1.0)

EXAMPLES

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

Initiates a rectangular mesh with number of grids 40 in *x*-direction and 17 in *y*-direction before possible grid elimination and requests the finished mesh to be stored in a (binary) file named mesh1.msh.

```
MESH INF=mesh1.msh OUT.ASC=mesh1.msh FLIP
+    ASCII.OUT
```

Reads a previously generated (binary) mesh file and generates an ASCII file for a mesh editor (the *y*-axis is inverted as specified by flag **FLIP** 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 GEOM INF=geom1 SMOOTH.K=13131
+    OUTF=mesh1.msh
```

Reads an ASCII geometry file, smooths the mesh, and stores the file for a later run (ASCII format). The smoothing does several averaging and flipping steps. The digits are read in a reverse order, so that the flipping comes first, followed by node averaging, and so on.

```
MESH GEOM INF=mos.geom POLY.ELEC ELEC.BOT
```

Reads in an ASCII geometry (mesh) file for a MOS structure generated by one of 2D process simulators, converts the polysilicon layer into the third contact (the first two are source and drain contacts), and add the fourth contact to the substrate. The numbering convention for contacts is currently hard-wired in the code.

COMMAND METHOD

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.

SYNTAX

METhod `<general parameters>`
+ `<method-dependent parameters>`

`general parameters`:

> **ITlimit** = <integer> **Xnorm** = <logical>
> **C.toler** = <real> **P.toler** = <real>
> **T.toler** = <real> **TL.toler** = <real>
> **Rhsnorm** = <logical> **LImit**=<logical>
> **PRint** = <logical> **Fix.qf**=<logical>
> **Biasparti** or **TRap** = <logical> **ATrap** = <real>

`method-dependent parameters`:

> <`Gummel method`>
>
> (The following parameters are for damping the Poisson updates)
> { **DVlimit** = <real> **|** **DAMPEd** = <logical> }
> **DElta** = <real> **DAMPLoop** = <integer>
> **DFactor** = <real>
>
> (The following parameters select acceleration methods for the

Gemmul iteration)

SInglepois = <logical>   **ICcg**=<logical>
LUcrit or **LU1crit** = <real>   **LU2crit** = <real>
Maxinner = <integer>   **ACCElerat** = <logical>
ACCSTArt = <real>   **ACCSTOp** = <real>
ACCSTEp = <real>

`<Newton method>`

AUtonr = <logical>   **NRcriter** = <real>
2ndorder = <logical>   **TAuto** = <logical>
TOl.time = <real>   **L2norm** = <logical>
Dt.min = <real>   **Extrapolate** = <logical>

PARAMETERS

ACCElerat, ACCSTArt, ACCSTOp, ACCSTEp

These parameters deal with an acceleration method for attaining faster overall convergence in the single-Poisson mode (**Carriers** = 0 in **SYmbolic** card). The logical flag **ACCElerat** 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 [3]. (Defaults: **ACCElerat**, false; **ACCSTArt**, 0.3; **ACCSTOp**, 0.6; **ACCSTEp**, 0.04)

AUtonr, Nrcriter

These two parameters are for implementing an automated Newton-Richardson procedure which attempts to reduce the number of LU decompositions per bias point. The logical flag **AUtonr** indicates that this algorithm is to be used. Real number parameter **Nrcriter** 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. (Defaults: **AUtonr**, false; **Nrcriter**, 0.1)

Biasparti or **TRap**, **ATrap**

Logical flag **Biasparti** (**TRap**) specifies that if a solution process starts to diverge, the electrode bias steps taken from the initial guess are reduced by the multiplicative factor **ATrap**, a real number. (Defaults: **Biasparti**, **TRap**, false; **ATrap**, 0.5)

DAMPEd

A logical flag to indicate the use of a more sophisticated damping scheme proposed by Bank and Rose [46] (this is the recommended option, particularly for large bias steps). (Default: false)

DAMPLoop

An integer for the maximum number of damping loops allowed to find a suitable damping coefficient. (Default: 10)

DElta

Real number parameter for the threshold value to determine the damping factor for $\Delta\psi$, must be between 0 and 1. (Default: 0.5)

DFactor

Real number parameter for the factor which serves to increase the initial damping coefficient for the next Newton loop. (Default: 10.0)

DT.min

Real number parameter used in transient simulation for the minimum time-step allowed, in seconds. (Default: 1×10^{-25})

DVlimit

Real number parameter to limit the maximum update in ψ for a single loop, in units of the thermal voltage kT/q. (Default: 0.1)

Extrapolate

A logical flag used in transient simulation. Uses a second-order extrapolation to compute initial guess for the successive time-step. (Default: false)

Fix.qf

A logical flag to fix the quasi-Fermi potential of each non-solved for carrier to a single value, instead of picking a value based on local bias (see Section 5 of Chapter 2 [3] and the “**P.bias**" and “**N.bias**" parameters on the **SOlve** card). (Default: false)

ICcg

A logical flag to choose whether or not to use iteration to solve the multi-Poisson loops. It should be set whenever doing multi-Poisson. (Default: false)

ITlimit

An integer parameter for the maximum number of allowed outer loops (i.e., Newton loops or Gummel continuity iterations). (Default: 15)

LImit

A logical flag to indicate that the convergence criterion should be ignored, and iterations are to proceed until **ITlimit** is reached. (Default: false)

LUCrit (LU1crit), LU2crit

Real number parameters to specify how much work is to be done per Poisson loop (see Section 3 [3]) in Gummel sequential solution method (flag **Gummel** in **SYmbolic** card). The inner norm is required to decrease by at least **LUcrit** before returning, or to reach a factor of **LU2crit** 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). (Default: **LUCrit**, **LU1crit**, 3×10^{-3}; **LU2crit**, 3×10^{-2})

L2norm

A logical flag to indicate use of L2 norm in transient simulation. It specifies that the error norms be L2 as opposed to infinity norms for calculating the time-steps. (Default: true)

Maxinner

An integer to set the maximum number of ICCG iterations. (Default: 25)

PRint

A logical flag to print 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. (Default: false)

P.toler, C.toler, T.toler, TL.toler

Real number parameters for the termination criteria in iterations for solving the Poisson, carrier continuity, energy balance, and lattice thermal diffusion equations, respectively, unitless. (Defaults: 1×10^{-5})

Rhsnorm

A logical flag. If selected, the Poisson error is measured in C μ^{-1} and the continuity error in A μ^{-1}. **P.toler** then defaults to 1×10^{-26} C μ^{-1}, and **C.toler** to 5×10^{-18} A μ^{-1}. (Default: false)

SInglepois

A logical flag to indicate that only a single Poisson iteration is to be performed per Gummel loop as opposed to the default where the continuity equation is only solved after Poisson has fully converged. (Default: false)

TAuto

A logical flag used in transient simulation to force the program to select time-steps automatically from the local truncation error estimates. Note that automatic time-stepping is the default for the second-order discretization but is not allowed for the first-order scheme. (Default: true when flag **2ndorder** set)

TOl.time

Real number parameter used in transient simulation for the maximum allowed local truncation error, unitless. (Default: 5×10^{-3})

Xnorm

A logical flag to indicate that the error norm in Poisson updates is measured in units of kT/q, and that in carrier updates is measured relative to the local carrier concentration. In this case the default value for both **P.toler** and **C.toler** is 1×10^{-5}. (Default: true)

2ndorder

A logical flag used in transient simulation to specify that the second-order discretization of Bank, *et al*. [47] be used as opposed to the first-order backward difference method. (Default: true)

EXAMPLES

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

Specifies that for a simulation using the Gummel method (as previously specified by an appropriate symbolic card), damping is to be employed and the Poisson error tolerance should be 1×10^{-30} C μ^{-1}. Note that because **Xnorm** defaults 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 TRAP ATRAP=0.5
SOLVE   INIT
SOLVE V2=3 V3=5 OUTFILE=outa
```

This example illustrates the **TRap** feature, which can be quite useful for capturing knees of IV curves for devices such as SCRs. 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 **ATrap** (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-II` 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 TOL.TIME=1E-3 AUTONR
```

This is an example of transient simulation. By default, the second-order discretization is used, but the required LTE, 1×10^{-3}, is smaller than the default. Newton-Richardson is also used. Note that because the Jacobian is exact for the second part (BDF-2) of the composite time-step, there should be very few factorizations for the BDF-2 interval when **AUtonr** is specified (see chapter 2 of this report).

COMMAND MOBILITY

The **MOBILITY** card specifies the values for the parameters in the mobility models.

SYNTAX

MOBility `<carrier> <region> <numerical parameters>`

`carrier:`

HOle = <logical>

`region:`

BULk = <logical>

`numerical parameters:`

UMAx = <real> **UMIn** = <real>
EXP1.tem = <real> **EXP.Conc** = <real>
EXP2.tem = <real> **Ncrit** = <real>
Vsat = <real> **EC.Crit** = <real>
UConst = <real> **G.surface** = <real>
CHar.int = <real> **Alpha** = <real>
U0.unv = <real> **ETa** = <real>
EC.Unv = <real> **Qss.conc** = <real>
Int.off = <real> **DIst.epe** = <real>
EXP.Eper = <real> **BNL** = <real>
BPL = <real> **CNPre** = <real>
CPPre = <real> **CNExp** = <real>
CPExp = <real> **DELTAN** = <real>
DELTAP = <real>

PARAMETERS

Alpha

Real number for the coefficient α in Intel’s local field dependent mobility model of transverse field reduction (Eq. (3.16)). (Defaults: 1.02 for electrons and 0.95 for holes)

BNL, BPL

Real number parameters for coefficient B in Eq. (3.18) (Lombardi’s mobility model) for electrons and holes, respectively. (Defaults: **BNL**, 4.75×10^{7}; **BPL**, 9.93×10^{7})

BUlk

A logical flag to indicate that parameters specified in the present card apply to the bulk as opposed to the surface mobility model. These parameters include: **EXP1.tem**, **EXP.Conc**, **EXP2.tem**, **Ncrit**, **UMAx**, and **UMIn** (Default: false, meaning that the surface mobility parameters are applied)

CHar.int

Real number parameter of λ in Eq. (3.5) to specify the decay length for the complementary error function used for the transition from bulk to surface mobility, in units of μ. (Default: 0.03)

CNExp, CPExp

Real number parameters for β in Eq. (3.18) (Lombardi’s mobility model) for electrons and holes, respectively. (Defaults: **CNExp**, 0.125; **CPExp**, 0.0317)

CNPre, **CPPre**

Real number parameters for α in Eq. (3.18) (Lombardi's mobility model) for electrons and holes, respectively. (Defaults: **CNPre**, 1.74×10^{5}, **CPPre**, 8.84×10^{5})

DELTAN, **DELTAP**

Real number parameters for δ in Eq. (3.18) (Lombardi's mobility model) for electrons and holes, respectively. (Defaults: **DELTAN**, 5.82×10^{14}, **DELTAP**, 2.05×10^{14})

DIst.epe

Real number parameter for the characteristic length L in computing the structure-dependent transverse field (Eq. (3.21)), in units of μ. (Default: 0.5)

EC.Crit

Real number parameter for E_{crit} in Eq. (3.15) (Intel's local field mobility model), in units of V/cm. (Default: 4.2×10^{4} for electrons and 3.0×10^{4} for holes)

EC.Unv

Real number parameter for E_{univ} in Eq. (3.16) (Intel's local field mobility model), in units of V/cm. (Defaults: 5.71×10^{5} for electrons and 2.57×10^{5} for holes)

ETa

Real number coefficient for η in Eq. (3.22) of Watt's surface mobility model. (Defaults: 0.5 for electrons and 0.33 for holes)

EXP.Conc

Real number parameter of c in Eq. (3.6) for analytical low-field mobility model. (Defaults: see Table 3.3)

EXP.Eper

Real number parameter of β in Eq. (3.15) for Intel's local field mobility model (Section 3.1.2.1). (Defaults: 0.5)

EXP1.tem

Real number parameter of b in Eq. (3.6) for the analytical low-field mobility model. (Defaults: see Table 3.3)

EXP2.tem

Real number parameter of d in Eq. (3.6) for the analytical low-field mobility model. (Defaults: see Table 3.3)

G.surface

Real number parameter of the reduction factor for the surface mobility with respect to the bulk one (g_{surf} in Eq. (C.3)), unitless. (Default: 1.)

Hole

Logical parameter to indicate that the parameters specified in the present card apply to holes if their values are different for holes from electrons. These parameters include: **EC.Unv**, **ETa**, **UCons**, **UMAx**, and **UMIn** (Default: false, meaning that parameters apply to electrons)

Int.off

Real number parameter of d in Eq. (3.5) to specify the vertical offset from the Si/SiO$_2$ interface, in units of μ. (Default: 0.06)

Ncrit

Real number parameter of N_0 in Eq. (3.6) for the analytical low-field mobility model. (Defaults: see Table 3.3).

Qss.conc

Real number parameter of c in Eq. (3.7) to specify the conversion factor for converting q_{ss} into equivalent impurity concentration for mobility reduction, in units of cm^{-1}. (Default: 1.0×10^{7})

UMAx, UMIn

Real number parameters of μ_{max} and μ_{min} in Eq. (3.6) for analytical low-field mobility model. (Defaults: see Table 3.3)

UConst

Real number parameter of the low-field mobility for general semiconductor unknown to the program, in units of cm^2 V $^{-1}$ s^{-1}. (No default)

U0.unv

Real number parameter of μ_0 in Eq. (3.8) for Intel's surface mobility model, in units of cm^2 V $^{-1}$ s^{-1}. (Defaults: 783.5 for electrons and 247.4 for holes)

Vsat

Real number parameter for carrier saturation velocity. (Defaults: see Table 3.9)

EXAMPLES

```
MODELS conmob intelmob print
MOBILITY ec.perp=8e4 char.int=0.04
```

Using Intel's structural mobility model for transverse field mobility reduction, with parameter of the critical field 8×10^{4} V/cm (default value:

4.2×10^4) in Eq. (3.15) and characteristic length 0.04μ (default value: 0.03) in Eq. (3.5).

COMMAND MODELS

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

SYNTAX

MODels `<model flags> <numerical parameters>`

`model flags:`

{ **SRH** = <logical> **|** **CONSrh** = <logical> }
AUger = <logical> **BGn** = <logical>
CONMob = <logical>
{ **ANalytic** = <logical> **|** **ARora** = <logical> **|**
CCsmob = <logical> **|** **User1** = <logical> }
FLDmob = <logical> **|**
{ **ENgy.mob =** <logical> **|** **FMob.new** = <logical> **)**
Hypertang = <logical> }
TFldmob = <logical>
{ **SRFmob** = <logical> **|** **INTELMOB** = <logical>
INTELMOB.par = <logical> **|** **STrfld** = <logical>
Lombardi = <logical> **)** **|** **OLdtfld** = <logical> }
{ **IMPAct** = <logical> **| IMP.Jt** = <logical> **|**
IMP.NT = <logical> **| IMP.NVt** = <logical> }
IMP.TN = <logical> **IMP.TP** = <logical>
{ **BOltzmann** = <logical> **|** **FErmidirac** = <logical> }
INComplete = <logical> **PHotogen** = <logical>
PRint = <logical>

`numerical parameters:`

> **TEmperature** = <real>   **B.Electrons** = <real>
> **B.Holes** = <real>   **E0**=<real>
> **FLUx** = <real>   **ABs.coef** = <real>
> **ACc.sf** = <real>   **INV.sf** = <real>
> **OX.Left** = <real>   **OX.Right** = <real>
> **OX.Bottom** = <real>   **TAU.WN** = <real>
> **TAU.WP** = <real>   **II.AN**=<real>
> **II.BN** = <real>   **II.CN**=<real>
> **II.AP** = <real>   **II.BP** = <real>   **II.CP**=<real>

PARAMETERS

ABs.coef

The optical absorption coefficient (α in Eq. (3.43), Section 3.3.2) in units of cm^{-1}, used in photo-generation (flag **PHotogen**) modeling. (Default: 0.0)

ACc.sf

The low-field mobility reduction factor for surface accumulation layers, used in conjunction with the non-local, transverse-field mobility model **TFldmob** (refer to [13] for details). (Default: 0.87)

ANalytic

A logical flag specifying an analytical doping and lattice-temperature dependent mobility model for silicon only (see Section 3.1.1.3). (Default: false)

ARora

A logical flag specifying an alternative doping and lattice-temperature dependent mobility model originally developed by Arora [5] for silicon and now is available for both silicon and GaAs [6] (see Section 3.1.1.4 for details). (Default: false)

AUger

A logical flag to specify the use of Auger recombination mechanism (Eq. (3.47)). (Default: false)

B.Electrons, B.Holes

Real number parameters used in the longitudinal field-dependent mobility reduction for silicon (β in Eq. (3.30)). (Defaults: **B.Electrons**, 1.395; **B.Holes**, 1.215)

BGn

A logical flag for including band-gap narrowing mechanism. (Default: false)

BOltzmann

A logical flag for using Boltzmann carrier statistics. (Default: true)

CCsmob

A logical flag to invoke the mobility model of Dorkel and Leturcq for silicon [7], which includes carrier-carrier scattering effect, doping and lattice-temperature dependence (see Section 3.1.1.6). (Default: false)

CONMob

A logical flag to specify use of doping-dependent mobility. The default model is a table look-up for data in both silicon and GaAs at 300K. When used with other flags for doping dependent mobility such as **ANalytic** and **CCsmob**, the default model is overritten. (Default: false)

CONSrh

A logical flag to specify use of Shockley-Read-Hall recombination with doping-dependent carrier lifetimes (Eq. (3.45)). (Default: false).

ENgy.mob

A logical flag for using the carrier-temperature dependent mobility model as described in Section 3.1.3.3. Currently it applies to silicon only. (Default: false)

E0

A real number parameter used in the (longitudinal) field-dependent mobility model for GaAs (E_0 in Eq. (3.31), Section 3.1.3.2). (Default: 4×10^3 V/cm)

FErmidirac

A logical flag for using Fermi-Dirac carrier statistics. (Default: false)

FLDmob

A logical flag for considering the longitudinal field-dependent mobility reduction. The default model is Caughey-Thomas formulation [14] (Eq. (3.30)) for silicon and Thim's model (with negative differential mobility) [15] (Eq. (3.31)) for GaAs. (Default: false)

FLUx

The incident photon flux at the y=0 surface (F_{ph} in Eq. (3.43), Section 3.3.2) in units of photons/cm^2. (Real number, default: 0.0)

FMob.new

A logical flag for use of a longitudinal field-dependent mobility model which is derived from the carrier-temperature dependent model (flag

ENgy.mob) by field-temperature (of carriers) mapping. See Eq. (3.35) for the formulation. It applies to silicon only. (Default: false)

Hypertang

A logical flag to use hyperbolic tangent form for electron mobility in GaAs (Eq. (3.32)). This model avoids the negative differential mobility as exhibited in Thim's model [15], thus improving the numerical stability. (Default: false)

II.AN, II.BN, II.CN; II.AP, II.BP, II.CP

Real number parameters used in the carrier-temperature dependent impact ionization model (Eq. (3.42), flagged by **IMP.NT**). (For defaults see Table 3.13)

IIV.AN, IIV.BN; IIV.AP, IIV.BP

Real number parameters used in carrier-temperature dependent impact ionization model (Eq. (3.41), flagged by **IMP.NVt**). (For defaults see Table 3.12)

IMPAct

A logical flag to include the impact ionization as part of the carrier generation term in the solution process. The default impact ionization model is the local field-dependent one (Eqs. (3.37)-(3.38)), and the model parameters can be changed through **IMPACT** card. This default model can be overwritten by one of the following flags: **IMP.Engy** (**IMP.NT**), **IMP.NVt**, and **IMP.JT**. (Default: false)

IMP.Engy

A logical flag to indicate that the carrier energy (i.e. temperature) dependent impact ionization model is to be used as opposed to the local field dependent model (default of flag **IMPAct**). There are several carrier-energy dependent models available (see Section 3.3.1.2 for details) and the default one is invoked by flag **IMP.NT**. (Default: false)

IMP.Jt

A logical flag for using carrier-temperature dependent ionization rate (Eq. (3.39)) in the impact ionization model as described in Eqs. (3.37)-(3.38). (Default: false)

IMPJt.ratio

Real number parameter, γ, in Eq. (3.39). It can be used to change the energy relaxation length in the mapping between electric field and carrier temperature to fit the experimental data. (Default: 1.0)

IMP.NT

A logical flag for using carrier-temperature dependent ionization model as described in Eq. (3.42). The unique feature of this model is that the carrier drift velocity (or equivalently, the current density) does not play any role in the generation rate due to the impact ionization. (Default: false)

IMP.NVt

A logical flag for using carrier-temperature dependent impact ionization model in which the carrier (drift) velocity is approximated by the saturation velocity (Eq. (3.40)). (Default: false)

IMP.TN

A logical flag to indicate that carrier-temperature dependent impact ionization model is to apply to electrons, whereas the impact ionization model for holes can be either local-field (no flag **IMP.TP**) or temperature (flag **IMP.TP** set) dependent. (Default: false)

IMP.TP

A logical flag to indicate that carrier-temperature dependent impact ionization model is to apply to holes, whereas the impact ionization model for electrons can be either local-field (no flag **IMP.TN**) or temperature (flag **IMP.TN** set) dependent. (Default: false)

INComplete

A logical flag to Indicate that incomplete-ionization of impurities should be accounted for (Eq. (C.2)). (Default: false)

INV.sf

The low-field mobility reduction factor for surface inversion layer, used in conjunction with the non-local, transverse-field mobility model **TFldmob** (refer to [13] for details). (Default: 0.75)

INTELMOB, INTELMOB.par

Logical flags for using Intel's local field dependent mobility models (Eqs. (3.15)-(3.16)). Mobility reduction due to both transverse and longitudinal fields are taken into consideration simultaneously. The difference between these two models is in the expression for transverse field reduction. (Default: false)

Lombardi

A logical flag to specify use of Lombardi local transverse filed dependent mobility model (Eq. (3.17), also see [8]). (Default: false)

OLdtfld

A logical flag to specify use of the earlier version of UT Austin's non-local transverse field dependent mobility model (Eq. (3.24), see also [11]). (Default: false)

OX.Left, OX.Right, OX.Bottom

Real number parameters to define the location of the gate oxide region (left, right, and bottom edges) in conjunction with the use of UT Austin's non-local mobility models (flags **OLdfld** and **TFldmob**). (No default values but initialized to 0.0)

PHotogen

A logical flag to specify the use of external photo-generation as part of the carrier generation term. Two model parameters: **FLUx** and **ABs.coef** must also be specified to use this model (Eq. (3.43)). (Default: false)

PRint

A logical flag to print the status of all models and a variety of coefficients and constants. (Default: false)

SRFmob

A logical flag to invoke Watt's surface mobility model (Eqs. (3.22)-(3.23)). (Default: false)

SRH

A logical flag to specify the Shockley-Read-Hall recombination (Eq. (3.44)) with constant carrier lifetimes. (Default: true)

STrfld

A logical flag to specify that the surface transverse field is evaluated by the structure information (Eq. (3.21)) rather by definition (Eq. (3.20). This flag is usually used together with one of the transverse field dependent mobility models such as **INTELMOB**. (Default: false)

TAU.WN, **TAU.WP**

Real number parameters of carrier energy relaxation times for electrons and holes (Eq. (3.34) and Table 3.9), in units of ps (pico-second). (Defaults: tau.wn=0.42 and tau.wp=0.25)

TEmperature

Specifies the environment temperature, in units of Kelvin. (Default: 300)

TFldmob

A logical flag to invoke the second version (as opposed to flag **OLdfld**) of UT Austin's non-local transverse field mobility model (Eq. (3.28), see also [13]). It is based on an extended version of the Schwarz-Russek formulation [12]. (Default: false)

User1

A logical flag to specify Arora's (**ARora**) doping and lattice temperature dependent mobility model but with the user-definable parameters (refer to Section 3.1.1.5). (Default: false)

EXAMPLES

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

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 ENGY.MOB IMP.TN IMP.JT
+    IMPJ.Rat=0.81
```

Selects concentration-dependent mobility, SRH recombination and a carrier temperature-dependent mobility reduction model. An impact ionization model IMP.JT for the electron system is specified. The energy relaxation time ratio in computing impact ionization rate is set to be 0.81.

COMMAND OPTIONS

The **OPTIONS** card sets options for an entire run.

SYNTAX

Options `<run control> <plot control>`

`run control:`

G.debug or **Debug** = <logical> **N.debug** = <logical>
CPUStat = <logical> **CPUFile** = <character>

`plot control:`

PLOTDevice or **PLOTTer** or **Terminal**= <character>
PLOTFile = <character>
X.Screen = <real> **Y.Screen**=<real>
X.Offset = <real> **Y.Offset**=<real>

PARAMETERS

CPUFile

Character string for the name of the file to log the CPU statistics during `PISCES` run. (Default: null)

CPUStat

A logical flag to indicate that CPU statistics during PISCES run is to be outputted to the file specified by **CPUFile**. (Default: false)

G.debug (Debug), N.debug

Logical flags to print debugging information to the standard output. **G.debug** (or **Debug**) prints general information, while **N.debug** outputs more specifically numerical parameters. (Defaults: false)

PLOTDevice (PLOTTer, Terminal)

A character string to specify the output plot device. If no device is given, a default (usually the user's graphics terminal) will be used. `PISCES` uses the PLOTCAP graphics package from Stanford for plotting purpose. Refer to the PLOTCAP document for further details. The full set of supported devices is contained in the PLOTCAP data base (an ASCII file). Table M.4 list many of the common possibilities.

Table M.4 Possibilites for supported PLOTCAP devices.

hp2648	hp2623	hp9873	tek4107
xterm	sunview	vt240	tek4010
ditroff	PSfile	tplot	gplot

If PSfile is used, a PostScript file named print.ps will be generated for the plot and can later be printed to obtain a hard copy using laser printers. Plots will be scaled to the size of the specific device. Also note that 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. (Default: login terminal)

PLOTFile

A character string for name of the output file which is generally defined by the plot device. For example, a graphics terminal will use the terminal as the output file. Printers may have the output file be a spooler. The graphics output file can explicitly be set by the **PLOTFile** command. All graphics output will then be routed to the given file. Note that the contents of the file will be in a format specific to the given device. (Default: device specific)

X.Offset, **Y.Offset**

Real number parameters, in units of inches, to indicate the *x*- and *y*-offsets from the bottom-left corner of the screen, respectively. (Defaults: 0.0)

X.Screen, **Y.Screen**

Real number parameters, in units of inches, for the physical width and height of the screen, respectively. They are set automatically, depending on the plot device. They, however, can be altered for special effects (split screen plots, for instance). (Defaults: device's size)

EXAMPLES

```
OPTIONS PLOTDEV=tek4107 X.S=6 Y.S=5 X.Off=1
+    Y.OFF=0.5 CPUSTAT
```

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

```
OPTIONS PLOTDEV=lw PLOTFILE=plot.ps
```

Here we set the plot device to the LaserWriter and save the output in a file called plot.ps.

COMMAND PLOT.1D

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 (mode B). 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, terminal characteristics can be plotted against each other by choosing the value to be plotted on each axis.

SYNTAX

PLOT.1d `<segment definition> <plotted quantity>`
`+    <control>`

`segment definition:`

X.Start or **A.X** = <logical>   **Y.Start** or **A.Y** = <logical>
X.End or **B.X** = <logical>   **Y.End** or **B.Y** = <logical>

`plotted quantity:`

{ **POTential** = <logical> **|** **QFN** = <logical> **|**
QFP = <logical> **|** **DOping** = <logical> **|**
ELectrons = <logical> **|** **Holes** = <logical> **|**
NET.CHarge = <logical> **|** **NET.CArrier** = <logical> **|**
J.Conduc = <logical> **|** **J.Electr** = <logical> **|**
J.Hole = <logical> **|** **J.Displa** = <logical> **|**
J.Total = <logical> **|** **E.field** = <logical> **|**
Recomb = <logical> **|** **BAND.Val** = <logical> **|**
BAND.Con = <logical> **|** **VAcuum** = <logical> **|**
VELO.Ele = <logical> **|** **VELO.Hol** = <logical> **|**
XMole = <logical> **|** **TEMP.Ele** = <logical> **|**

TEMP.Hol = <logical> **|** **TEMP.Lat** = <logical> **|**
GEN.Elec = <logical> **|** **GEN.Hole** = <logical> **|**
IMpact or **II.gener** = <logical> }

or

X.Axis = <character> **Y.Axis** = <character>
INFile = <filename> **DAta** = <logical>

`control:`

LOgarithm or **Y.Log** = <logical> **X.Log** = <logical>
ABsolute = <logical> **X.ABsolute** = <logical>
NO.Clear = <logical> **NO.Axis** = <logical>
AScii = <logical> **Unchanged** = <logical>
INTegral = <logical> **NEGative** = <logical>
NO.Order = <logical> **POInts** = <logical>
PAuse = <logical> **LIne.type** = <integer>
MIn.value = <real> **MAx.value** = <real>
X.MAx = <real> **X.MIn** = <real>
X.Component = <logical> **Y.Component** = <logical>
Spline = <logical> **NSpline** = <logical>
OUTFile = <filename>

PARAMETERS

ABsolute, X.ABsolute

Logical flags to indicate use of absolute values for quantities to be plotted in *y*- and *x*-axes, respectively. (Defaults: false)

AScii

Specifies that the output file will have an ascii format (the default binary format is for use with the Stanford dplot system). (Default: false)

BAND.Val

A logical flag for plotting valence-band potential. (Default: false)

BAND.Con

A logical flag for plotting conduction-band potential. (Default: false)

DAta

Logical flag to indicate an HP measured I-V file will be used as an input file for plotting. (Default: false)

DOping

A logical flag for plotting doping. (Default: false)

E.field

A logical flag for plotting electric field. (Default: false)

ELectrons

A logical flag for plotting electron concentration. (Default: false)

GEN.Elec, GEN.Hole

Logical flags to plot the generation rates due to impact ionization caused by electron and hole currents, respectively. (Defaults: false)

Holes

A logical flag for plotting hole concentration. (Default: false)

IMpact (II.gener)

A logical flag for generation rate due to impact ionization (G in Eq. (3.37)). (Default: false)

INFile

Character string for name of the input file in which the solution is stored. (Default: null)

INTegral

Plots the integral of the specified ordinate. (Default: false)

J.Conduc, J.Electr, J.Hole, J.Displa, J.Total

Logical flags to plot conductive, electron, hole, displacement, and total currents, respectively. (Defaults: false)

LIne.type

Specifies the line type for the plotted curve. (Default: 1)

MIn.value, MAx.value

Specify minimum and maximum values, respectively, for the ordinate of the graph. Default: found automatically from the data to be plotted.

NEGative

Negates the ordinate values. `PISCES-II` by default will order the plot coordinates by abscissa value; this ordering will result in unusual plots for IV curves with negative resistance, for example. (Default: false)

NET.CArrier

Plot net carrier concentration. (Default: false)

NET.CHarge

Plot net charge concentration. (Default: false)

NO.Axis

Indicates that the axes for the graph are not to be plotted. (Default: false)

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. (Default: false)

NO.Order

A logical flag to force `PISCES` to plot the data points as they naturally appear. (Default: false)

Outfile

Character string for name of an output file in which the data points plotted are to be put. If specified, the graphics output will be directed to that file. For further discussion, see the **Options** card. (Default: from **Options** card)

PAuse

Causes `PISCES-II` to stop at the end of the plot so that a hard copy may be made before continuing. Execution can be resumed by hitting a carriage return. (Default: false)

POInts

Marks the data points on the plotted curve. (Default: false)

POTential

Plot mid-gap potential. (Default: false)

QFN, QFP

Logical flags to plot electron and hole quasi-fermi levels, respectively. (Defaults: false)

Recomb

Plot net recombination. (Default: false)

Spline, **NSpline**

The **Spline** option indicates that spline-smoothing should be performed on the data using **NSpline** interpolated points (maximum is 500). (Default: **Spline**, false; **NSpline**, 100)

TEMP.Ele, **TEMP.Hol**, **TEMP.Lat**

Logical flags to plot electron, hole, and lattice temperatures, respectively. (Defaults: false)

Unchanged

A synonym for **NO.Axis** and **NO.Clear**, but additionally it forces the use of the previous axis bounds so that a number of curves can easily be put on the same axis. (Default: false)

VAcuum

Logical flag to plot the vacuum energy level, a useful quantity for simulation of heterostructures because it is always continuous even at the abrupt material interface. (Default: false)

VELO.Ele, **VELO.Hol**

Logical flags to plot the electron and hole velocities, respectively. (Defaults: false)

X.AXis, **Y.Axis**

Character strings to indicate which quantities are to be used as *x*- and *y*-axes, respectively.

In mode B, terminal characteristics can be plotted against each other by choosing the value to be plotted on each axis. Quantities available for plotting include applied biases (VA1, VA2,..., VA9, VA0), actual contact bias which may differ from applied bias in the case of lumped element boundary conditions (V1, V2, etc.), terminal current (I1, I2, etc.), AC capacitances (C11, C12, C21, etc.), AC conductance (G11, G12, G21, etc.) and the magnitude of AC admittance (Y11, Y12, Y21, etc.). Additionally, any of the voltages or currents can be plotted versus time for transient simulations, and any AC quantity can be plotted versus frequency. The values plotted are the I-V or AC data of the present run, provided a log is being kept (see the **LOG** card). (Defaults: null)

X.Component, **Y.Component**

Logical flags to force the *x* or *y* components, respectively, of any vector quantities to be plotted, as opposed to the default magnitude. (Defaults: false)

X.Log, **Y.Log** (**LOgarithm**)

Logical flags for using logarithmic values of the quantities for plotting in *x*- and *y*-axes, respectively.

For rapidly varying quantities, the use of logarithmic values is often more revealing. Since many of the quantities may become negative, `PISCES` actually uses the following expression

$$\log(x) = \text{sign}(x) \cdot \log_{10}(1 + |x|) \tag{M.3}$$

to avoid overflow. To get the true logarithm of a quantity, specify **ABsolute** and **X.Log / Y.Log** - the absolute is taken first and there is no danger of negative arguments. (Default: false)

X.MAx, **X.MIn**

Real number parameters to specify the maximum and minimum values for the abscissa to be plotted. (Defaults: the maximum and minimum abscissa values in the data to be plotted)

XMole

A logical flag to plot the mole fraction of a compound material. For quaternaries, only x nor y mole fraction is plotted. (Default: false)

X.Start, **X.End**, **Y.Start**, **Y.End** (**A.X**, **B.X**, **A.Y**, **B.Y**)

Real number parameters to define the Cartesian coordinates of the start (**X.Start**, **Y.Start** or **A.X**, **A.Y**) and end (**X.End**, **Y.End** or **B.X**, **B.Y**) of a 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 and it is required in mode A.)

EXAMPLES

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

Plots a graph of potential along a straight line from (0.0,0.0) to (5.0,0.0).

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

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 X.AXIS=V2 Y.AXIS=I1
PLOT.1D X.AXIS=V2 Y.AXIS=I1 INF=logf0 UNCH
```

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=V3 Y.AXIS=VA3 OUTFILE=save.plot
```

Plots the actual contact voltage on a contact versus the applied voltage.

```
PLOT.1D INFIL=FILE.AC X.AXIS=FREQ Y.AXIS=C21
+    X.LOG
```

```
PLOT.1D INFIL=FILE.AC X.AXIS=FREQ Y.AXIS=C31
+    X.LOG UNCH LINE=4
```

This shows a plot of two capacitance components versus frequency saved in an AC log file from either a previous or present run. A different line type is chosen for the second component.

COMMAND PLOT.2D

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

SYNTAX

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

`+    <control>`

`area definition:`

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

`plotted quantity:`

Grid or **Mesh** = <logical> **Crosses** = <logical>
Boundary = <logical> **Depl.edge** = <logical>
Junction = <logical>

`control:`

NO.TIc = <logical> **NO.TOp** = <logical>
NO.Fill = <logical> **NO.Clear** = <logical>
NO.Diag = <logical> **LAbels** = <logical>
Flip.x = <logical> **Pause** = <logical>
L.Elect = <integer> **L.Deple** = <integer>
L.Junct = <integer> **L.Bound** = <integer>
L.Grid = <integer> **Outfile** = <filename>

PARAMETERS

Crosses

A logical flag to plot crosses at the locations of grid points. (Default: false)

Boundary

A logical flag to indicate that the boundaries around the device and between regions are to be plotted. (Default: false)

Depl.edge

A logical flag to indicate that depletion edges are to be plotted. Note that depletion edges can only be plotted when a solution is present. (Default: false)

Grid (Mesh)

A logical flag to plot the grid including lines which delineate elements. (Default: false)

Flip.x

A logical flag to flip the plot about the *y*-axis; i.e., it negates all *x* coordinates so that the plot is mirrored. (Default: false)

Junction

A logical flag to specify that the junctions from the doping profile are to be plotted. (Default: false)

LAbels

A logical flag to indicate that room is to be made for color contour labels on the right side of the plot device. (Default: false)

L.Elect, **L.Deple**, **L.Junct**, **L.Bound**, **L.Grid**

Integers to set line types for electrodes, depletion edges, junctions, region boundaries, and grids (i.e, mesh), respectively. (Defaults: 1)

NO.Clear

A logical flag to specify that the screen is not to be cleared before plotting. (Default: false)

NO.Diag

A logical flag to indicate not to plot the diagonals for a rectangle-based mesh. (Default: false)

NO.Fill

A logical flag to force the program 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. (Default: false)

NO.TIc

A logical flag to indicate that tic marks are not to be included around the plotted area. (Default: false)

NO.TOp

A logical flag to indicate that tic marks are not to be put on the top of the plotted region. (Default: false)

Outfile

Character string for name of an output file in which the data points plotted are to be put. If specified, the graphics output will be directed to that file. For further discussion, see the **Options** card. (Default: from **Options** card)

Pause

A logical flag causing the program to stop at the end of the plotting so that a hard copy may be made before continuing. Execution can be resumed by hitting a carriage return. (Default: false)

X.MIn, X.MAx, Y.MIn, Y.MAx

Real number parameters to define a rectangular area in the device to be plotted. (Defaults: a rectangle encircling the entire device)

EXAMPLES

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

Plots the entire grid to scale with tic marks.

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

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.

COMMAND PRINT

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

SYNTAX

PRint `<location> <quantity> <flags>`

`location:`

{ **X.MIn** = <real>   **X.MAx** = <real>
Y.MIn = <real>   **Y.MAx** = <real> }
|
{ **IX.Low** = <integer>   **IX.High** = <integer>
IY.Low = <integer>   **IY.High** = <integer> }

`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>

`flags:`

X.Compon = <logical>   **Y.Compon** = <logical>

PARAMETERS

Current

A logical flag to print currents (electron, hole, conduction, displacement, and total) at each grid for the present solution. (Default: false)

Elements

A logical flag to print information on triangular elements (sequence number, nodes, and material). (Default: false)

Geometry

A logical flag to print geometrical information (vertex coordinates) of triangles. (Default: false)

IX.Low, IX.High, IY.Low, IY.High

Integers for bounding indices (valid only for a rectangular mesh) to define an area in which the points of interest lie. (Defaults: a rectangle encircling the entire device)

Material

A logical flag to print material information (permittivity, band-gap, etc.) including the value of the doping dependent mobility and lifetime (if model flags specified) at each grid. (Default: false)

P.CURR1, P.CURR2

Logical flags to print currents (electron, hole, conduction, displacement, and total) at each grid for previous first and second solutions, respectively. (Defaults: false)

POints

A logical flag to print grid information (coordinates, doping, etc.). (Default: false)

P.QUE1, P.QUE2

Logical flags to print space charge, recombination rate, and electric field for the previous first and second solutions, respectively. (Defaults: false)

P.SOL1, P.SOL2

Logical flags to print *previous* first and second solutions (ψ, *n*, *p*, and quasi-Fermi potentials), respectively. (Defaults: false)

Que

A logical flag to print space charge, recombination rate, and electric field for the present solution. (Default: false)

Solution

A logical flag to print the present solution (ψ, *n*, *p*, and quasi-Fermi potentials). (Default: false)

X.Compon, Y.Compon

Logical flags to specify how any of the various vector quantities (current and field) should be printed. The default is the magnitude of the vector. **X.Compon** specifies that the magnitude of the *x*-component of the vector be printed, while **Y.Compon** specifies the *y*-component. Only one of these (or neither) can be specified on a single card. (Defaults: false)

X.MIn, X.MAx, Y.MIn, Y.MAx

Real number parameters for the physical coordinates, in units of μ, to define an area in which the points of interest lie. (Default: a rectangle encircling the entire device)

EXAMPLES

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

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

Solution information is printed for $0 < x < 1$ µm and $0 < y < 2$ µm.

COMMAND REGION

The **REGION** card assigns material type and its composition, i.e., mole fraction profile for a compound semiconductor, to a rectangular region bounded by high and low node numbers in *x*- and *y*-directions. Every triangular element must be given a certain type of material. If there is an overlap among region assignments, the latter one takes precedence. Currently, there are four types of compound semiconductors available in the code and they are Ge_xSi_{1-x}, $Al_xGa_{1-x}As$, $Al_xIn_{1-x}As$, and $Ga_xIn_{1-x}As_yP_{1-y}$. The composition profile can be either constant or linear.

SYNTAX

REGIon `<number> <position> <material> <mole spec>`

`number:`

NUmber = <integer>

`position:`

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

`material:`

SILicon = <logical> **|** **GAAs** = <logical> |
SIGe or **GEsi** = <logical> **|** **ALGaas** = <logical> |
ALInas = <logical> **|** **GAInasp** = <logical> |
SEmiconductor = <logical> **|** **Oxide** or **SIO2** = <logical>|
NItride or **SI3n4** = <logical> **|** **SApphire** = <logical> |
INsulator = <logical>

```
mole spec:
```

XMole = <real>   **YMole** = <real>
YLinear = <logical> **XInitial** = <real>
{ **XEnd** = <real> **|** **XSlope** = <real> }
YInitial = <real>
{ **YEnd** = <real> **|** **YSlope** = <real> }

PARAMETERS

ALGaas

A logical flag indicating an $Al_xGa_{1-x}As$ region. (Default: false)

ALInas

A logical flag indicating an $Al_xIn_{1-x}As$. (Default: false)

GAAs

A logical flag to indicate a gallium arsenide (GaAs) region. (Default: false)

GAInasp

A logical flag indicating a $Ga_xIn_{1-x}As_yP_{1-y}$ region. (Default: false)

INsulator

A logical flag to indicate a general insulator region. (Default: false)

IX.Low, IX.High, IY.Low, IY.High

These parameters are the indices of a box in the rectangular mesh. (Default: the entire device)

NItride or **SI3n4**

A logical flag to indicate a nitride (Si_3N_4) insulator region. (Default: false)

NUmber

This parameter selects the rectangular region in question. Currently in the code the maximum number of regions is set to 1,000, but can be changed under the user's request.

Oxide or **SIO2**

A logical flag to indicate a silicon dioxide (SiO_2) region. (Default: false)

SApphire

A logical flag to indicate a sapphire (Al_2O_3) region. (Default: false)

SEmiconductor

A logical flag to indicate a general semiconductor region. Forty material parameters can be defined by the user through **MATERIAL** card. (Default: false)

SIGe or **GEsi**

A logical flag indicating a Ge_xSi_{1-x} region. (Default: false)

SILicon

A logical flag to indicate a silicon (Si) region. (Default: true)

STrained

A logical flag to indicate that the material is a strained layer. Currently in the code, it is only applied to $Ga_xIn_{1-x}As_yP_{1-y}$ and only the bandgap and the effective mass for holes (both light and heavy ones) are affected by this flag (for details see Section 4.4.3). (Default: false)

XInitial, **XEnd**, **XSlope (YInitial**, **YEnd**, **YSlope)**

These real number parameters are used to specify the linear profile of mole fraction (*x* or *y*) in the region. Two combinations of parameters can be used to uniquely specify the profile. When **XInitial** and **XEnd** are used, the position-dependent mole fraction is computed as

$$\text{mole}\,(d) = \mathbf{XIinital} + \frac{\mathbf{XEnd} - \mathbf{XIinital}}{d_{\text{high}} - d_{\text{low}}} \cdot (d - d_{\text{low}}) \qquad \textbf{(M.4)}$$

where *d* is either *x* or *y* depending upon the logical flag **YLinear** (see **YLinear**) d_{low} and d_{high} are coordinates corresponding to **XInitial** and **XEnd**, respectively. Another possible combination is for **XInitial** to be used with **XSlope**, the mole fraction is then computed as

$$\text{mole}\,(d) = \mathbf{XIinital} + \mathbf{XSlope} \cdot (d - d_{\text{low}}) \qquad \textbf{(M.5)}$$

The same rules apply to *y* mole fraction.

XMole, **YMole**

These real number parameters are used to specify the mole fraction for compound semiconductors, be binary (Ge_xSi_{1-x}), ternary ($Al_xGa_{1-x}As$ and $Al_xIn_{1-x}As$), or quaternary ($Ga_xIn_{1-x}As_yP_{1-y}$). If the composition profile is flat (constant) in the region, the mole fraction is specified using **XMole** for *x* in binary/ternary and **XMole** and **YMole** for *x* and *y*, respectively, in quaternary.

YLinear

A logical flag to indicate that the linear composition profile changes along the y-direction instead of x-direction. (Default: false)

EXAMPLES

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

Defines a silicon region extending from nodes 1 to 25 in the *x* direction and nodes 4 to 20 in the *y* direction.

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.

```
REGION NUM=1 IX.L=1 IX.H=100 IY.L=1 IY.H=3
+    INSUL
REGION NUM=2 IX.L=1 IX.H=3 IY.L=1 IY.H=9
+    GAINASP XMOL=0.47 YMOL=1.0
REGION NUM=2 IX.L=98 IX.H=100 IY.L=1 IY.H=9
+    GAINASP XMOL=0.47 YMOL=1.0
REGION NUM=2 IX.L=3 IX.H=6 IY.L=1 IY.H=3
+    GAINASP XMOL=0.47 YMOL=1.0
REGION NUM=2 IX.L=41 IX.H=98 IY.L=1 IY.H=3
+    GAINASP XMOL=0.47 YMOL=1.0
```

Defines an oxide layer partially overwritten by the source and drain regions consisting of $Ga_{0.47}In_{0.53}As$ (*y* is set to 1.0 such that quaternary GaInAsP becomes ternary GaInAs).

```
REGION NUM=2 IX.L=1 IX.H=40 IY.L=18 IY.H=24
+    ALGAAS XINI=0.27 XEND=0.0
```

```
REGION NUM=3 IX.L=1 IX.H=40 IY.L=24 IY.H=27
+    ALGAAS XMOLE=0.0
```

This example shows the specification of linear grade of AlAs composition in $Al_xGa_{1-x}As$ ternary along the *y* direction in region 2.

COMMAND REGRID

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. The default regrid is to refine all regions for potential and electric field.

SYNTAX

REGRid `<location> <plotted quantity>`
`+     <control> <files>`

`location:`

{ **X.MIn** = <real> **X.MAx** = <real>
Y.MIn = <real> **Y.MAx** = <real> } |
Region = <integer> **|** **Ignore** = <integer>

`plotted quantity:`

Potential = <logical> **|** **EL.field** = <logical> **|**
QFN = <logical> **|** **QFP** = <logical> **|**
DOPIng = <logical> **|** **ELEctrons** = <logical> **|**
Holes = <logical> **|** **NET.CHarge** = <logical> **|**
NET.CArr = <logical> **|** **Min.carr** = <logical>

`control:`

STep or **RAtio** = <real> **CHange** = <logical>
ABsolute = <logical> **LOGarithm** = <logical>
LOCaldop = <integer> **MAx.level** = <integer>
SMooth.key =< logical> **COs.angle**=<real>

```
files:
```

Outfile = <filename>   **DOPFile**=<filename>
AScii = <logical>

PARAMETERS

ABsolute

A logical flag to specify that the absolute value of the quantity is to be used. (Default: false)

AScii

A logical flag to indicate that all mesh files and triangle trees (not including **DOPFile**) for this card should be done in ASCII rather than binary (a default). (Default: false)

CHange

A logical flag to determine whether to use the magnitude (false) or the difference (true) of a variable in a triangle as the criterion of refinement. (Default: true)

COs.angle

A real number for the cosine of the angle, which defines the “obtuse criterion" to limit the creation of obtuse angles in the mesh. If regrid would create a triangle with an angle whose cosine (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** flag in this card. It can be turned off everywhere by using a value of **COs.angle** greater than 1.0. The default is to turn it off everywhere. (Default: 2.0)

DOPFile

Character string for the name of the binary mesh file, which contains the doping for the device (see **DOPING** card). If set to a non-null string, interpolation of doping values at any newly created grid points to re-dope the structure based on the initial doping specification (a default action) will not occur. (Default: null)

DOPIng

A logical flag to indicate regriding based on doping concentration (in units of cm^{-3}). (Default: false)

ELEctrons

A logical flag to indicate regriding based on electron concentration (in units of cm^{-3}). (Default: false)

EL.field

A logical flag to indicate regriding based on electric filed (in units of V/cm). (Default: false)

Holes

A logical flag to indicate regriding based on hole concentration (in units of cm^{-3}). (Default: false)

Ignore

An integer number for region(s). In these regions, regriding will not happen (i.e., ignored), nor will be smoothed after regriding. This flag is similar to **REGion** in function, but opposite in effect. (Default: 0, i.e., none ignored)

LOCaldop

A logical flag related to the minority carrier regrids. If set, when the minority carrier concentration exceeds the local doping, the grid will be refined. (Default: false)

LOGarithm

A logical flag to use logarithmic instead of the actual value of the quantity concerned. If set, parameter **STep** (**RAtio**) is interpreted in the logarithm. (Default: false.)

MAx.level

An integer for 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. Values less than or equal to zero are interpreted relative to the current maximum level. (Default: dynamic)

MIn.carr

A logical flag to indicate regriding based on minority carrier concentration (in units of cm^{-3}). (Default: false)

NET.CHarge

A logical flag to indicate regriding based on the net charge density (in units of cm^{-3}). (Default: false)

NET.CArr

A logical flag to indicate regriding based on net carrier concentration (in units of cm^{-3}). (Default: false)

Outfile

Character string for the binary output mesh file, which is necessary if the mesh is to be used for subsequent runs. A history of the triangle tree is always generated to assist further regriding steps and its name is the one specified by **Outfile** and concatenated by the letters “tt” to the end. In the case that this file is used as an input mesh file for a PISCES run including a regrid action, the program will look for a file with the same name as the input mesh file plus “tt” at the end. (Default: null)

Potential

A logical flag to indicate regriding based on electrostatics potential which usually refers to the intrinsic Fermi level (in units of V). (Default: false)

QFN, QFP

Logical flags to indicate regriding based on electron and hole quasi-Fermi levels, (in units of cm^{-3}), respectively. (Defaults: false)

Region

An integer for region(s) in which mesh is to be refined according to the user criterion. Note that other regions might be refined as well as a side effect to maintain well-shaped triangles. (Default: all regions)

SMooth.key

An integer parameter which has the same meaning as **SMooth.key** in **MEsh** card. See **MEsh** card for details. (Default: 0)

STep (RAtio)

Real number parameter as the numerical criterion for refining a triangle. It is interpreted in logarithm if flag **LOGarithm** is set. (No default)

X.MIn, X.MAx, Y.MIn, Y.MAx

Real number parameters for the bounds of the region for mesh refinement; Only those triangles which have nodes falling inside the region are considered for refinement. (Defaults: a rectangle encircling the entire device).

EXAMPLES

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

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

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

A similar effect is obtained with just one **REGRID** statement. 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.

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

This time we perform an initial solution and refine triangles which exhibit large potential steps.

COMMAND SOLVE

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

SYNTAX

SOLve `<initial guess> <dc bias> <transient>`
`+    <ac> <files>`

`initial guess:`

{ **INitial** = <logical> **|** **PREvious** = <logical> **|**
EXtrapolate = <logical> **|** **PROject** = <logical> **|**
LOCal = <logical> }

`dc bias:`

V1 = <real> **I1** = <real> **T1** = <real>
V2 = <real> **I2** = <real> **T2** = <real>
. . .
V9 = <real> **I9** = <real> **T9** = <real>
V0 = <real> **I0** = <real> **T0** = <real>
ELectrode = <integer> **VSTep** = <real>
IStep = <real> **NSteps** = <integer>
N.bias = <real> **P.bias** = <real>

`transient:`

Dt or **TSTEp** = <real> **TSTOp** or **TFinal** = <real>
Ramptime = <real> **ENdramp** = <real>

```
ac:
```

{{ **AC.analysis** = <logical>   **FRequency** = <real>
SOr = <logical>   **Qmr** = <logical>
FStep = <real>   **MUlt.freq** = <logical>
NFstep = <integer>   **VSS** = <real>
S.omega = <real>   **MAx.inner** = <integer>
TOlerance = <real>
} **|** **LOWfreq.ac** = <logical>
} **TErminal** = <integer>

```
files:
```

Outfile = <filename>   **AScii** = <logical>
Currents = <logical>   **BAnd** = <logical>

PARAMETERS

AC.analysis

A logical flag to indicate that AC sinusoidal small-signal analysis be performed after the DC condition is solved for. Note that the full Newton method (2 carriers) must be used for this analysis. Use of flag **G.Debug** in the **Options** card will provide some detailed information on the AC solution procedure. (Default: false)

AScii

A logical flag to indicate that the solution file saved in **Outfile** will be ASCII as opposed to binary (a default). (Default: false)

BAnd

A logical flag to include the band profile (vacuum level, conduction and valence band edges) in the output solution file in addition to potential, carrier concentrations, etc. (Default: false)

Currents

A logical flag to indicate that the electron, hole, and displacement currents, and the electric field, will be computed and stored with the solution. (Default: false)

Dt or TSTEp

Real number for the time-step to be taken during time transient analysis. For automatic time-step runs (flag **TAuto** in **MEthod** card), it is used to select the first time step only. (Default: 0.0)

ELectrode

An integer for the number of the electrode being stepped. 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. Note that if there are 10 electrodes, don't put electrode 0 first in the sequence!). (Default: 0)

EXtrapolate

A logical flag to indicate that the initial guess for the solution at the present bias is to be obtained by linear extrapolation of the previous two solutions which are either just obtained in the current PISCES run or loaded in using parameters **IN1file** and **IN2file** in **LOad** card. (Default: false)

FRequency

Real number parameter for the frequency, in units of Hz, at which AC analysis is to be performed. (No default)

FStep, **MUlt.freq**, **NFstep**

The AC analysis can be repeated at a number of different frequencies (without resolving the DC condition) by selecting a real number parameter **FStep**. **FStep** is a frequency increment which is *added* to the previous frequency by default, or it may be *multiplied* to by setting logical flag **MUlt.freq**. The number of increments is given by integer parameter **NFsteps**. (Defaults: **FStep**, 0.0; **MUlt.freq**, false; **NFsteps**, 0)

INitial

A logical flag to indicate the solution at the thermal equilibrium (i.e., zero bias) is to be sought. Note that the first bias point in any PISCES run must be set with this flag unless there is a solution loaded using **LOad** card. (Default: false)

IStep

Real number parameter for the current increment to be added to one or more electrodes, as specified by the integer assigned using parameter **ELectrode** in this card. (Default: 0.0)

I1, **I2**, **...** , **I9**, **I0**

Real number parameters for the terminal currents, in units of A μ^{-1}, applied to contacts 1, 2, ... , 9, 0. (Default: currents from the previous solution)

LOCal

A logical flag to use the carrier quasi-Fermi levels in the previous solution (**INFile** in **LOAd** card) as the initial guess. (Default: false)

LOWfreq.ac

A logical flag to use zero-frequency for AC analysis (see Section 5.1.4 for details). (Default: false)

MAx.inner

Integer parameter for the maximum number of SOR iterations. (Default: 25)

N.bias, P.bias

Real number parameters to set the levels for electron and hole quasi-Fermi potentials, respectively. These parameters only apply to those carriers which are not being solved for in the PISCES run (e.g. holes when only electrons are specified to solve in **Solve** card). If **N.bias** or **P.bias** is not specified, then program will either choose local quasi-Fermi potentials based on bias and doping (see Chapter 2 of [3]) or if **Fix.qf** is set on the **MEthod** card, set the quasi-Fermi levels where applicable to values which produce the least amount of free carriers (maximum bias for electrons and minimum bias for holes).

NSteps

The number of bias increments (steps) to be taken; i.e., if **VStep** (**IStep**) is specified, the specified electrode is incrementted **NSteps** times. (Default: 0.0)

Outfile

Character string for the name of the binary output file to save the solution at this bias point. 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 incrementted by one for each bias point in succession, resulting in a unique file for each bias point. (Default: null)

PREvious

A logical flag to specify that the solution at the previous bias point, either loaded in using **LOad** card or just solved, be used as the initial guess for the solution at the present bias point. (Default: true)

PROject

A logical flag to indicate Newton Projection method (NPM) is to be used in obtaining the initial guess to the solution at the present bias based on the previous solution. For details of NPM, see Section 5.4. (Default: false)

Qmr

A logical flag to use QMR (Quasi-Minimal Residual) method for AC analysis (see Section 5.1.3 for details) (Default: true)

Ramptime, **ENdramp**

Real number parameters to specify the time duration of a bias ramp and the ending instant of the ramp, respectively, for transient analysis, in units of seconds. Note that bias ramp must be linear. Specifically, is the ramp begins at $t = t_0$, then it ends either at $t = t_0 +$ **Ramptime** or at $t =$ **ENdramp**, depending on which parameter is used. (Defaults: 0.0)

S.omega

A real number parameter for the relaxation factor used in SOR method (ν in Eqs. (5.13)-(5.14)). Note that it is not a frequency. (Default: 1.0)

SOr

A logical flag to use SOR method for AC analysis (refer to Section 5.1.1. and [36] for details) (Default: false)

TErminal

An integer parameter for the sequence number of contact(s) to which the AC bias (i.e., stimulus) is to be applied. More than one contact number may be specified (via concatenation), but each will be solved separately. Each contact that is specified yields a column of the admittance matrix. (Default: all contacts)

TOlerance

Real number parameter used as the criterion for the iteration to terminate in AC analysis, unitless. (Default: 1×10^{-5})

TSTOp (**TFinal**)

Real number parameter to specify the end of the time interval to be simulated, in units of seconds. Thus the simulation begins at $t = t_0$, it will end at t = **TSTOp** (**TFinal**). Alternatively, **NSteps** can be used to signal the end of the interval; That is, the final time would be $t = t_0$ + **NSteps** x **TSTEp**. (Default: 0.0)

T1, **T2**, **...** , **T9**, **T0**

Real number parameters for the lattice temperatures at terminals (either thermal or electrical) if thermal Dirichlet boundary conditions are applied to the corresponding contacts. (Defaults: from previous solution)

VSS

Real number parameter for the magnitude of the applied small-signal bias (stimulus) (V_i in Eq. (2.19) of [4]). (Default: 0.1 x kT/q where T is the environment temperature)

VSTep

Real number parameter for the voltage increment to be added to one or more electrodes, as specified by the integer assigned to parameter **Electrode** in **Solve** card. (Default: 0.0)

V1, **V2**, **...** , **V9**, **V0**

Real number parameters for the bias voltages (for non-current boundaries only) applied at contacts 1, 2, ... , 9, 0. (Default: the potentials from the previous solution)

EXAMPLES

```
SOLVE INIT OUTF=OUT0
```

Performs an initial bias point, saving the solution to the data file OUT0.

```
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
```

In this 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-II` 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 V1=0 V2=0 V3=1 VSTEP=.5 NSTEPS=2 ELECT=23
SOLVE V2=2 V3=3
```

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-II` 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
SOLVE VSTEP=.5 NSTEPS=2 ELECT=23
SOLVE VSTEP=1 NSTEPS=1 ELECT=23
```

This repeats the previous example as a three-card sequence. If no new voltages are specified and a **VStep** is included, the first bias point solved for is the preceding one incrementted appropriately by **VStep**.

```
METHOD 2ND TAUTO AUTONR
SOLVE V1=1 V2=0
SOLVE V1=2 TSTART=1E-12 TSTOP=25E-9
+    RAMPTIME=10E-9 OUTF=UP1
SOLVE V1=-1 TSTOP=100E-9 RAMPTIME=20E-9
+    OUTF=DOWN1
SOLVE V1=-1 V2=0
```

This sequence is an example of a time-dependent solution. The **METHOD** card specifies the second-order discretization and automatic time-step selection option, along with Newton-Richardson. The first **SOLVE** card then 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 ramped to 2 volts over a period of 10ns and is left on until 25ns. Each solution is written to a file; the name of the file is incrementted in a manner similar to that described above for a dc simulation (UP1, UP2, etc.). Note that an initial time step had to be specified on this card. The third **SOLVE** card ramps V1 down from 2 volts to -1 volts in 20 ns (end of ramp is at t = 45ns). The device is then solved at this bias for another 55ns (out to 100ns). Note that again each solution is saved in a separate file (DOWN1, DOWN2, etc.) and that no initial time-step was required since one had been estimated from the last transient solution for the previous **SOLVE** card. Finally, the fourth **SOLVE** card performs the steady-state solution at V1 = -1 and V2 = 0.0

```
SOLVE V1=0 V2=0 V3=0 VSTEP=0.5 NSTEPS=4
+    ELECT=1 AC FREQ=1E6 FSTEP=10 MULT.F
+    NFSTEP=5 VSS=0.01
```

Here an AC example is presented. Assume the device to be simulated has 3 electrodes. Starting from solved DC conditions at V1 = 0, 0.5, 1.0, 1.5 and 2.0 volts, 10 mV AC signals of frequency 1 MHz, 10 MHz, 100 MHz, 1 GHz, 10 GHz and 100GHz are applied to each electrode in the device. Note that the number of AC solutions to be performed is 5 X 6 X 3 = 90.

```
SOLVE V1=0 V2=0 V3=0 VSTEP=0.5 NSTEPS=4
+    ELECT=1 lowf
```

Same as the above example but use zero-frequency analysis to extract quasi-static *y*-parameters.

COMMAND SPREAD

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 MOSFETs. The **SPREAD** card is somewhat complicated; it is suggested to follow the supplied examples very carefully (particularly the MOSFET example in Chapter 5 in [3]).

SYNTAX

SPread `<direction> <region> <specifics>`

`direction:`

{ **LEft** = <logical> **|** **Right** = <logical> }

`region:`

Width = <real>
Upper = <logical> **LOwer** = <logical>

`specifics:`

{ **Y.Lower** = <real> **|** **Thickness** = <real> }
Vol.ratio = <real> **Encroach** = <real>
GRAding = <real>
GR1 = <real> **GR2** = <real>
Middle = <real> **Y.Middle**=<real>

PARAMETERS

Encroach

A real number factor which defines the abruptness of the transition between a distorted and non-distorted mesh. The transition region becomes more abrupt with smaller **Encroach** factors (the minimum is 0.1). An important note: depending on the characteristics of the undistorted mesh, very bad triangles (long, thin, and obtuse) may result if **Encroach** is set too low. (Default: 1.0)

GRAding

A real number parameter to specify 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. (Default: 1.0)

GR1, GR2, Middle, Y.Middle

Real number parameters used as an 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**. (Defaults: **GR1**, 1.0; **GR2**, 1.0; and no defaults for **Middle** and **Y.Middle**)

LEft, Right

Logical flags to specify that the left and right sides of the grid, respectively, will be distorted. (Defaults: false)

Upper, LOwer

Integer parameters to specify the upper and lower *y*-grid lines, respectively, between which the distortion will take place. (No default)

Vol.ratio

Real number parameter as the ratio of the downward displacement of the lower grid line to the net increase in thickness. It is ignored if **Y.Lower** is specified. (Default: 0.44 which is derived from oxide/silicon growth/consumption ratio during oxidation such that the Si/SiO_2 interface can move correctly)

Width

A real number parameter to specify 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** and max[x] - **Width** for **Right**) will lie in the middle of the transition region between the distorted and undistorted grid regions, in units of μ. (No default)

Y.Lower and Thickness

Real number parameters to define the distorted grid region, in units of μ. Only one of them 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 and it will usually move the positions of both the **Upper** and **LOwer** grid lines unless **Vol.ratio** is set to 0 or 1. (No default)

EXAMPLES

```
$ *** 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
```

This spreads what was previously a uniform 400Å of oxide to 1000Å on the left side of the device. This will result in a net increase in thickness of 600Å of oxide. Because the default **Vol.ratio** is used, 0.44 X (600) = 264Å of the net increase will lie below the original 400Å and 0.56 X (600) = 336Å of the net increase will lie above the original 400Å. 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.

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

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).

COMMAND SYMBOLIC

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.

SYNTAX

SYmbolic `<solution method> <device equation>`
+ `<options>`

`solution method:`

{ **Newton** = <logical> | **Gummel** = <logical> }

`device equation:`

Carriers = <integer>
{ **Electrons** = <logical> | **Holes** = <logical> }
ENGY.Elect = <logical> | **ENGY.Hole** = <logical>
ENGY.Lat = <logical>

`options:`

Min.degree = <logical> **Strip** = <logical>
Infile = <filename> **Outfile** = <filename>
Print = <logical>

PARAMETERS

Carriers

An integer parameter to specify the number of types of carriers to be simulated. 0 for solving the Poisson’s equation only, 1 for solving either electrons or holes depending on flags **ELectrons** and **Holes**, and 2 for solving both electrons and holes**.** (Default: 1)

ELectrons, Holes

Logical flags to indicate which type of carriers is to be solved if **Carriers** is set to 1. (Defaults: false)

ENGY.Elect, ENGY.Hole

Logical flags to indicate if electron or hole or both carrier energy balance equation(s) is to be solved in the simulation. (Defaults: false)

ENGY.Lat

A logical flag to indicate that the lattice thermal diffusion equation is to be solved during the simulation. (Defaults: false)

Infile, Outfile

Character strings (up to 20 characters long) to specify the input and output (binary) file names, respectively, for the symbolic factorization. Note that these binary files can be quite large, so it is advised not to use this feature. (In some computing environments it is also faster to compute the symbolic information for each run than to read it from the file). (No default, strings set to null)

Min.degree

A logical flag to specify the use of a minimum degree ordering of the pivots for decomposition in order to reduce the size of the generated L and U matrices, hence to reduce the amount of CPU time spent in solving linear systems. This parameter is definitely recommended. (Default: true)

Newton, Gummel

Logical flags to specify the Newton simultaneous or Gummel sequential method in solving the Poisson’s, electron and hole continuity equations. In the case of solving the Poisson’s equation only (**Carriers** = 0), Gummel method is the default one. When one of or both carrier energy balance equations is to be solved, Newton method must be used for the current implementation. (Defaults: false)

Print

A logical flag to indicate that information about the memory allocated for the run is to be printed to the standard output file. (Default: false)

Strip

A logical flag to specify redundancy (zero couplings) be removed from the symbolic map, and is naturally on. (Default: true)

EXAMPLES

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

Specifies a symbolic factorization for a simulation with only holes and using the Gummel method (the symbolic factorization is saved in a file called SYMB.OUT).

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

A previously generated symbolic factorization is read in from a file called SYMB.IN. The method used is the full Newton method, and both carriers are included in the simulation. Additionally, the sizes and dimensions of all arrays associated with the sparse matrix solution are printed.

```
SYMBOLIC NEWTON CARR=2 engy.ele
```

Solve electron energy balance equation together with Poisson's and two carrier continuity equations for electron temperature and ψ, n, and p.

COMMAND TITLE

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

SYNTAX

Title

<character string>

PARAMETERS

None.

EXAMPLES

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

COMMAND VECTOR

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

SYNTAX

Vector `<plotted quantity> <control>`

`plotted quantity:`

{ **J.Conduc** = <logical> **|** **J.Electr** = <logical> **|**
J.Hole = <logical> **|** **J.Displa** = <logical> **|**
J.Total = <logical> **|** **E.field** = <logical> }

`control:`

LOgarithm = <logical> **MInimum** = <real>
MAximum = <real> **Scale** = <real>
Clipfact = <real> **LIne.type** = <integer>

PARAMETERS

Clipfact

A threshold below which vectors are not plotted. (Default: 0.1)

E.field

Plot the electric field. (Default: false)

J.Conduc, **J.Displa**, **J.Electr**, **J.Hole**, **J.Total**

Logical flags for plotting the conduction, displacement, electron, hole, and total current densities, respectively. (Defaults: false)

LIne.type

An integer parameter for the vector line type for plotting. (Default: 1, a solid line)

LOgarithm

A logical flag to specify use of logarithmic value of the magnitude. By default, all vectors in a linear plot are scaled by the maximum magnitude of the quantity of interest over the grid, while if **LOgarithm** is set, the default scaling uses the minimum (non-zero) magnitude. (Default: false)

MAximum, **MInimum**

Real number parameters for the minimum and maximum values of the quantities to be plotted. These values will be printed during the execution of a plot. With these parameters is possible to plot two bias conditions or two devices with the same scaling. (Defaults: 0.0)

Scale

A real number parameter used as the scale factor by which all magnitudes are multiplied. (Default: 1.0)

EXAMPLES

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

Plot electron and hole currents over a device.

COMMAND X.MESH, Y.MESH

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

SYNTAX

X.mesh or **Y.mesh** `<node> <location> <ratio>`

`node:`

Node = <integer>

`location:`

Location = <real>

`ratio:`

Ratio = <real>

PARAMETERS

Location

Real number parameter to specify where to locate the line, in units of µ. (No default)

Node

An integer parameter for the numbering of the line in the mesh. Lines should be assigned consecutively, beginning with the first and ending with the last. (No default)

Ratio

Real number parameter as the ratio used for the interpolation of lines between the given lines, unitless. The spacing grows/shrinks by ratio in each subinterval, and the ratio should usually lie between 0.667 and 1.5. (Default: 1.0)

EXAMPLES

```
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
```

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

PART II

Curve Tracer

Stephen G. Beebe, Zhiping Yu, Ronald J. G. Goossens,
and Robert W. Dutton

Acknowledgments

The Curve Tracer algorithm is based on and extends ideas previously developed by W.M. Coughran, Jr., M.R. Pinto, and T.J. Hughes. This project was supported by Semiconductor Research Corporation Contract Numbers, SRC 94-SJ-116 and SRC 93-MC-106. We would also like to acknowledge the mentorship of Dr. Ben Tseng of AMD as well as their support of both a Summer internship and ongoing access to both facilities and device technology during this period of study.

SECTION 7
Introduction

In Technology CAD, the use of software to simulate the testing of semiconductor devices is known as the Virtual Instrumentation. A virtual instrument should be able to automatically generate simulation data, e.g., *I-V* curve along a bias sweep, given only the simple specifications a user would input to a real programmable instrument testing an IC device. Numerical device simulators such as `PISCES` provide a means of creating virtual devices and simulating electrical tests on the devices. However, these simulators cannot trace through *I-V* curves with sharp turns unless the user carefully controls the bias conditions near these turns -- a tedious and time-consuming process. This deficiency prompted the creation of `Tracer`.

`Tracer` is a C program which automatically guides `PISCES` and other semiconductor device simulators through complex *I-V* traces and is ideally suited for device-failure phenomena such as latchup, BV_{CEO}, and electrostatic-discharge (ESD) protection. Given a `PISCES` input deck and a specification file with a PISCES-like syntax, a simulation can be run over any current or voltage range without the user's intervention. `Tracer` is limited to DC, one-dimensional traces, i.e., only one electrode can be swept per run. It sweeps the electrode by dynamically setting the most stable bias condition at each solution point. Additionally, `Tracer` has the ability to maintain zero-current bias conditions at one or two electrodes during the trace, even at low device-current levels where such bias conditions are unstable using traditional device simulation. The theory implemented in the `Tracer` program can be found in [1].

SECTION 8
Shell Command Line

Usage: `%tracer inputfile tracefile [outputfile]`

where

- `tracer` is the program name, i.e. it is the shell command name
- `inputfile` is the name of the `PISCES` input deck which defines the device structure to be simulated and specifies what physical models are to be used. Basically, it contains everything in a normal `PISCES` deck except the solve card specifications (SECTION 10).
- `tracefile` is a file containing instructions on how to conduct the tracing as well as specifying bias conditions for all electrodes (SECTION 9).
- `outputfile` is an optional specification of the name of the file where the simulation data is to be written (SECTION 11). If `outputfile` is not given, the name of the output file defaults to `inputfile`.out.

SECTION 9
Trace File

The trace specification file, `tracefile`, is similar to a `PISCES` or `SUPREM` input deck. Each line begins with a word designating what type of statement (or “card”) it is. The four available cards are **CONTROL**, **FIXED**, **OPTION**, and **SOLVE**. Also, a line may start with a “$” for comments. Such lines are ignored during program execution. The cards may appear in any order. A card may be continued on following lines by placing a “+” at the beginning of each subsequent line. The “+” should be separated from the parameters on the line by at least one space.

Each option in a card should have the following structure: “param = paramvalue”. Spaces separating the “=” sign are optional. The parameters for each card are described below. As with `PISCES` syntax, parameter names and values are not case-sensitive and may be abbreviated provided they remain unambiguous. Square brackets, [], enclosing a parameter indicate that it is optional (note that some of these parameters are optional only in the sense that they will default to a certain value if not specified in `tracefile`). A vertical line, |, represents a logical OR -- only one of a list of parameters separated by “|” signs can be specified.

All electrodes in the device must have representation in the `tracefile`. Each electrode must appear as one, and only one, of the following: the **CONTROL** electrode, a **FIXED** electrode, or an open contact (**OPENCONT1** or **OPENCONT2**) on the **SOLVE** card.

9.1 CONTROL Card

9.1.1 Description

The **CONTROL** card is used to designate the electrode which will be swept through the trace as well as the boundaries of the trace. This electrode is referred to as the control electrode. To define the start of the simulation range, an initial voltage and an initial voltage step must be specified for the control electrode. The end of the trace is specified by either a maximum electrode voltage, a maximum electrode current, or the total number of simulated points to be found.

9.1.2 Syntax

NUM=<int> **CONTROL**=<char> [**BEGIN**=<real>] [**INITSTEP**=<real>]
[**ENDVAL**=<real> | **STEPS**=<int>]

9.1.3 Parameters

- **NUM** is the sequence number of the electrode in the `PISCES` input deck designated as the control electrode, whose voltage or current is swept through the trace. Its integer value must be between 1 and 9, inclusive. (Default: none)
- **CONTROL** is either **VMAX**, **IMAX**, or **STEP**. **VMAX** denotes that a maximum voltage on the control electrode, specified by **ENDVAL**, is used as the upper bound on the trace. **IMAX** denotes that **ENDVAL** specifies a maximum control-electrode current for the trace. **STEP** signifies that the trace will proceed for a certain number of simulation points, specified by the `STEPS` parameter. In most cases **VMAX** or **IMAX** will be used because it is not known how many simulation steps will be taken to reach a certain voltage or current. (Default: none)
- **BEGIN** is the value of the voltage, in volts, at the starting point of the curve trace for the electrode designated by **NUM** (the control electrode). If an initial solution is performed by `Tracer`, **BEGIN** should be 0.0. If a previous solution is loaded into the input deck at the start of `Tracer` (see **SOLVE** card below), **BEGIN** should be equal to the voltage of the control electrode in this solution. (Default: 0.0V)
- **INITSTEP** is the initial voltage increment, in volts, of the control electrode. Thus, at the second solution point the control electrode will have a voltage of **BEGIN** + **INITSTEP**. A recommended initial step size is 0.1V. The sign of **INITSTEP** determines the direction in

which the curve tracing will initially be proceeded. If **INITSTEP** proves to be too large and `PISCES` cannot converge on the second solution point, `Tracer` will automatically reduce **INITSTEP** until convergence is attained, then proceed with the trace from this point. (Default: 0.1V)

- **ENDVAL** is used when **CONTROL**=**VMAX** or **IMAX**. `Tracer` stops tracing when the voltage (**CONTROL**=**VMAX**) or current (**CONTROL**=**IMAX**) of the specified electrode equals or exceeds the value specified by **ENDVAL**. Note that it is the absolute values of the voltage or current and of **ENDVAL** which are compared. (Default: 10.0V (**CONTROL**=**VMAX**), 10.0A/µm (**CONTROL**=**IMAX**))
- **STEPS** is used when **CONTROL**=**STEP**. It specifies the number of solution points `Tracer` should find. (Default: 10)

9.1.4 Examples

1. Electrode 3 is the control electrode. `Tracer` will initially proceed in the negative-voltage direction with an initial step of -0.1V. `Tracer` will proceed until the absolute value of the control current equals or exceeds 3A/µm.

```
control num=3 begin=0.0 initstep=-0.1 control=IMAX end=-3.0
```

2. Electrode 4 is the control electrode. `Tracer` will run until 65 solutions are found, starting at v4=0.0V with an initial v4 step of 0.5V.

```
control num=4 begin=0.0 initstep=0.5 control=step steps=65
```

9.2 FIXED Card

9.2.1 Description

A **FIXED** card is used to designate an electrode whose bias remains fixed throughout the simulation. There should always be at least one **FIXED** electrode and usually there are two or more. The two types of bias conditions available are voltage sources and current sources. The value of the bias is arbitrary, with one exception: a zero-current source (open contact) should be specified through the open-contact option on the **SOLVE** card and not on the **FIXED** card. If non-zero current sources are used for some electrodes in a simulation, in `inputfile` the user must create contact cards with the "current" option for each of these electrodes (see SECTION 10).

9.2.2 Syntax

NUM=<int> [**TYPE**=<char>] [**VALUE**=<real>] [**RECORD**=<char>]

9.2.3 Parameters

- **NUM** is the sequence number of an electrode in the `PISCES` deck. Its integer value must be between1 and 9, inclusive. (Default: none)
- **TYPE** is either **VOLTAGE** or **V** for a voltage source or **CURRENT** or **I** for a current source. (Default: **VOLTAGE**)
- **VALUE** is the fixed value of the current or voltage for the electrode specified by **NUM**. **VALUE** has units of either volts or amps/µm, depending on the specification of **TYPE**. Note that the specification of **VALUE** is optional since it is merely for reference and is not used by `Tracer`. (Default: 0.0)
- **RECORD** is either **YES** or **NO**. For **RECORD**=**YES**, the simulated current is recorded in the output file for a fixed-voltage electrode, while the simulated voltage is recorded for a fixed-current electrode. (Default: **NO**)

9.2.4 Examples

In every `Tracer` solution, electrode 1 has a voltage of 0.0V. The current in this node is recorded in `outputfile` at every solution point.

```
fixed num=1 type=voltage value=0.0 record=yes
```

9.3 OPTION Card

9.3.1 Description

An **OPTION** card is used to specify convergence criteria and solution-method options for any open electrodes, parameters which affect the smoothness and step-size control of the trace, which `PISCES` solution files are saved, and whether extra solution data is saved in `outputfile`.

9.3.2 Syntax

Simulations with one or two open contacts:

[**ABSMAX**=<real>] [**RELMAX**=<real>] [**DAMP**=<real>] [**TRYCBC**=<real>]

Smoothness and step-size control:

[**ANGLE1**=<real>] [**ANGLE2**=<real> [**ANGLE3**=<real>] [**ITLIM**=<int>]
[**MINCUR**=<real>] [**MINDL**=<real>]

Control of output files:

[**FREQUENCY**=<int>] [**TURNINGPOINTS**=<char>] [**VERBOSE**=<char>]

9.3.3 Parameters

- **ABSMAX** is the maximum current allowed in an open contact and is only relevant when open contacts are used and voltage biases are applied to these contacts. Convergence is satisfied when either the **ABSMAX** or **RELMAX** condition is met. (Default: 1.0×10^{-19} A/μm)
- **RELMAX** is the maximum ratio of open-contact current to control-electrode current and is only relevant when open contacts are used and voltage biases are applied to these contacts. Convergence is satisfied when either the **ABSMAX** or **RELMAX** condition is met. (Default: 1.0×10^{-9})
- **DAMP** is a number between 0 and 1.0 determining how quickly Tracer will converge on an open-contact solution using voltage biasing. The closer **DAMP** is to 1.0, the more quickly Tracer will converge, but there is also an increased chance of slower convergence due to overshoot. Usually the user should not be concerned with the value of **DAMP**. (Default: 0.9)
- **TRYCBC** is used only if there is an open contact. Tracer will only attempt to use zero-current biasing when the current of the control electrode is greater than **TRYCBC**. Otherwise, voltage biasing is used. In most cases the user does not have to worry about this parameter. (Default: 1.0×10^{-17} A/m)
- **ANGLE1**, **ANGLE2**, and **ANGLE3** are critical angles (in degrees) affecting the smoothness and step size of the trace. They are described in detail in [1]. If the difference in slopes of the

last two solution points is less than **ANGLE1**, the step size will be increased for the next projected solution. If the difference is between **ANGLE1** and **ANGLE2**, the step size remains the same. If the difference is greater than **ANGLE2**, the step size is reduced. **ANGLE3** is the maximum difference allowed, unless overridden by the **MINDL** parameter. **ANGLE2** should always be greater than **ANGLE1** and less than **ANGLE3**. (Defaults: **ANGLE1** = 5°, **ANGLE2** = 10°, **ANGLE3** = 15°)

- **ITLIM** is the maximun number of Newton loops for a given solution as specified in the method card of the `PISCES` input deck. The user should make sure that the value of **ITLIM** specified here is the same as that in the input deck. In certain cases, a `PISCES` solution may be aborted in `Tracer` because the solution will not converge within the given number of iterations. In some of these cases `Tracer` will try to redo the solution with a doubled number of iterations. If **ITLIM** is specified here, such attempts will be made. If there is no itlim statement or **ITLIM**=0, no attempts will be made. It is recommended that **ITLIM** be set to a low value, around 10 or 15 (or at least high enough to allow convergence of the initial solution). However, for GaAs devices a larger **ITLIM** of 20 or 25 is recommended. (Default: 0)
- **MINCUR** is the value of the control current, in A/µm, above which `Tracer` carefully controls step size and guarantees a smooth trace. Below this current level, the program simply takes voltage steps as large as possible, i.e., as long as numerical convergence can be achieved, without regard for smoothness. If **MINCUR** is set to 0.0, `Tracer` will not begin smoothness control until it is past the first sharp turn in the *I-V* curve. This value should be used when the user is only interested in the rough location of a break in the curve, such as the breakdown voltage of a single-junction device. If smoothness is required, a lower value should be specified. Setting **MINCUR** below $1.0\times10^{-15}\text{A}/\mu\text{m}$ is not recommended because `Tracer` has problems controlling smoothness at such low currents. (Default: 0.0A/µm)
- **MINDL** is the minimum normalized step size allowed in the trace. Usually the user does not need to adjust this parameter. Increasing **MINDL** will reduce the smoothness of the trace by overriding the angle criteria, resulting in more aggressive projection and fewer simulation points. Reducing **MINDL** will enhance the smoothness and increase the number of points in the trace. (Default: 0.1)
- **FREQUENCY** specifies how often the binary output (solution) files of the trace are saved. All *I-V* points are saved in `outputfile`. However, the `PISCES` solution files corresponding to these points are saved only if they are designated by **FREQUENCY**. If **FREQUENCY**=0,

none of the solutions is saved, except perhaps the turning points (see below). If **FREQUENCY**=5, e.g., the solution file of every fifth point will be saved to files named soln.5, soln.10, etc., along with its `PISCES` input file (input.5, input.10,...) and output *I-V* file (iv.5, iv.10,...). (Default: 0)

- **TURNINGPOINTS** is either **YES** or **NO**. If it is **YES**, the binary output (solution) file from `PISCES` will be saved whenever the slope of the *I-V* curve changes sign, i.e., there is a turning point. The name of the output file is soln.`num`, where `num` is the number of the current solution. For example, if the 25th point has a different sign than the 24th point, `Tracer` will save a file called soln.25. (Default: **NO**)
- **VERBOSE** is either **YES** or **NO**. If it is **YES**, certain information about each solution (which the user may not be interested in) is printed in `outputfile`. The information consists of the external control-electrode voltage, the load resistance on the control electrode, the slope (differential resistance) of the solution, the normalized projected distance of the next simulation *I-V* point, and the normalized angle difference between the last two simulation points. (Default: **NO**)

9.3.4 Examples

1. Step-size control will begin when the control electrode's current exceeds 1.0×10^{-14} A/µm. In the input deck itlim has been set to 12. Only essential information is saved in `outputfile`. The solution file of every tenth point, as well as any turning points, will be saved.

```
option mincur=1e-14 itlim=12 verbose=no frequency=10
+ turningpoints=yes
```

2. In a simulation with one or two open contacts, we want to keep the current through the open electrodes below 1.0×10^{-16} A/µm, regardless of the current through the control electrode. Thus **RELMAX** is set to a very low value so that it will not be a factor in determining the current at the open contact(s).

```
option absmax=1e-16 relmax=1e-25
```

9.4 SOLVE Card

9.4.1 Description

The solve card is used to specify how the initial solution is obtained, what simulator is used, and whether there are any open contacts (zero-current bias conditions). A `Tracer` run will start either with an initial solution or by loading a solution from a previous `PISCES` simulation. If such a previous simulation has one or two zero-current electrodes, the user has the option of either specifying the voltages on these electrodes or of simply designating them as open contacts.

9.4.2 Syntax

> **FIRSTSOLUTION**=<char> [**OPENCONT1**=<int>] [**OPENCONT2**=<int>
> [**SIMULATOR**=<char>] [**VOPEN1**=<real>] [**VOPEN2**=<real>]

9.4.3 Parameters

- **FIRSTSOLUTION** is either **INITIAL**, **LOAD**, or **CURRLOAD**. In all cases a solve statement should be present in the `PISCES` input deck (`inputfile`). The parameters of this solve card in `inputfile` are not used but rather the card itself is used to mark where a `PISCES` solve card should be placed by `Tracer` in `inputfile` (see SECTION 10).

  If **FIRSTSOLUTION**=**INITIAL**, a solution at thermal equilibrium will be solved by `Tracer` first. This implies that there cannot be any non-zero voltages or currents on a **FIXED** card. If the device has an open contact, i.e., a zero-current source, the user should <u>not</u> specify “current” on the contact line of the `PISCES` input deck to indicate a zero-current bias condition. Specifying **OPENCONT1** or **OPENCONT2** on the `tracefile` solve card is all that is needed.

  If **FIRSTSOLUTION**=**LOAD**, a load statement should be present directly above the solve card in `inputfile`, and it should designate the infile (see SECTION 10). This option is used if the trace is to begin from a previously generated input solution file. <u>The simulation which created this solution file must have used only voltage bias conditions</u>. An open-contact trace can still be generated from such an input solution file if the voltage bias condition on the open electrode(s) results in near-zero current for that electrode (see **VOPEN1**, **VOPEN2** below). Such an open-contact case would most likely arise if the user wanted to extend a previous

`Tracer` run in which voltage bias conditions were used on the zero-current electrodes for the last simulation point.

If the loaded solution is from a simulation using a zero-current bias condition, **FIRSTSOLUTION**=**CURRLOAD** should be used. In this case “current” should be specified on a contact card for each open electrode. As in the **FIRSTSOLUTION**=**LOAD** case, the existing `inputfile` load card is used by `Tracer`, which means the correct “infile” should be specified on a load card directly above the solve card in `inputfile`. (Default: none)

- **OPENCONT1** and **OPENCONT2** are the numbers of electrodes (between 1 and 9, inclusive) with a zero-current bias condition. There can be either zero, one, or two open contacts. When a device has an open contact, the user does not have to worry about convergence at low device-current levels. `Tracer` will automatically adapt the bias conditions to guarantee convergence. (Default: none)
- **SIMULATOR** is either **PISC2ET** (`PISCES-2ET`) or **MD3200** or **MD10000** (TMA). It designates the device simulator to be used by `Tracer`. Other additions may be made in the future. (Default: **PISC2ET**)
- **VOPEN1** and **VOPEN2** must be used if and only if there is an open contact and **FIRSTSOLUTION**=**LOAD** (voltage bias condition on open contact(s)). The values of **VOPEN1** and **VOPEN2** are the voltages of the open contacts **OPENCONT1** and **OPENCONT2**, respectively, in the loaded solution file designated on the load card of `inputfile`. If there is only one open contact, **VOPEN2** should not be specified. (Defaults: 0.0)

9.4.4 Examples

1. The trace starts by solving an initial solution at zero bias and uses `PISCES-2ET` as the simulator. Electrode 2 is an open contact.

```
solve opencont1=2 firstsolution=init simulator=pisc
```

2. The trace starts with a previous solution using only voltage bias conditions. In this loaded solution the open contacts 2 and 4 have voltages of 0.641V and 0.509V, respectively.

```
solve firstsolution=load simulator=pisc opencont1=2
+ opencont2=4 vopen1=0.641 vopen2=0.509
```

SECTION 10 Input Deck Specifications

The input deck used by `Tracer`, `inputfile`, is a standard `PISCES` file, but `Tracer` has certain requirements. For understanding the basic flow of an input deck, consult the `PISCES` manual. The mesh, region, electrode, doping, and model cards must already be present in the input deck. Additionally, the Newton solution method must be specified in the symbolic card. Other requirements are described below.

10.1 Load and Solve Cards

In `Tracer`, the user specifies whether to start with an initial solution or to load a previous solution (see Section 9.4). In either case, the user must mark a line in `inputfile` where the solve statement should go by starting the line with "solve". Any parameter specified in this solve statement is irrelevant. If `Tracer` is to start with a previous solution, `inputfile` must contain a standard load statement, above the solve line, containing the name of the input file to be used (i.e., load infil=<solution file name>). In the case of loading a solution with a zero-current bias condition, "current" should be specified on a contact card for the open electrode.

10.2 Contact Card

Contact cards are optional in `inputfile` except in the case of electrodes biased with a current source. The case of the zero-current source is noted in Section 10.1 above. If there are any electrodes with a finite-current bias condition, a contact card with the "current" option should be placed in `inputfile` for each such electrode, regardless of whether `Tracer` is to begin with an initial solution or a loaded solution.

Even if no contact cards are required in `inputfile`, a line starting with “$contact” must be present so that `Tracer` will know where to add a contact statement. This contact card is necessary because this is where the load resistance of the control electrode is specified by `Tracer`. There is no problem with placing a contact card for the control electrode in the input deck as long as it does not specify a resistance value (which should never happen). Note that at least the first five letters of “contact” must appear for `Tracer` (and `PISCES`) to recognize it.

10.3 Method Card

In order to specify the maximum number of Newton iterations per solution, the itlim statement of the method card must be used in `inputfile`. If no method card is present, `PISCES` uses a default itlim of 20. However, in order to use the double-itlimit option (see Section 9.3), a method card must be present in the input deck and itlim must be set to some value.

Another option must be specified in the method card if TMA’s MEDICI is used. In MEDICI, if a solution is aborted MEDICI will try to solve for an intermediate solution and then retry the original solution. This is not desirable when using `Tracer` since `Tracer` needs to keep track of aborted solutions. Thus, “stack=0” should be specified in the method card of MEDICI so that it does not attempt intermediate solutions. Analogously, the “trap” option should <u>not</u> be specified on the method card in a `PISCES-2ET` deck.

10.4 Options Card

When using `PISCES-2ET`, “curvetrace” should be specified on the options card so that `PISCES` will abort nonconverging solutions. Additionally, “nowarning” can be specified to prevent `PISCES` from printing warning messages which clutter the output, especially the warning issued when the load resistance changes value from one solution to the next.

SECTION 11
Data Format in Output Files

As each solution is found, it is recorded in `outputfile`. Naming `outputfile` is described in SECTION 8. At the start of each line is the number of the solution. The second column of data contains voltage values of the control electrode, while the third column contains current values of the control electrode. If there is a zero-current electrode, the voltage and current values of **OPENCONT1** will go in the next two columns, followed by the voltage and current of **OPENCONT2** if there is a second open electrode.

Values in the next columns depend on which data are recorded. If requested in the **FIXED** statements of `tracefile`, current values of fixed-voltage electrodes and voltage values of fixed-current electrodes will be recorded for each solution point in `outputfile`. The order from left to right is from low to high electrode number.

After the electrode information is recorded, further columns contain information about each solution if **VERBOSE=YES** in the **SOLVE** card of `tracefile`. These columns are, from left to right, external control-electrode voltage, load resistance on the control electrode, differential resistance, normalized distance of the next projection, and the angle difference between the current and previous solution points (see [1] for a description of these parameters).

The **FREQUENCY** and **TURNINGPOINTS** parameters in the **OPTION** card allow data to be saved for certain specified solutions. In `outputfile`, those points which are saved are marked with an asterisk next to the solution number. The files saved are the input deck, input.*i*; the *I*-*V* data file, iv.*i*; and the solution file, soln.*i*; where *i* is the number of the solution in `outputfile`.

SECTION 12
Comments

As of January 1994, `Tracer` works with `PISCES-2ET`, some in-house versions of Stanford `PISCES`, and to some extent md3200 or md10000, MEDICI Version 1.2.2.[1] Use of MEDICI is not yet robust and thus `Tracer` may or may not complete a trace using MEDICI. Also, MEDICI cannot yet be used for simulations with open contacts. `Tracer` has not been thoroughly tested for simulations with two open contacts.

The capability to calculate admittances using the difference method must be added to `Tracer` if it is to use simulators which cannot perform AC analysis when a load resistor is present (a previous version has this capability, so it should not be hard to implement).

A tar file has been created which contains the `Tracer` source code; an executable version of `PISCES`; this document in FrameMaker, MIF, and postscript formats; and a test suite of verified examples. This test suite contains all the files needed to run the examples in the appendix.

1. These implementations were developed in connection with Advanced Micro Devices, where TMA software is used, as part of a summer internship.

SECTION 13 Examples

In each of the `Tracer` examples below, a description of the simulation is given along with the command line used to invoke `Tracer` and figures with the listings of `inputfile` (the `PISCES` input deck), `tracefile`, and `outputfile`.

13.1 BV_{CEO}

The BV_{CEO} experiment is conducted by biasing an *npn* bipolar transistor's collector positively with respect to the emitter while the base is left open. The `PISCES` input deck, bvceo.pis, shown in Figure 13.1, defines the mesh, region, electrodes, doping, emitter contact, physical models, and solution method. Even though the contact card is not for the collector, which will be the control electrode, the presence of the card ensures that `Tracer` will be able to find the correct place to insert a contact card for the collector when it needs to. If we did not wish to use the contact card in bvceo.pis, we would still have to insert a line beginning with "$contact" above the model and symbolic cards. Notice that "nowarn" and "curvetrace" are specified on the options card and "newton" is specified on the symbolic card, while nothing is specified on the solve card.

In the trace file bvceo.tra (Figure 13.2), the **FIXED** card sets the voltage on the emitter electrode (num=1, as defined by bvceo.pis) to a constant value of 0.0V and states that the current through this electrode will not be recorded in `outputfile`. Electrode 3, the collector electrode, is designated as the control electrode. The **CONTROL** card states that the first solution will have a

```
title NPN Simulation for Toshiba w/ coarse mesh (1/19/92)
options nowarn curvetrace

mesh rect nx=11 ny=12
x.m n=1 l=0 r=1
x.m n=4 l=0.7 r=0.65
x.m n=11 l=2 r=1.2
y.m n=1 l=0 r=1.0
y.m n=3 l=0.2 r=0.7
y.m n=7 l=0.4 r=1.0
y.m n=12 l=2.5 r=1.3

region num=1 ix.l=1 ix.h=11 iy.l=1 iy.h=12 silicon

$electrode 1=emitter 2=base 3=collector
elec num=1 ix.l=1 ix.h=3 iy.l=1 iy.h=1
elec num=2 ix.l=10 ix.h=11 iy.l=1 iy.h=1
elec num=3 ix.l=1 ix.h=11 iy.l=12 iy.h=12

dop ascii n.type infil=npn1.p x.l=0 x.r=2 ra=0.8
dop ascii p.type infil=npn1.b x.l=0 x.r=2 ra=0.8
dop ascii n.type infil=npn1.as x.l=0 x.r=0.6 ra=0.8
dop gauss conc=1e18 p.type x.left=1.9 x.r=2 y.top=0 y.bot=0
+ char=0.3 ra=0.8

contact num=1 surf.rec vsurfn=8e5 vsurfp=8e5

model temp=300 srh auger conmob fldmob bgn impact
symbolic newton carr=2
method itlimit=15

solve
end
```

Figure 13.1 The input file bvceo.pis for the BV_{CEO} example.

```
fixed num = 1 type=voltage value=0.0 record = no
control num=3 begin=0.0 initstep=0.1 control=vmax end=20
solve opencont1=2 first=init sim=pisc
option verbose=no itlim=15 turnpts=yes freq=5
+ mincur=5e-12 absmax=5e-19
```

Figure 13.2 The trace file bvceo.tra for the BV_{CEO} example.

collector voltage of 0.0V, while the second solution will have a collector voltage of 0.1V. Tracing will continue until the collector voltage equals or exceeds 20V. If the initial step of 0.1V proves to be too large for convergence, `Tracer` will cut the step size in half, possible more than once, until it converges on a solution, and then will proceed from this solution.

In the **SOLVE** card, we specify that the base electrode (num=2) is to be treated as an open contact during the trace. Also, tracing will begin with a thermal-equilibrium solution and `PISCES-2ET` will be used for the simulation. Finally, the **OPTION** card specifies that only essential *I-V* data will be saved in the output file; the `PISCES` iteration limit is set to 15, agreeing with the `PISCES` deck in the input file; `PISCES` solutions will be saved for any turning points as well as for every fifth solution point; smoothness of the *I-V* curve will not be enforced until the collector current is greater than 5×10^{-12}A/µm; and while voltage biasing is used on the open base contact, a solution will be accepted only if the current through the base is less than 5×10^{-19}A/µm (unless the **RELMAX** condition predominates).

To run `Tracer`, the following command is typed at the prompt:

```
machine-prompt% tracer bvceo.pis bvceo.tra bvceo.out
```

While `Tracer` is running, the output of the `PISCES` runs are sent to the standard output, along with messages announcing when solutions are written to the output file. The output file, named bvceo.out in the command line, is shown in Figure 13.4, and a plot of the collector current vs. collector voltage is shown in Figure 13.4. In bvceo.out, we see that every fifth solution, along with solutions 24 and 36 (the turning points), has been saved in files named soln.5, soln.10, etc. Additionally, the last solution was saved in the file soln.last, although there is no asterisk marking the last solution in bvceo.out.

```
Soln  #Vctrl           Ictrl             Vcurr           Icurr
  1   0.000000e+00     6.640216e-19     0.000000e+00    -1.365566e-18
  2   1.000000e-01     4.536067e-17     1.000000e-01     1.341435e-19
  3   3.000000e-01     1.625653e-14     2.519331e-01     1.110998e-19
  4   7.000000e-01     1.258870e-13     3.047182e-01    -1.057191e-19
 *5   1.500000e+00     5.969134e-13     3.445794e-01    -6.215285e-20
  6   3.100000e+00     9.010139e-13     3.543271e-01     3.507138e-20
  7   6.300000e+00     1.937138e-12     3.725358e-01    -2.823590e-20
  8   1.270000e+01     5.523255e-12     3.960896e-01     4.401873e-20
  9   1.303983e+01     7.789304e-12     4.048689e-01     7.261209e-20
 10   1.331971e+01     1.233772e-11     4.167027e-01    -2.392157e-20
 11   1.351640e+01     2.144845e-11     4.310039e-01    -3.015821e-20
 12   1.364322e+01     3.968015e-11     4.469627e-01    -2.586306e-20
 13   1.375854e+01     1.126228e-10     4.740828e-01    -3.055604e-20
 14   1.383613e+01     4.044189e-10     5.073686e-01     2.662945e-20
 15   1.389759e+01     1.571640e-09     5.427516e-01    -6.997169e-20
 16   1.395684e+01     6.240608e-09     5.787376e-01     4.017923e-20
 17   1.401870e+01     2.491678e-08     6.149198e-01     5.800187e-20
 18   1.408638e+01     9.962280e-08     6.512089e-01     2.496370e-19
 19   1.416639e+01     3.984529e-07     6.876080e-01    -7.339886e-20
 20   1.427994e+01     1.593805e-06     7.241677e-01    -1.364024e-19
 21   1.450307e+01     6.375431e-06     7.610238e-01     1.500092e-21
 22   1.508580e+01     2.550435e-05     7.985911e-01     1.631978e-19
 23   1.653961e+01     1.020878e-04     8.384486e-01    -8.203646e-20
 24   1.743830e+01     2.563710e-04     8.696470e-01    -1.839253e-19
 25   1.721139e+01     3.337692e-04     8.797490e-01     2.752857e-21
 26   1.608475e+01     4.874279e-04     8.953096e-01    -6.556564e-20
 27   1.467484e+01     6.389540e-04     9.070167e-01     4.997494e-19
 28   1.349730e+01     7.523351e-04     9.136248e-01    -3.868294e-19
 29   1.275292e+01     8.310109e-04     9.178346e-01     1.061968e-19
 30   1.208031e+01     9.464580e-04     9.248369e-01     7.411538e-21
 31   1.149536e+01     1.064806e-03     9.311878e-01    -4.402454e-19
 32   1.072725e+01     1.238428e-03     9.391391e-01    -1.234551e-19
 33   1.032238e+01     1.369005e-03     9.444611e-01    -5.772530e-19
 34   1.018224e+01     1.459092e-03     9.480149e-01     3.337310e-19
 35   1.012632e+01     1.578200e-03     9.526528e-01     5.859350e-19
 36   1.015785e+01     1.736090e-03     9.586154e-01     1.039733e-19
 37   1.033709e+01     2.050704e-03     9.697170e-01     3.375426e-19
 38   1.082413e+01     2.676637e-03     9.898170e-01    -5.421011e-20
 39   1.216369e+01     3.909572e-03     1.027822e+00    -3.201785e-19
 40   1.300269e+01     4.379769e-03     1.042119e+00     3.947174e-19
 41   1.372551e+01     4.658421e-03     1.050298e+00    -6.979551e-19
 42   1.482329e+01     4.950367e-03     1.058339e+00     1.677125e-19
 43   1.612039e+01     5.192516e-03     1.064355e+00     1.389134e-19
 44   1.757689e+01     5.415434e-03     1.068990e+00    -4.269046e-19
 45   1.886187e+01     5.634169e-03     1.071871e+00    -2.778268e-19
 46   2.017690e+01     6.057492e-03     1.073324e+00     6.572976e-19
```

Figure 13.3   The output file bvceo.out for the BV_{CEO} example.

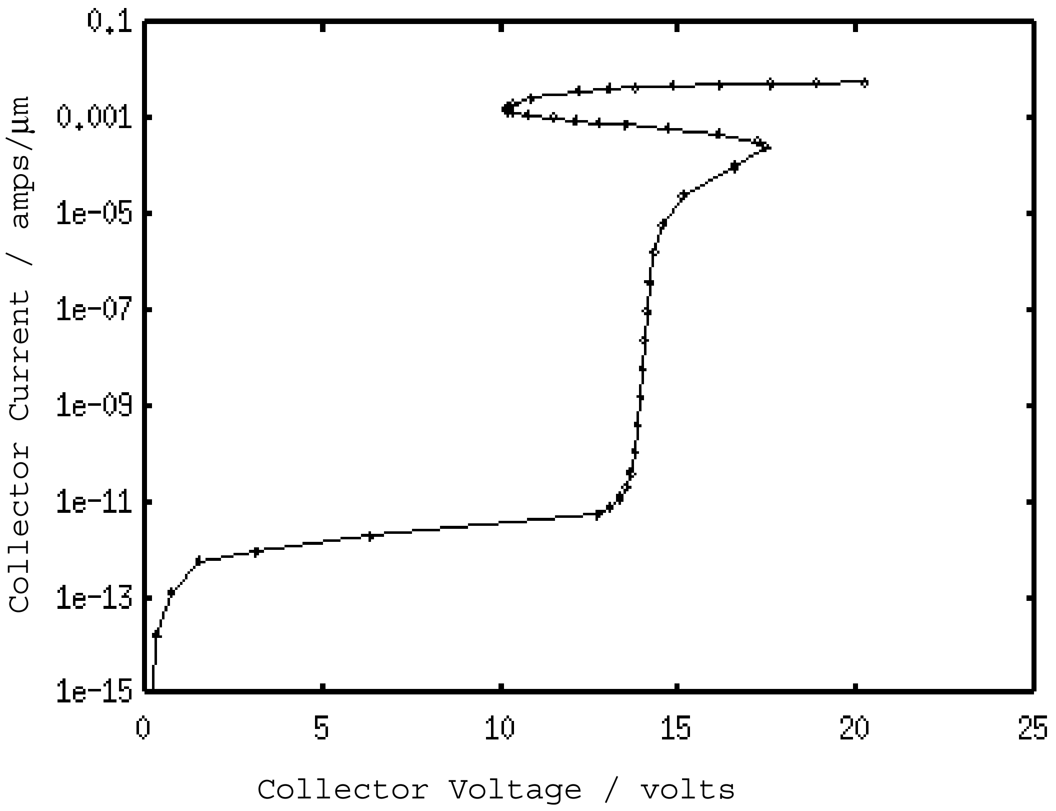


Figure 13.4 Collector current vs. collector voltage for the BV_{CEO} example.

At the top of bvceo.out, column headings mark the solution number, control-electrode (collector) voltage, control-electrode current, open-contact (base) voltage, and open-contact current as Soln, Vctrl, Ictrl, Vcurr, and Icurr, respectively. We see that the collector voltages for the first, second, and last solutions are 0.0, 0.1, and 20.18V, respectively. The final solution does not have a collector voltage of exactly 20V, as specified in bvceo.tra, because `Tracer` only guarantees that the curve will be traced out to at least 20V, not exactly 20V.

Other information regarding the trace must be inferred from the `PISCES` output displayed while `Tracer` is running (not shown). From this output we can see that voltage biasing was used on the open base contact for the first few solutions, in which the collector current is too small to allow stable use of zero-current biasing. A few `PISCES` simulations are actually run for each *I-V* point, with minor adjustments on the base voltage being made until the base current is less than **ABSMAX**. When the collector current is large enough, `Tracer` places a zero-current bias on the base. We can also see that a variable load resistor is placed on the collector when the collector current exceeds **MINCUR**. After this, the step sizes are regulated to produce a smooth curve.

13.2 GaAs MESFET

In this example, the drain of a GaAs MESFET is biased with respect to the grounded source with the gate set at -0.5V and the substrate grounded. Before `Tracer` can be used to sweep the drain electrode, a solution must be created, using `PISCES-2ET`, to set up the gate bias. The input deck shown in Figure 13.5 defines the device, finds the thermal-equilibrium solution, and then steps the gate bias to -0.5V while holding the other electrodes at 0V. The mesh and solution files are saved to the files mes.mesh and mesvg.5.ini, respectively.

For `Tracer`, another `PISCES` input deck must be created to use as the input file (Figure 13.6). In mesvg.5.pis the mesh file generated by mes.pis, mes.mesh, is read in, preempting the mesh, eliminate, region, electrode, and doping cards. Since `Tracer` will be starting with a previous solution, the name of the solution file to load must be given in mesvg.5.pis. This load statement appears directly above the solve card with the file name mesvg.5.ini, the solution file generated by mes.pis.

The trace file mesvg.5.tra is shown in Figure 13.7. In the three **FIXED** cards, the voltages of the source and substrate (num=1 and num=4, respectively, as defined by mes.pis) have been fixed at 0V, while the gate voltage (num=2) has been fixed at -0.5V. The current through the gate electrode will be recorded for each solution in the output file. The **CONTROL** card of mesvg.5.tra specifies that the drain (num=3) will be swept from 0.0V to a voltage where the current is greater than or equal to 4.1×10^{-4} A/μm, with an initial drain voltage step of 0.2V. On the **SOLVE** card, **FIRSTSOLUTION** is specified as `LOAD`, consistent with the input file mesvg.5.pis, and `PISCES-2ET` is designated as the simulator to use. Since **VERBOSE** is **NO** on the **OPTION** card, only the essential *I-V* data will be recorded in the output file. The iteration limit is 30, consistent with mesvg.5.pis, and every ninth solution, as well as those corresponding to turning points, will have its solution file saved.

To run `Tracer`, the following command is typed at the prompt:

```
machine-prompt% tracer mesvg.5.pis mesvg.5.tra mesvg.5.out
```

```
title mes.pis

mesh nx=53 ny=41 rect diag.fli outf=mes.mesh
x.m n=1 l=0 r=1
x.m n=5 l=1 r=0.85
x.m n=8 l=2 r=1.3
x.m n=11 l=3 r=0.7
x.m n=13 l=3.5 r=1
x.m n=18 l=4 r=0.8
x.m n=24 l=4.5 r=1.15
x.m n=32 l=5 r=0.85
x.m n=40 l=6 r=1.2
x.m n=43 l=7 r=1
x.m n=46 l=8 r=1.35
x.m n=49 l=9 r=0.7
x.m n=53 l=10 r=1.15

y.m n=1 l=-.01 r=1
y.m n=4 l=0.0 r=1
y.m n=7 l=0.01 r=1
y.m n=9 l=0.025 r=1
y.m n=20 l=0.19 r=1
y.m n=26 l=0.36 r=1.15
y.m n=39 l=3.0 r=1.25
y.m n=41 l=6.0 r=1.25

elim y.dir iy.lo=1 iy.hi=3 ix.lo=1 ix.hi=4
elim y.dir iy.lo=1 iy.hi=2 ix.lo=1 ix.hi=4
elim y.dir iy.lo=1 iy.hi=6 ix.lo=19 ix.hi=31
elim y.dir iy.lo=1 iy.hi=5 ix.lo=19 ix.hi=31
elim y.dir iy.lo=1 iy.hi=4 ix.lo=19 ix.hi=31
elim y.dir iy.lo=1 iy.hi=3 ix.lo=19 ix.hi=31
elim y.dir iy.lo=1 iy.hi=3 ix.lo=50 ix.hi=53
elim y.dir iy.lo=1 iy.hi=2 ix.lo=50 ix.hi=53
elim y.dir iy.lo=23 iy.hi=41 ix.lo=2 ix.hi=52
elim y.dir iy.lo=29 iy.hi=41 ix.lo=2 ix.hi=52
elim y.dir iy.lo=33 iy.hi=41 ix.lo=2 ix.hi=52
elim y.dir iy.lo=40 iy.hi=41 ix.lo=2 ix.hi=52
elim x.dir iy.lo=2 iy.hi=40 ix.lo=2 ix.hi=5
elim x.dir iy.lo=2 iy.hi=40 ix.lo=2 ix.hi=52
elim y.dir iy.lo=2 iy.hi=40 ix.lo=2 ix.hi=52
elim x.dir iy.lo=2 iy.hi=40 ix.lo=2 ix.hi=52
elim y.dir iy.lo=2 iy.hi=40 ix.lo=2 ix.hi=52
```

```
$*** regions
region num=1 ix.lo=1 ix.hi=53 iy.lo=4 iy.hi=41 gaas
region num=2 ix.lo=1 ix.hi=53 iy.lo=1 iy.hi=4 oxide
region num=2 ix.lo=16 ix.hi=34 iy.lo=1 iy.hi=7 oxide

$*** electrodes: 1=source 2=gate 3=drain 4=substrate
elec num=1 ix.lo=1 ix.hi=5 iy.lo=1 iy.hi=4
elec num=2 ix.lo=18 ix.hi=32 iy.lo=1 iy.hi=7
elec num=3 ix.lo=49 ix.hi=53 iy.lo=1 iy.hi=4
elec num=4 ix.lo=1 ix.hi=53 iy.lo=41 iy.hi=41

$*** doping
dop ascii x.l=0.0 x.r=10.0 inf=mei.dop
dop gaus x.l=-1 x.r=3.0 dos=5.0e13 cha=0.0607 peak=-0.0709
+ n.t erfc.lat lat.cha=0.0866
dop gaus x.l=7.0 x.r=11 dos=5.0e13 cha=0.0607 peak=-0.0709
+ n.t erfc.lat lat.cha=0.0866

$*** material
material num=1 eg300=1.42 affinity=4.07 vsat=10.0e6
+ permi=13.1 nc300=4.35e17 nv300=8.35e18
interface qf=-1e12 x.min=0.0 x.max=10 y.min=-.01 y.max=6.0

$*** contact
contact num=2 alu workf=4.84 surf

model conmob fldmob srh
symb newton carrier=0
method itlim=30 trap

solve ini
symb newton carrier=2
solve v2=-0.25
solve v2=-0.5 proj outfil=mesvg.5.ini

end
```

Figure 13.5 Mesh generation file, mes.pis, for MESFET example.

```
title mesvg.5.pis

option nowarn curvetrace

mesh inf=mes.mesh

material num=1 eg300=1.42 affinity=4.07 vsat=10.0e6
+ permi=13.1 nc300=4.35e17 nv300=8.35e18
interface qf=-1e12 x.min=0.0 x.max=10 y.min=-.01 y.max=6.0

contact num=2 alu workf=4.84 surf

model conmob fldmob srh hypert impact
symb newton carrier=2
method itlim=30

load infil=mesvg.5.ini
solve

end
```

Figure 13.6 The input file mesvg.5.pis for the GaAs MESFET example.

```
fixed num = 1 type=voltage value=0.0 record = no
fixed num = 2 type=voltage value=-0.5 record = yes
fixed num = 4 type=voltage value=0.0 record = no
control num=3 begin=0.0 initstep=0.2 control=imax end=4.1e-4
solve first=load sim=pisc
option verbose=no itlim=30 turnpts=yes freq=9
```

Figure 13.7 The trace file mesvg.5.tra for the GaAs MESFET example.

Figure 13.9 shows the output file, mesvg.5.out, in which the solution number, drain voltage, drain current, and gate current have been recorded as Soln, Vctrl, Ictrl, and I2, respectively. The solution files of points 9, 18, 27, and 29 (a turning point), as well as of the last point (not marked in the output file) were saved as soln.9, soln.18, soln.27, soln.29, and soln.last, respectively. A plot of the drain current vs. drain voltage is shown in Figure 13.8.

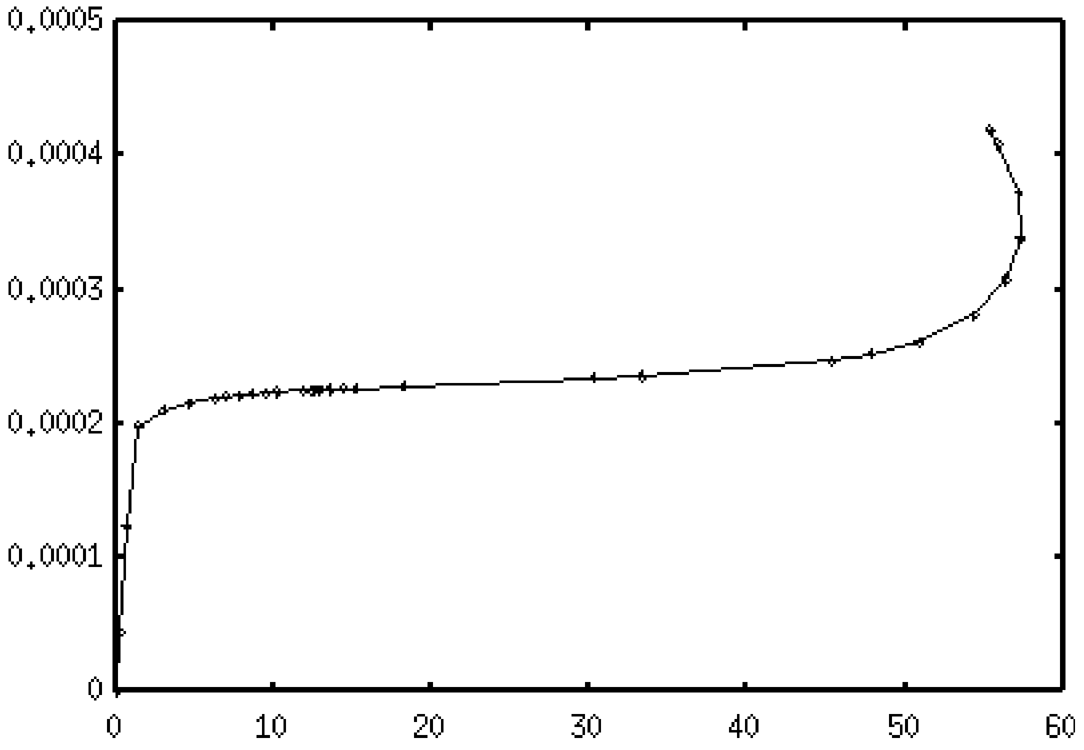


Figure 13.8 Drain current vs. drain voltage for the GaAs MESFET

```
#Soln   #Vctrl          Ictrl            I2
    1   0.000000e+00     1.903203e-16     -4.790550e-16
    2   2.000000e-01     4.411067e-05     -1.166555e-15
    3   6.000000e-01     1.233240e-04     -2.412571e-15
    4   1.400000e+00     1.985247e-04     -3.623966e-15
    5   3.000000e+00     2.104332e-04     -3.936303e-15
    6   4.600000e+00     2.155307e-04     -4.374722e-13
    7   6.200000e+00     2.189477e-04     -2.059647e-11
    8   7.000000e+00     2.202971e-04     -6.990150e-11
   *9   7.800000e+00     2.214088e-04     -1.707327e-10
   10   8.600000e+00     2.224028e-04     -3.638914e-10
   11   9.400000e+00     2.232137e-04     -6.557824e-10
   12   1.020000e+01     2.238725e-04     -1.043467e-09
   13   1.180000e+01     2.249711e-04     -2.208417e-09
   14   1.227965e+01     2.252584e-04     -2.671811e-09
   15   1.258651e+01     2.254261e-04     -2.980391e-09
   16   1.290104e+01     2.255882e-04     -3.308354e-09
   17   1.354587e+01     2.258872e-04     -3.995203e-09
  *18   1.441648e+01     2.262760e-04     -5.076190e-09
   19   1.517061e+01     2.266211e-04     -5.076190e-09
   20   1.818697e+01     2.280737e-04     -1.389891e-08
   21   3.025183e+01     2.340778e-04     -1.841648e-07
   22   3.326644e+01     2.357641e-04     -3.360147e-07
   23   4.526743e+01     2.472668e-04     -2.984657e-06
   24   4.787346e+01     2.524107e-04     -4.783998e-06
   25   5.085805e+01     2.612441e-04     -4.783998e-06
   26   5.436859e+01     2.808594e-04     -1.633647e-05
  *27   5.639889e+01     3.073427e-04     -2.612463e-05
   28   5.729060e+01     3.385567e-04     -3.460666e-05
  *29   5.719992e+01     3.725458e-04     -3.844234e-05
   30   5.586886e+01     4.090192e-04     -3.611460e-05
   31   5.534786e+01     4.193991e-04     -3.517154e-05
```

Figure 13.9 The output file mesvg.5.out for the GaAs MESFET.

Reference

[1] R.J.G. Goossens, S.G. Beebe, Z. Yu, and R.W. Dutton, “An Automatic Biasing Scheme for Tracing Arbitrarily Shaped I-V Curves,” *IEEE Trans. on Computer-Aided Design*, vol. 41, pp. 310-317, March 1994.

PART III

Mixed-Mode Device/Circuit Simulator

Francis M. Rotella, Zhiping Yu, and Robert W. Dutton

Acknowledgments

The prototype to the mixed-mode simulator was first developed in the summer of 1992 by Hui Wang, a visiting scholar from Tsinghua University, Beijing, China [1]. Funding for this project came from AASERT contract DAAH04-93-G-0178 on the parent Army contract DAAH04-93-G-0178. The MESFET example is provided by Matsushita Electric Industrial Co. in Japan and the GaAs/AlGaAs LED example is from OCD, HP, in San Jose, CA.

CHAPTER 14
Mixed-Mode Circuit and Device Simulation

14.1 Introduction

The Stanford mixed-mode interface provides a mechanism to include complex numerical devices in the SPICE circuit simulator when compact models are either inadequate or not available. Such devices include GaAs MESFET's, heterostructure devices, short channel MOSFET's, and optical devices. SPICE3f2 developed at UC Berkeley is used as the circuit simulator upon which the numerical device model is added.

The mixed-mode simulator comes configured to work with Stanford's version of PISCES-2ET. However, the modularity of the interface between SPICE and PISCES is such that any other device simulator may be interfaced with SPICE. Refer to "System Reconfiguration for a Different Device Simulator" on page 303 for more information.

SPICE has many different types of analyses which Stanford's mixed-mode simulator utilizes. Currently, a numerical device may be used in DC, AC small-signal, and transient analyses. Pole zero analysis has been included; however, this feature tends to have trouble in convergence for all but the simplest of circuits. Sensitivity analysis will be added in a later release.

14.2 Equations for Circuit Simulation

A typical circuit simulator solves the non-linear circuit equations by a modified form of Newton-Raphson (NR) [2]. The non-linear system of equations for the circuit can be represented as shown in the following equation according to the Kirchoff's current law:

$$\boldsymbol{F}(\boldsymbol{V}) = 0 \tag{14.1}$$

where $\boldsymbol{V}$ is the vector of node voltages and $\boldsymbol{F}$ represents the sum of the currents into each node in the circuit, and both $\boldsymbol{V}$ and $\boldsymbol{F}$ have dimension of N, the number of nodes in the circuit.

Applying Newton-Raphson to the above equation yields the linear matrices shown in Equation 14.2 where $J(\boldsymbol{V})$ is the Jacobian as given in Equation 14.3.

$$\boldsymbol{V}^{i+1} = \boldsymbol{V}^{i} - J^{-1}(\boldsymbol{V}^{i})\,\boldsymbol{F}(\boldsymbol{V}^{i}) \tag{14.2}$$

The index i is the iteration count during NR iterations.

$$J(\boldsymbol{V}) = \begin{bmatrix} \frac{\partial F_1}{\partial V_1} & \frac{\partial F_1}{\partial V_2} & \cdots & \frac{\partial F_1}{\partial V_N} \\ \cdots & & & \cdots \\ \frac{\partial F_N}{\partial V_1} & \frac{\partial F_N}{\partial V_2} & \cdots & \frac{\partial F_N}{\partial V_N} \end{bmatrix} \tag{14.3}$$

For each Newton iteration, the previous voltage $\boldsymbol{V}^{i}$ is known and hence $\boldsymbol{V}^{i+1}$ can be computed. A circuit interpretation of Equation 14.2 and the Jacobian matrix (Equation 14.3) can be made as follows. Because each Jacobian element has units of the conductance, we hereafter use G to replace J. By multiplying G on both sides of Equation 14.2 from the left, one obtains the following set of linear equations at $(i+1)$ -th iteration where i starts from 0:

$$G^{i}\boldsymbol{V}^{i+1} = G^{i}\boldsymbol{V}^{i} - \boldsymbol{F}^{i} \tag{14.4}$$

The matrix G^i and vector $\boldsymbol{F}^i$ are evaluated at $\boldsymbol{V}^i$. The above equation represents a linear circuit as far as node voltages $\boldsymbol{V}^{i+1}$ because both the coefficient G^i and the source term, i.e., the right hand side (RHS) of the above equation, are independent from $\boldsymbol{V}^{i+1}$. Furthermore, G^i can be interpreted as the linear conductance components, including both the linear components in the original circuit and differential conductance at $\boldsymbol{V}^i$, in the equivalent circuit. $G^i\boldsymbol{V}^i - \boldsymbol{F}^i$ can be viewed as the current sources. For a more detailed discussion, refer to McCalla [2].

The NR iteration is terminated when the convergence is reached, i.e., the change in $\boldsymbol{V}$ between two consecutive iterations is smaller than a predefined tolerance. In SPICE, it is required that the current change in each circuit branch is also below certain criterion when the convergence is considered to be reached.

Transient analysis is performed in a similar manner. For each time step, Newton-Raphson iterations are performed until convergence is met. In addition, the truncation error due to the time discretization is checked to determine if the time step is acceptable in terms of accuracy. If this error is too large, the time step is reduced and the computation is repeated.

For AC analysis, the DC solution is first sought and then the non-linear devices are linearized at the solution. This small signal equivalent circuit is used to find the AC response given an AC excitation vector.

14.3 Equations for Device Simulation

The details of the PISCES device simulator is discussed in Part I of this manual. This section discusses those details which are relevant to mixed-mode simulation.

The device simulator is responsible for supplying the terminal currents and the admittance matrix at the given voltage boundary conditions. The device simulator first solves $F(w, V, t) = 0$ where $w = f(\psi, n, p)$ for the drift/diffusion model and $w = f(\psi, n, p, T_n, T_p, T_L)$ for the duel energy transport model (see PART I) V is the terminal voltage on the device. From the solution, the terminal current density and conductance matrix can be calculated. For AC analysis, the admittance matrix must be calculated using frequency dependant AC analysis which is available in PISCES.

14.4 Mixed-Mode Interface

The mixed-mode interface is the code written at Stanford to provide the communication between SPICE and PISCES. Figure 14.1 shows how the two simulators are configured to talk with each other. Similarly to an analytic model (like a BJT or MOSFET), a numerical device is seen as just another block in SPICE. This block communicates to the numerical device simulator through an interface routine. Like other analytic models, the numerical device provides terminal currents (and conductances) for given node voltages. The difference is that the latter finds the solution numerically.

This interface utilizes a two level Newton scheme to solve a circuit that includes numerical devices. On the upper level is the circuit iterations to determine the node voltages. For each one of these

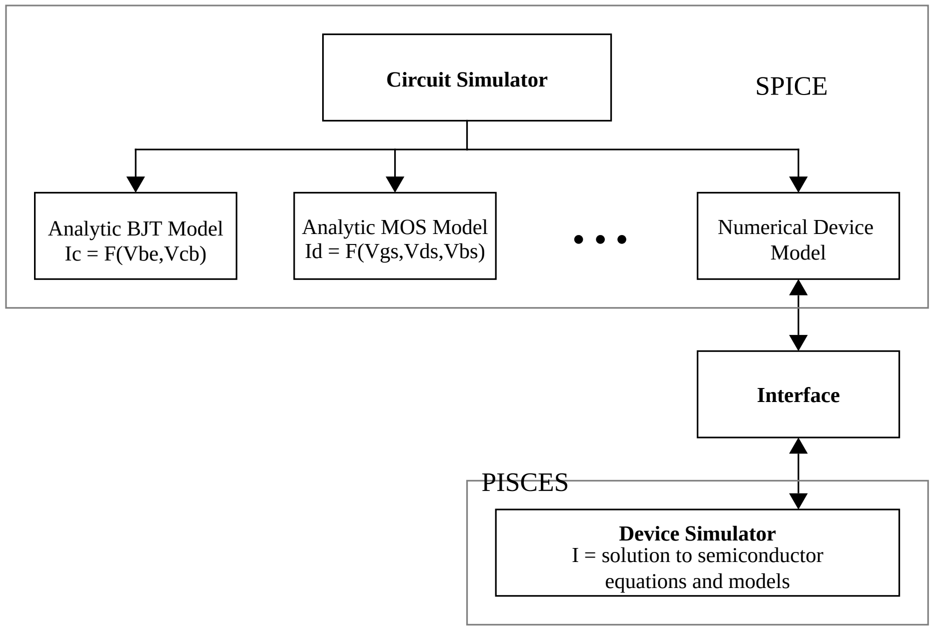


Figure 14.1 Configuration of a numerical device model in the SPICE circuit simulator.

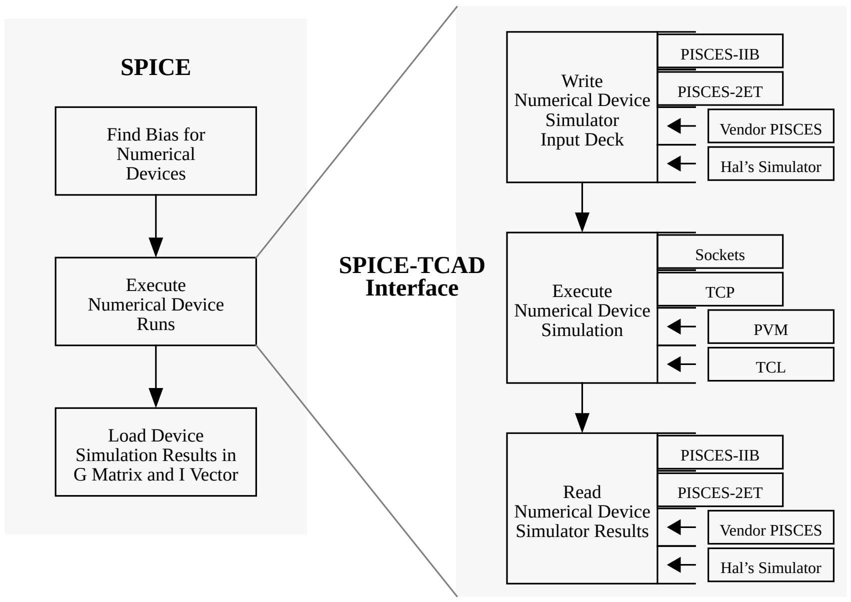

SPICE
Find Bias for Numerical Devices
Execute Numerical Device Runs
Load Device Simulation Results in G Matrix and I Vector
SPICE-TCAD Interface
Write Numerical Device Simulator Input Deck
PISCES-IIB
PISCES-2ET
Vendor PISCES
Hal's Simulator
Execute Numerical Device Simulation
Sockets
TCP
PVM
TCL
Read Numerical Device Simulator Results
PISCES-IIB
PISCES-2ET
Vendor PISCES
Hal's Simulator

Figure 14.2 Mixed-mode interface routines

iterations, the device simulator solves for the terminal currents and conductance matrix using another Newton iteration within the device simulator (the lower level). The advantage of this method is the modularity created in the code.

Figure 14.2 shows the flow diagram for the execution of a single iteration step at the upper level. The blocks on the far right hand side represents different modules for the device simulator and/or method of execution. The sequence of steps is as follows.

1. Determine the voltage boundary conditions, time step, and/or frequency SPICE requires for the current iteration/solution. The exact information depends on the type of analysis.
2. Create the numerical device input deck that solves for the information requested by SPICE. A separate module exists for each device simulator that is interfaced to SPICE.

3. Execute the numerical device simulations on the current host or on a remote host by some specified protocol.
4. Read the results of that simulation.
5. Based upon the analysis type, load the SPICE conductance (admittance) matrix and current vector.
6. Repeat for each device at each circuit iteration.

Since a device simulation is required for each circuit iteration, the interface algorithms are given some intelligence to limit the computations required by the numerical device. The most important aspects or those algorithms are as follows:

- Maintain files containing previous device solutions and utilize those solutions as a starting point for the requested bias.
- Maximize the use of SPICE's predictor corrector routines to decrease the calls to the device simulator.
- Use a-priori knowledge of the device behavior to provide better guess of node voltages and to limit changes across pn junctions

14.5 Computational Cost and Benefits

The two level Newton method for mixed circuit/device simulation incurs a significant computational cost. Mayaram had investigated that cost for a variety of algorithms including the two level Newton [3]. The most common of the alternate algorithms is the full Newton which Rollins had implemented previously [4].

In the full level Newton algorithm, the circuit matrix and device matrix are combined into a single matrix and all variables are solved simultaneously. As a result, the computational time is reduced, but there are trade-offs.

The two level Newton algorithm has the following advantages:

1. It has better convergence for DC analysis if node voltages are unknown. The full Newton requires all circuit node voltages to be specified within a certain percentage; otherwise, it fails to converge.

2. The modularity in two level Newton allows many device simulators to be used simultaneously. For example, PISCES may be used for standard devices in the circuit whereas an in house device simulator may be used for novel devices
3. It is easier to implement a parallel version of the two level Newton algorithm. Each numerical device simulation can be relegated to a node of a parallel machine whereas the matrix for the full Newton algorithm has to be partitioned to each node.
4. By utilizing SPICE as the circuit simulator, any improvements/modifications in SPICE automatically benefit the mixed-mode simulation.

Likewise, there are a number of disadvantages to the two level Newton algorithm:

1. Given a good estimate of all the circuit node voltages, the full Newton algorithm converges more quickly than the two level Newton.
2. Similarly, since transient analysis involves small voltage changes at each time step making the problem well behaved, the full Newton method converges much more quickly for this case as well. Mayaram determined a factor of 1.7 times as fast [3].

14.6 Parallelization

In order to reduce the overall user time per circuit iteration, the numerical devices (i.e. the lower level Newton iterations) can be solve on different nodes on a network of machines. Because each device simulation is an independent unit, parallelization is relatively trivial.

The mixed-mode interface contains two mechanisms by which the parallel simulations are executed. The most basic of these algorithms utilizes standard Unix sockets. A more elegant mechanism involves PVM (Parallel Virtual Machine) which allows a heterogenous network of computers to act like a parallel machine [5]. Figure 14.2 shows a block diagram of how PVM is utilized.

The mixed-mode interface spawns a queueing process on a node of the virtual machine. This process ascertains the number of nodes that can be used for a device simulation. Whenever the interface program needs solutions from a number of numerical devices, it sends a request to the queueing

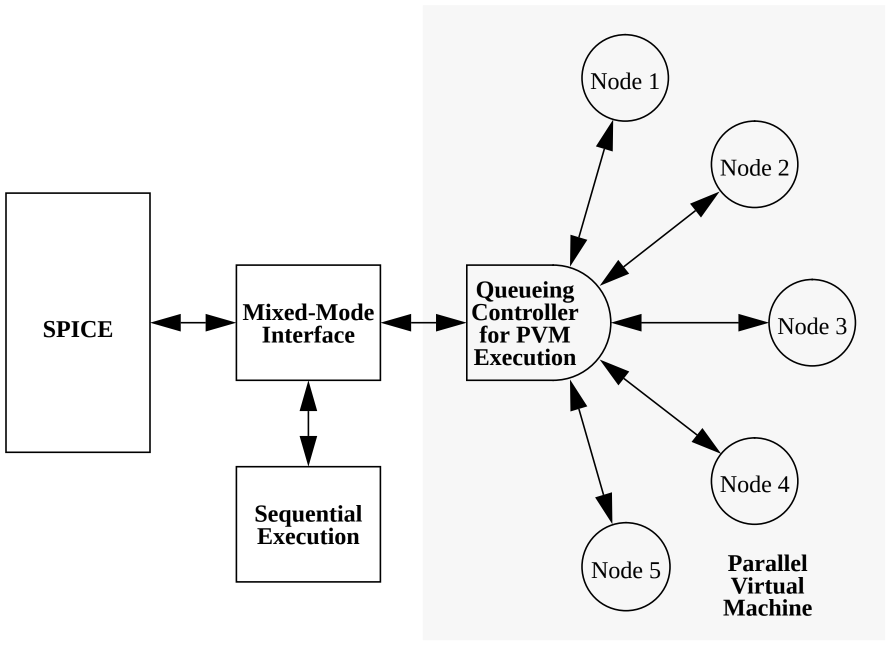


Figure 14.3 Execution of the device simulations.

program which then executes the device simulations on the available nodes. Once the device simulations are complete, the queueing program sends back the solutions to the interface. As a result, the computational load on the nodes is optimized.

CHAPTER 15
Use of Mixed Mode Simulation

15.1 Introduction

The Stanford mixed-mode simulator provides a mechanism to incorporate numerical devices into a full circuit simulation with linear and non-linear components. The mixed-mode simulator uses the industry standard SPICE as the circuit simulator with an added model for the numerical device. This numerical model is configured modularly such that any device simulator may be added. (Refer to "System Reconfiguration for a Different Device Simulator" on page 303 for more information.) For the purpose of this document, Stanford PISCES is used.

This chapter describes how to include a numerical model in a circuit simulation. All descriptions are based upon SPICE version 3f and hereafter referred to as SPICE only.

15.2 New SPICE Cards

Upon installing and configuring SPICE, the numerical model may be utilized in any circuit. The next two sections describe how to use the SPICE element card to specify the numerical device instance and the SPICE **.model** card for setting the parameters for the numerical device. The format for the cards is given in the SPICE manual notation [6].

COMMAND SPICE Element Card

As with all element cards in SPICE, this card describes a specific instance of the numerical device in the circuit. The name of this instance is used as the predecessor for all files accessed in solving the specific numerical device.

SYNTAX

```
Nxxxxxxxx n1 n2 . . . nM mname <area> <off>
+ <ic=v1M, v2M . . . v(M-1)M> <temp=T>
```

PARAMETERS

Nxxxxxxxx

The **N** identifies the element type as being a numerical device and the **x**'s are used to represent a unique character string, which can be up to eight characters long, to identify the device.

n1 n2 . . . nM

The node numbers in the circuit to which the **nM** nodes of the numerical device are connected. They correspond in sequence to the electrodes of the device as specified in PISCES. There is a maximum limit of ten nodes per device. The voltage on any node is specified with respect to the last node as given by the **nodes** parameter on the **.model** card.

mname

The model name links this instance of the numerical device to the appropriate **.model** card.

area

The area or length factor for the device. All currents, capacitances, and conductances are multiplied by this value. The default is 1.0.

off

If **off** is specified, the DC operating point is determined with the terminal voltages for that instance of the device set to zero.

ic

These initial conditions for the device are used with the **UIC** specification on the **.tran** card. Each voltage initial condition is specified with respect to the last node.

temp

Operating temperature of the device. If none is given, the circuit temperature is used as the default.

EXAMPLES

```
Nbjt1   8   7   5   npndev   5
```

This adds a numerical device with the name of Nbjt1 connected to circuit nodes 8, 7, and 5. These nodes correspond to electrode numbers 1, 2, and 3 in the numerical device's mesh. Electrode number 3 is taken as ground. Analogously with SPICE's compact model for a bipolar junction transistor, Electrode 1 (Node 8) is the collector, Electrode 2 (Node 7) is the base, and

Electrode 3 (Node 5) is the emitter. An area factor of 5.0 scales the device appropriately.

```
Npn1   6   4    pndev   area=6.5   ic=0.75
```

This is a two node numerical device with circuit Node 6 corresponding to Electrode 1 and circuit Node 4 corresponding to Electrode 2. An initial condition of V_{64}=0.75 V is placed on the device. The area is scaled by a factor of 6.5.

BUGS

The **ic** and **off** parameter have not been tested extensively.

Some software paths have not been tested and may result in problems.

COMMAND SPICE Model Card

As with all **.model** cards in SPICE, this one is used to describe all instances of the same type. Common parameters are specified and used by all instances that reference the given model name.

SYNTAX

```
.model mname mtype nodes=ival [dtype=ival] [vto=rval]
+ [model=ival] [method=ival] [vmax=rval] [vmin=rval]
+ [hifreq=rval] [maxtrys=ival] [level=ival] [float=ival vbc=rval
+ bcdep=ival] [oedev=ival] [tnom=rval]
```

PARAMETERS

mname

Model name used for reference on the element cards. The mixed-mode interface expects to find certain files with this name as the predecessor to the file names.

mtype

The model type is given by a word mnemonic from Table 15.1. The mnemonic indicates the polarity of the numerical device and the mechanism by which the numerical simulation is executed.

A plain **pis** device (independent of polarity) tells the mixed-mode interface to execute each of the numerical device runs sequentially on the same machine as SPICE. For the **prl** model, the mixed-mode interface uses UNIX sockets to execute each numerical simulation on a different machine.

For the **pvm** model, the mixed-mode interface uses PVM message passing to send the biases to a queueing routine and ultimately a “slave” routine on a node of the parallel virtual machine. Refer to “Running a Mixed-Mode Simulation” on page 286 for more information.

Table 15.1

mtype	Description
pis	generic numerical device (default)
npis	n polarity numerical device (npn, nmos, pn junction)
ppis	p polarity numerical device (pnp, pmos, np junction)
prl	generic numerical device simulated in parallel
nprl	n polarity numerical device simulated in parallel
pprl	p polarity numerical device simulated in parallel
pvm	generic numerical device simulated using pvm
npvm	n polarity numerical device simulated using pvm
ppvm	p polarity numerical device simulated using pvm

nodes

The number of numerical device nodes. This parameter defaults to 10 if no value is given. SPICE does not have a problem if this number is larger than the actual number of electrodes on the numerical device. The unidentified nodes are connected to ground by default. Likewise, one must be certain that the numerical device simulator can handle a “solve” statement with voltages specified for electrodes that do not exist. To avoid this problem, one should always specify this parameter.

The **nodes** parameter may be less than the actual number of nodes in the numerical device. In this case, the extra nodes on the numerical device are considered floating and always ignored by the SPICE circuit simulator.

Other **.model** parameters related to floating nodes include **float**, **vbc**, and **bcdep**. These parameters allow the user to switch boundary conditions on the floating node during a DC circuit solution.

dtype

This integer value provides SPICE with some a-priori knowledge of the device behavior. This parameter is not necessary, but is useful in achieving circuit convergence. Specifying a device type allows SPICE to make good guesses and limit voltage changes across pn junctions. The supported devices and their integer representations are configured in the NPISCdefs.h file. The default configuration is given in Table 15.2. The **mtype** parameter determines the polarity of the device. The electrode sequence for the electrodes is analogous to the SPICE compact model equivalents.

Table 15.2

dtype	Description	Electrode Sequence
0	generic (default)	does not matter
1	BJT	Collector Base Emitter Substrate
2	MOSFET	Drain Gate Source Bulk
3	diode	Anode Cathode
4	MESFET	Drain Gate Source Bulk
5	LED	Anode Cathode
6	Inverter	Psource Pgate Pdrain Ndrain Ngate Nsource Pbulk Nbulk
7	Resistor	Contact1 Contact2 Substrate (Cathode of diode)
8	JFET	Drain Gate1 Gate2 Source

vt0 or vto

This parameter represents the absolute value of the turn on voltage for the device. For a bipolar junction transistor it is Vbe_{on}, for a diode it is Vd_{on}, and for a FET it is Vgs_{on}. The default value at room temperature is 0.7 V which is typical for a silicon pn junction. The polarity of the device model determines the sign.

vmax

This is the maximum voltage (in signed real number) placed on any numerical device electrode with respect to the reference electrode. The default value is 50.0 volts which one needs to increase for power devices.

vmin

This is the minium voltage (in signed real number) placed on any numerical device electrode with respect to the reference electrode of that numerical device. The default value is -50.0 volts.

hifreq

This value separates low and high frequency simulation of the numerical device. When the circuit frequency is below **hifreq**, the device's small signal parameters are calculated from a zero frequency device simulation. When the circuit frequency is above **hifreq**, the device's small signal parameters are calculated from a small signal device simulation at the given frequency. The trade-off is accuracy versus simulation time. The default value for **hifreq** is 1 kHz.

maxtrys

The maximum number of attempts the mixed-mode interface tries to get the device simulation to converge. Different bias stepping schemes are attempted with each try.

level

This integer represents a specific device simulator as defined in the NPISCdefs.h. Currently, the only one supported (and the default) is Stanford PISCES which is defined as level=0. This integer value determines how the **method** and **model** parameters are interpreted as well as the calling of the correct device simulator. More **levels** may be added so that other device simulators may be used in a mixed-mode simulation. Refer to “Adding Another Device Simulator” on page 311 for more information.

model

This is an integer value for the binary encoded representation of the model card in the device simulator input deck. The meaning of the integer value is described in the NUMDEVdefs.h file and interpreted by a routine specifically written for that device simulator. If this parameter is not specified and the file **mname.model** exists, the model card is taken as the content of that file. If neither is given, this card is not included in the input deck for the device simulator. Refer to “Running a Mixed-Mode Simulation” on page 286 for more information.

method

This is an integer value for the binary encoded representation of the method card in the device simulator deck. The meaning of the integer value is described in the NUMDEVdefs.h file and interpreted by a routine specifically written for that device simulator. If this parameter is not specified and the file **mname.method** exists, the method card is taken as the content of that file. If neither is given, this card is not included in the input deck for the device simulator. Refer to “Running a Mixed-Mode Simulation” on page 286 for more information.

float, vbc, and **bcdep**

These three parameters are used to control the boundary conditions on a floating node that can switch between a current and voltage boundary condition during DC analysis. This capability aids in the convergence of numerical devices with a floating node. At low currents in the device one uses a voltage boundary condition on the floating node and for high currents, one uses a current boundary condition on the floating node. The value of the voltage and current source on the floating node is zero.

The parameter **float** contains the integer number for the floating node. It must be greater than the number specified by the **nodes** parameter. The parameter **vbc** determine the voltage at which the boundary condition is switched. The parameter **bcdep** specifies the positive and negative electrode on which the boundary condition depends. Refer to the examples for clarification.

oedev

An integer value of one for **oedev** tells SPICE that the numerical device will produce optical output. As a result, SPICE adds an equation to its circuit matrix to store the value of that output. In a future release, an integer value of two will tell SPICE that the numerical device has light incident upon it. The amount of incident light will be determined by referencing a circuit node. The default of zero means no photons are involved.

tnom

This parameter specifies the nominal temperature at which the parameters are given. Currently, the only parameter affected by **tnom** is **vt0**. The default value is 300K.

EXAMPLES

```
.model npndev npis vt0=0.7 dtype=1 model=93 method=336
+ nodes=3
```

This example defines the **npndev** model used in the previous example of an element card. This **.model** card tells SPICE that the **npndev** is of polarity **n** and that sequential simulations are executed for each instance of this model. The device type is a bipolar transistor as specified by the **dtype** parameter. The turn-on voltage for the device is 0.7 volts. The **model** and **method** for the device simulation is specified on the card and is explained in the next section. The number of **nodes** is given as three which corresponds to the three nodes on the previous element card.

```
.model hpled ppis vt0=1.4 maxtrys=2 nodes=2 dtype=5
+ float=3 vbc=1.0 bcdep=21 oedev=1
```

This is a very complex model used for the simulation of an GaAs/AlGaAs LED with a floating layer. The turn-on voltage for the LED is 1.4 volts as defined by **vt0**. The number of **nodes** is two. The maximum number of tries for a given bias is limited to two. The floating node is electrode number 3 as given by the **float** parameter. The voltage boundary condition is used when V_{21} is below 1.0 volts and the current boundary is used when V_{21} is greater than or equal to 1.0 volts. The parameter **bcdep** specified V_{21} as the controlling voltage and the parameter **vbc** specifies the voltage of V_{21} when the boundary condition change is to takes place.

In addition, this device will produce optical output and hence, an optical response. The **model** and **method** lines for the numerical simulator input deck is given in the files **hpled.method** and **hpled.model.**

BUGS

The advanced parameters for floating layers have been generalized, but may not be useful for all cases.

When **vmax** and **vmin** are set stringently, they tend to cause SPICE to oscillate between the extremes. These parameters should be set such that they prevent the numerical device from entering breakdown.

Some software paths have not been tested and may result in problems.

15.3 Running a Mixed-Mode Simulation

This section will describe how to set up a mixed-mode simulation using **level**=0 (Stanford's version of PISCES). The first subsection describes the files the user must supply and those generated during the simulation. The next subsection describes the concept of the binary encoded method line and binary encoded model line. Finally, each of the different execution methods is addressed and the requirements for each is described.

15.3.1 Files Required and Generated

mynetlist.spi

This file contains the circuit netlist of all the element cards and model cards for the circuit. Any valid SPICE construct is permitted. The name of the file can be arbitrary, but it is recommended that the extension **.spi** is used so that it may easily be recognized.

mynetlist.raw

The easiest way to run SPICE is to create a net list such that the circuit simulation is performed in batch mode. As a result, the user needs to save the solution SPICE computes. The name of the file can be arbitrary, but it is recommended that the extension **.raw** is used. To run SPICE in batch mode one types the following on the command line:

```
spice3 -b mynetlist.spi -r mynetlist.raw
```

Refer to the SPICE manual for more information [6].

mname.msh or mname.mesh.pis

One of these files must be supplied to load or create the mesh during the PISCES simulation. The first is a standard PISCES mesh file. The second contains valid PISCES commands for creating the mesh from scratch. The **mname** corresponds to the model name given on the SPICE **.model** card. The extension **.msh** or **.mesh.pis** must be exactly as shown. The mixed-mode code first looks for **mname.msh**, if that file does not exist, it looks for **mname.mesh.pis**. If neither exists, SPICE aborts.

Although these files are based on PISCES, they should be completely compatible and/or analogous for other device simulators. The actual format is determined by the programmer who adds the new device simulator.

Using **mname.msh** provides for faster simulation because PISCES just loads the mesh and doesn't have to create it each time. However, for some simulations, one may choose to create the mesh each time and therefore, the second option is available. When using **mname.mesh.pis**, the user should not include an end card.

mname.model and/or **mname.method**

These files contain the model and method cards for the PISCES input file, respectively. In writing an input deck, the mixed-mode interface should be given these cards. The binary encoded **model** and **method** parameters on the **.model** card in SPICE are the most efficient way to provide these cards without additional files or overhead. However, the encoding only accounts for logical variables; hence using the files provides a capability for including numerical variables on these PISCES cards.

mname.soln.init

This is a PISCES solution file containing the initial zero bias solution for the model **mname**. This only has to be found once for the model since all instances use the same starting point.

Nxxxxxxxx.pis

This file contains the PISCES input deck that the mixed-mode interface writes for the instance named **Nxxxxxxxx**.

Nxxxxxxxx.out

This file contains the output of the last PISCES simulation run for the specific instance.

Nxxxxxxxx.soln.prev0, Nxxxxxxxx.soln.prev1, and **Nxxxxxxxx.soln.prev2**

These files are generated by PISCES and contain the solutions for the specific instance at some previous bias. For DC analysis, the **.prev0** file contains the solution for the current circuit iteration, the **.prev1** file contains the solution for the previous circuit iteration, and the **.prev2** file contains the solution for iteration before the previous. For transient analysis, **.prev0** file contains the solution for the current time step, the **.prev1** file contains the solution for the previous time step, and the **.prev2** file contains the solution for the time step before the previous time step.

Nxxxxxxxx.log.ac and **Nxxxxxxxxx.log.iv**

These files are generated by PISCES and contain the conductance/small signal parameters and the current vector respectively.

hostname.pvm and **hostname.prl**

These files are supplied by the user and contain a listing of hostnames if the mixed-mode simulator is used in a parallel configuration. Refer to “Types of Execution” on page 289 for more information.

15.3.2 Method and Model Parameters

The user may specify the binary encoded representation of the PISCES model card or PISCES method card on the SPICE **.model** line. This is accomplished by summing the numeral next to the appropriate action as given in the following tabulations.

The binary encoded model parameter is computed as follows:

+1	turns on srh
+2	turns on consrh
+4	turns on auger
+8	turns on bgn
+16	turns on conmob
+32	turns on analytic
+64	turns on fldmob
+128	turns on surfmob
+256	turns on impact
+512	turns on ccsmob
+1024	turns on fermi
+2048	turns on incomplete
+4096	turns on photogen

The binary encoded method parameter is computed as follows:

+1	turns off xnorm
+2	turns on rhsnorm
+4	turns on limit
+8	turns on fixqf
+16	turns on trap
+32	turns on autonr
+64	turns off 2nd
+128	turns on tauto
+256	turns off tauto
+512	turns off l2norm
+1024	turns of extrapolate

For example, to generate the following model and method lines in the PISCES input deck,

```
model consrh conmob bgn auger
method trap ^2nd ^tauto
```
[1]

the SPICE input file should contain a line similar to:

```
.model fmrdev npis nodes=3 vt0=0.7 model=30 method=336
```

15.3.3 Types of Execution

There are three types of execution modes available to the user, but the compiled version of the mixed-mode code may not necessarily have all these capabilities. The type of execution is determined by the type of model given for the numerical device.

pis, npis, and **ppis**

This is the simplest of the possible models. A **pis** model sequentially executes the instances on the machine SPICE is executed. The minimum requirement is a SPICE netlist and a file containing the mesh.

1. 2nd and tauto must be turned off for mixed-mode simulations using PISCES.

prl, nprl, and **pprl**

This type of model takes advantage of a network of computers. It uses standard UNIX sockets to execute the device simulations on remote hosts. The file system must be mounted on all these hosts with the same absolute path name. After a failure of **maxtrys** attempts, the parallel mode switches to the sequential mode for one attempt only. As a result, if a remote machine crashes or becomes unavailable for some reason, the mixed-mode simulation may not necessarily halt.

The **hostname.prl** file contains a listing of the hosts on which the device simulations are executed. Each numerical device in the circuit is assigned a machine from the list and it is always executed on that machine. If there are more devices than machines then some machines can have more than one device executed on it. It is recommended that the machines be listed in decreasing computational power.

The login and password for the remote machine is required. The mixed-mode code looks for the host name in the **.netrc** file. However, since the login and password for many hosts are the same, the user can use a machine name of “spice” in the **.netrc** file and the mixed-mode code uses that login and password for all hosts not found in the **.netrc**.

pvm, npvm, and **ppvm**

This type of model uses PVM to execute the numerical device simulations in parallel. PVM stands for Parallel Virtual Machine and may be obtained from netlib@ornl.gov. The user’s account must be configured for use with PVM as described in the manual and the user should have a working knowledge of PVM [5].

Three files are necessary for using PVM to execute a mixed-mode simulation. The user should place the executable files **npiscctrl** and **npiscslave** in the PVM bin directories of the appropriate machine architectures. In addition, **hostname.pvm** needs to contain a listing of all the nodes used for the mixed-mode simulation; however, they do not need to be added to the parallel virtual machine. If they are not in the parallel virtual machine, the mixed-mode simulator attempts to add them.

Finally, the user should start and/or make sure that the PVM daemon is running on the local machine and then start the SPICE simulation. If the PVM daemon is not running, mixed-mode defaults to sequential execution.

CHAPTER 16
Examples

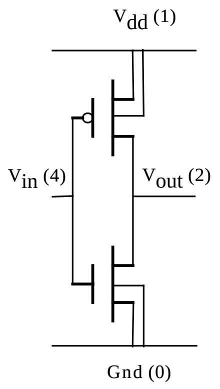
16.1 Introduction

This chapter describes examples using three types of analyses. The first is a CMOS inverter analyzed with a DC sweep on the input. The second is a GaAs MESFET analyzed for its high frequency response. The third is a six transistor SRAM cell analyzed through a read/write cycle. Another transient analysis is used to analyze a fiber optic transmitting circuit which includes a heterostructure LED with a floating layer.

16.2 Single Stage CMOS Inverter (DC Analysis)

This simple example demonstrates the basic capabilities of mixed-mode. The V_{out} versus V_{in} characteristics of a single CMOS inverter is simulated. Figure 16.1 shows the circuit schematic and the SPICE net list for the circuit. The two MOS transistors are numerical devices with electrode number 1 being the drain, 2 the gate, 3 the source and 4 the bulk.

For this simulation the mesh files for the two devices are created separately and are called **fmrpmos.msh** and **fmrnmos.msh**. These mesh files contain the grid and doping for the numerical device. The PISCES method and model options are chosen by the binary encoded SPICE parameters **method** and **model** as follows.



```
* cmos inverter example (DC analysis)

vin 4 0 dc 0
vdd 1 0 dc 5

nm1p 2 4 1 1 fmrpmos area=5
nm1n 2 4 0 0 fmrnmos area=2

.model fmrpmos ppis dtype=2 vt0=0.8 model=94 method=336 nodes=4
.model fmrnmos npis dtype=2 vt0=0.8 model=94 method=336 nodes=4

.dc vin 0 5 0.1
.end
```

Figure 16.1 CMOS inverter circuit schematic and netlist.

method ^2nd ^tauto trap

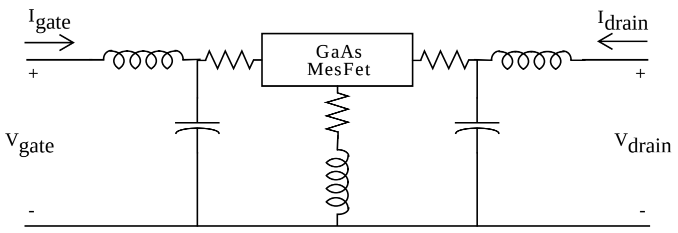
model consrh auger bgn conmob fldmob

For the DC analysis, the input voltage is swept from zero volts to five volts and the output voltage and supply current are monitored. The output waveform is as what would be expected from a CMOS inverter and is not shown.

16.3 High Frequency GaAs MESFET (AC Analysis)

The analysis of the GaAs MESFET mixed-mode simulation consists of finding the S-parameters of the two port network shown in Figure 16.2. SPICE can not directly find the S-parameters; however, it can easily find the Y-parameters with an AC perturbation applied to one input while the other is shorted. Two SPICE net lists are created, **mes1.spi** and **mes2.spi**. One perturbs port 1 of the network and the other perturbs port 2; hence each provides Y*1 and Y*2 respectively. One of the netlists is shown in Figure 16.2.

For this simulation, the file **mes.mesh.pis** contains the PISCES cards for creating the MESFET’s mesh during each device simulation. Note that nothing specific has to be written in the SPICE netlist



```
* MESFET circuit

vgg 9 0 dc 0.0 ac 1
vdd 10 0 dc 2.0

vgate 9 8 0
vdrain 10 1 0

ld 1 2 0.08n
rd 2 3 1.52
cd 2 0 212f

lg 8 7 0.105n
rg 7 4 0.92
cg 7 0 208f

ls 0 6 0.005n
rs 6 5 1.38

nm1 3 4 5 0 mes area=100

.model mes npis vto=1 maxtrys=10 nodes=4 dtype=4
.ac lin 40 1G 40G
.end
```

Figure 16.2 Two port network and netlist for AC analysis to find the Y-parameters

to differentiate using a previously created mesh file. In addition, the files **mes.model** and **mes.method** are also supplied and each contains one of the following lines respectively.

```
model conmob fldmob srh hypert impact
method trap ^2nd ^tauto itlim=30
```

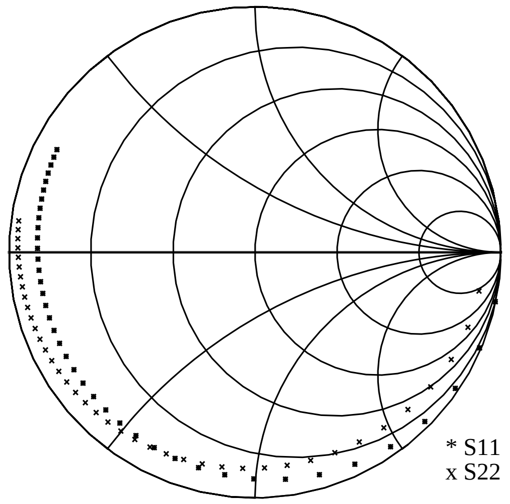


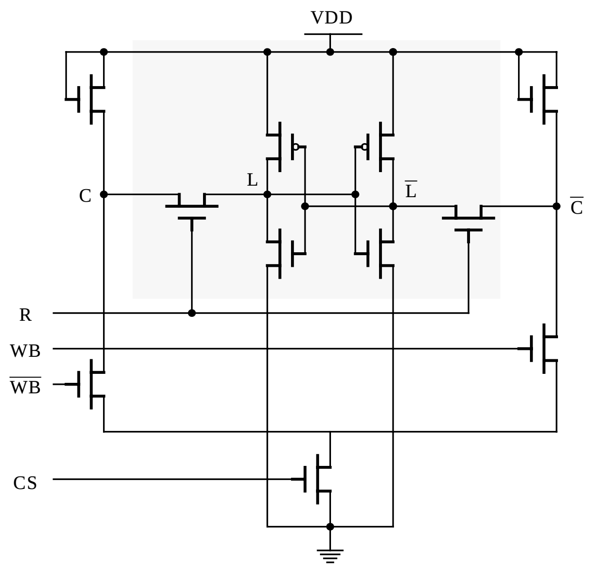
Figure 16.3 S-parameters for GaAs MESFET in the frequency range of 1GHz to 40GHz.

An AC analysis is performed at each input of the two port network with other shorted. The Y-parameters are calculated by taking the currents I_{gate} and I_{drain} and dividing by the AC excitation magnitude. A simple transformation yields the S-parameters for a 50 ohm cable. The results are displayed in Figure 16.3.

16.4 SRAM Cell (Transient Analysis)

This example shows a circuit simulated with transient analysis and utilizing a network of computers. Figure 16.4 shows the circuit diagram for the SRAM circuit. The shaded region is the six transistor SRAM cell and is simulated using numerical MOS devices. The unshaded region is the control circuitry and is simulated with MOS level 2 compact models.

The transient analysis of the circuit consists of a write and read cycle. The output of the flip-flop and the data lines are observed. Figure 16.5 contains the net list for the circuit. There is an extra compact NMOS transistor for leveling the charges of the data line capacitances. After a read or write cycle, this



CS:	Selects appropriate column
R:	Selects appropriate row
Read:	CS high, R high, WB low, $\overline{WB}$ low
Write 1:	CS high, R high, WB high, $\overline{WB}$ low
Write 0:	CS high, R high, WB low, $\overline{WB}$ high
Output:	Measured across C$\overline{C}$

Figure 16.4 Simplified circuit diagram of SRAM cell including control circuitry.

transistor turns on to allow the charge between the data lines to distribute evenly. The data lines must be equal before anymore actions can take place.

The models for the numerical devices are the same as those used for the CMOS inverter. The compact model is SPICE Level 2 model which closely represents the numerical model. The devices in the circuit are not the state-of-the-art, but they do provide a good example. The SPICE **.nodeset** line is

```
* supply line
Vdd 1 0 5

* the cell itself
NM1 4 5 1 1 fmrpmos area=2
NM2 5 4 1 1 fmrpmos area=2
NM3 4 5 0 0 fmrnmos area=4
NM4 5 4 0 0 fmrnmos area=4

* estimated parasitic of the cell
Ci1 5 0 0.01p
Ci2 4 0 0.01p

* pass gates
NM5 4 7 2 0 fmrnmos area=2
NM6 5 7 3 0 fmrnmos area=2

* control lines
M7 1 1 2 0 ndev l=2u w=25u
M8 1 1 3 0 ndev l=2u w=25u
M9 2 9 6 0 ndev l=2u w=25u
M10 3 8 6 0 ndev l=2u w=25u

* access line
M11 6 10 0 0 ndev l=2u w=50u

* level line
M12 2 12 3 0 ndev l=2u w=25u

* estimated load on lines
Crow 7 0 0.5pf
Ccol 2 0 2pf
Ccolb 3 0 2pf

* access control sources
VCS 10 0 pulse(0 5 0.2n 0.2n 0.2n 8n 20n)
VR 7 0 pulse(0 5 0.2n 0.2n 0.2n 8n 20n)

* data control sources */
VWBb 8 0 pulse(0 5 40.2n 0.2n 0.2n 8n 80n)
VWB 9 0 pulse(0 5 0.2n 0.2n 0.2n 8n 80n)
```

Figure 16.5 SPICE netlist for SRAM simulation.

```
* data line level sources
VL 12 0 pulse(0 5 9n 0.2n 0.2n 8n 20n)

.nodeset v(5)=0 v(4)=5 v(2)=3.5 v(3)=3.5
.tran 0.1n 40n

* compact models
.model pdev pmos
+ level = 2.000 vto = -0.5677 gamma = 0.51
+ phi = 0.73 tox = 2.5000E-08 nsub = 4.0251E+14
+ nfs = 6.8412E+11 xj = 2.5976E-8 ld = 1.8553E-07
+ uo = 1.705E+02 ucrit = 2.2295E+05 uexp = 0.2918
+ vmax = 6.8950E+04 neff = 9.285E+1 delta = 0.5587
+ cj = 2.7923E-04 cjsw = 2.1805E-10 pb = 0.7316
+ mj = 0.4729 mjsw = 0.2195 fc = 0.5
+ cgdo = 4.71e-10 cgso = 4.71e-10

.model ndev nmos
+ level = 2.000 vto = 0.6000 gamma = 0.4500
+ phi = 0.6000 tox = 2.5000E-08 nsub = 1.6300E+15
+ nfs = 3.0000E+11 tpg = 1.000 xj = 3.5000E-07
+ ld = 1.0000E-07 uo = 612.0 ucrit = 2.0000E+05
+ uexp = 0.2500 vmax = 5.8000E+04 neff = 45.00
+ delta = 3.900 cj = 1.4620E-04 cjsw = 3.1626E-10
+ pb = 0.6495 mj = 0.3640 mjsw = 0.2558
+ fc = 0.5000 cgdo = 4.39e-10 cgso = 4.39e-10

* numerical devices
.model fmrpmos npvm dtype=2 vt0=0.8 model=94 method=336 nodes=4
.model fmrnmos npvm dtype=2 vt0=0.8 model=94 method=336 nodes=4
```

Figure 16.5 Continuation of SPICE netlist for SRAM simulation.

used to set the flip-flop to an initial state rather than having it start indeterminately. The **.tran** card specifies the limits of the transient analysis.

The numerical device models are specified as **pvm**. As a result, each numerical device simulation is executed on a different node of a parallel virtual computer. The file **hostname.pvm** is provided with a list of hostnames for the nodes. The network uses a common file server whose disks are mounted on each of the remote hosts.

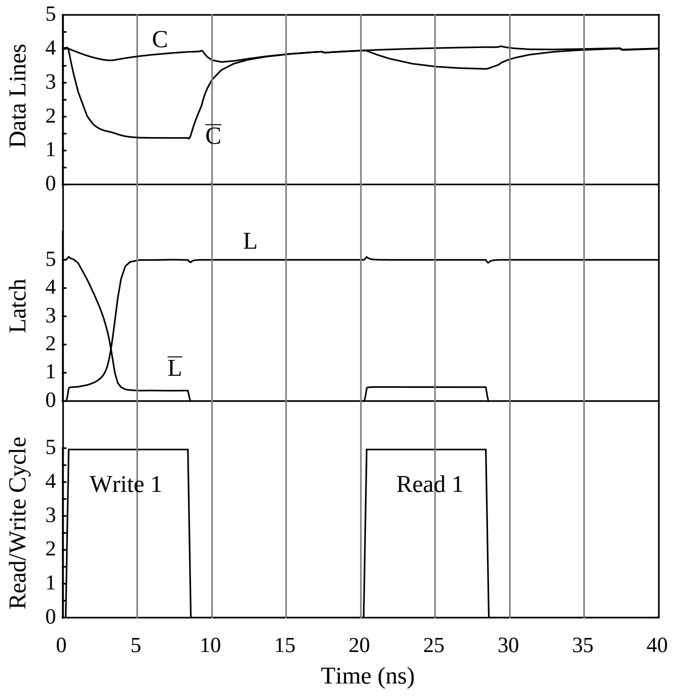


Figure 16.6 Read/Write cycle for SRAM cell.

Figure 16.6 shows the output waveforms of the SRAM cell. It is important to note how fast the difference between the data lines becomes small. This time determines the minimum read/ write cycle time.

16.5 GaAs/AlGaAs LED (Transient Analysis)

This final example involves a device with a floating node. Figure 16.7 shows the cross section of the cylindrical LED. The cathode contact is electrode number 1, the anode contact is electrode number 2, and the floating layer is electrode number 3. In order to improve convergence in the device simulation, the floating layer must switch from a voltage boundary condition to a current boundary condition when high currents are flowing. Refer to the paper by Dutton for a detailed discussion [7].

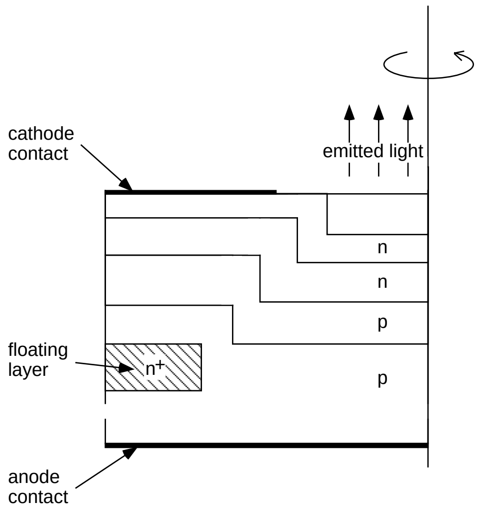
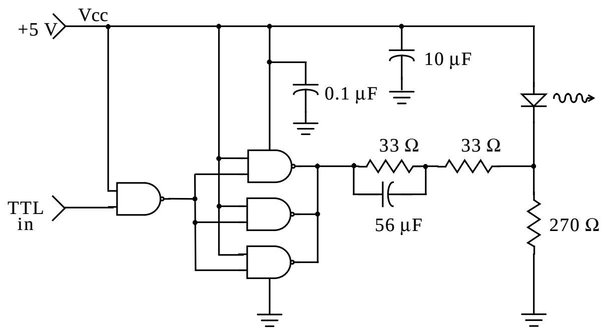


Figure 16.7 Cross section of LED structure that is used in a fiber optic transmitter circuit.

At low currents and biases, the voltage boundary condition introduces some error, but it is tolerable since that is not the range of interest for the operation of the device. For higher currents, the node is kept floating by placing a current boundary condition with a current source value of zero. In order to handle this switching, there are three special parameters.

This device is placed into the transmitter circuit shown in Figure 16.8. The purpose of this circuit is to take an ECL signal, convert to TTL, and either turn on or off the LED. The SPICE netlist for the circuit is also given.

First, one notes that this device is called using **ppis** model because a **ppis** device is defined such that the cathode is electrode 1 and the anode is electrode 2. An **npis** device has the reverse. In general, the n-polarity devices are such that the device is in its SPICE standard forward operating state.



```
* driver circuit for HP transmitter

* supply
Vcc 6 0 DC 5

Vin 1 0 pulse(5 0 1n 1.2n 1.2n 20n 40n)
Rin 1 2 1

* driver circuit
R8 2 3 33
C4 2 3 75p
R9 3 4 33
r10 4 0 270

Vhp1 6 5 DC 0
nhp1 4 5 hpled

.options abstol=1e-9
.model hpled ppis vt0=1.4 maxtrys=2 nodes=2 dtype=3 float=3
+ vbc=1.0 bcdep=21

.tran 100p 40n 0 200p
.end
```

Figure 16.8 Circuit schematic and netlist for LED transmitter simulation.

The rest of the parameters on the **.model** line are set to account for the heterostructure device. **vt0** is set to turn on the device. The **maxtrys** parameter limits the maximum number of tries to two. This parameter is set low because the cutting of the bias step does not take into account a changing boundary condition. The number of nodes is set to the actual number of contacts in the SPICE circuit. The floating node is not connected to anything and hence, SPICE does not need to know about its existence.

The three parameters **float**, **vbc**, and **bcdep** specify the node and the switching point for the boundary condition.The floating node is electrode number 3 as given by the **float** parameter. The voltage boundary condition is used when V_{21} is below 1.0 volts and the current boundary is used when V_{21} is greater than or equal to 1.0 volts. The parameter **bcdep** specifies V_{21} as the controlling voltage and the parameter **vbc** specifies the voltage of V_{21} when the boundary condition change is to take place.

The device uses cylindrical coordinates and hence, the area factor is allowed to default to 1.0. The mesh is created each time the device is solved because the version of PISCES being used can not save the heterostructure parameters. Hence, the file **hpled.mesh.pis** must be supplied. In addition, the model and method cards are given in the files **hpled.model** and **hpled.method** and are as follows.

```
models fldmob auger conmob rad consrh
method itlim=50 p.tol=5.e-5 c.tol=5e-5 ^2nd ^tauto
```

A communications circuit engineer is most interested in the turn-on turn-off characteristics of the diode. Therefore, the transient simulation consists of a pulse input to switch the diode on and back off. The TTL nand gates used for supplying the current are substituted with a voltage source and a resistor. The voltage source takes into account the finite rise and fall time of the nand gates while the resistor simulates the output resistance of the nand gate.

Figure 16.9 shows the response of the diode. This current is directly related to the amount of photons out for the device.

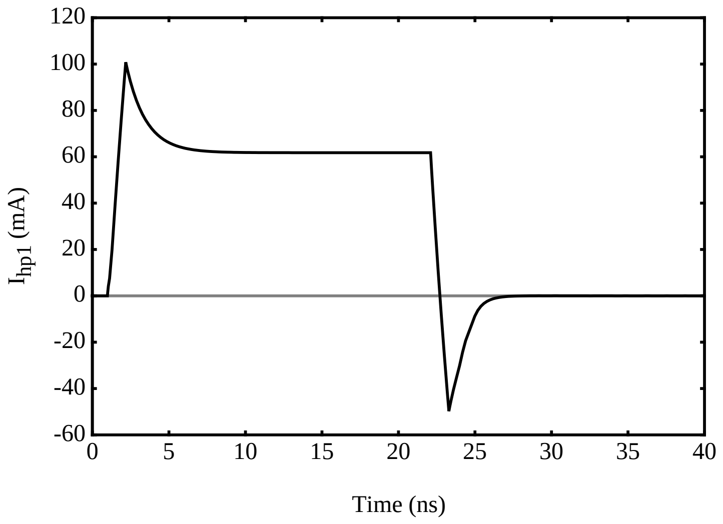


Figure 16.9 Time transient response of the current flowing through LED in a transmitter circuit.

CHAPTER 17
System Reconfiguration for a Different Device Simulator

17.1 Changes in SPICE

This chapter addresses the changes needed in SPICE so that a numerical device simulator may be added to the in-house version of SPICE. These instructions are based upon the 3f release of SPICE. The configuration for the 3e release is unknown at the writing of this manual. The 3d release has a slightly different configuration and those differences are described in this chapter. In order to configure 3e or 3c release of SPICE, the administrator can use the descriptions in this chapter and also refer to the SPICE software documentation [8][9].

When referring to a directory, the home directory is to be taken as **spice3f/src** unless otherwise specified.

17.1.1 New Files

Table 17.1 contains a manifest of files needed for including the numerical devices.

Table 17.1

inp2n.c	makedefs	npisc.c
npiscacload.c	npiscask.c	npiscconvtest.c
npiscctrl.c	npiscdefs.h	npiscdelete.c
npiscexec.c	npiscexecprl.c	npiscexecpvm.c
npiscexecseq.c	npiscgethost.c	npiscgetic.c
npiscitf.h	npiscload.c	npiscmdelete.c
npiscmethod.c	npiscmodel.c	npiscmparam.c
npiscmvfiles.c	npiscparam.c	npiscpvmmesg.c
npiscpzload.c	npiscreadpvm.c	npiscreadstanford.c
npiscsetup.c	npiscslave.c	npisctemp.c
npisctrunc.c	npiscwritestanford.c	numdevdefs.h
makefile.3d2		

These are subject to change for later versions, but differences should not be significant.

17.1.2 Steps for Adding Numerical Devices to SPICE

1. In **bin/config.c** there are two sets of include statements for the **devitf.h** files. Add each of the following lines to the appropriate list.

```
#include “npiscitf.h”
#include “npisc/npiscitf.h”
```

2. Also in **bin/config.c**, there is a series of #ifdef structures to which one adds the following.

```
#ifdef DEV_npisc
   &NPISCinfo
#endif
```

3. In **include/inpdefs.h**, there are two listings for the routines that are used to read an individual SPICE device type. The following lines are added to the appropriate list.

   ```
   void INP2N( GENERIC*, INPtables*, card* );
   void INP2N();
   ```

4. The following line is added to the subroutine inp_numnodes() in **lib/fte/subckt.c**. The number that is returned is the maximum number of nodes that the numerical device can support.

   ```
   case: ‘n’: return(10);
   ```

5. Add the file **inp2n.c** to the directory **lib/inp**. Modify the **makedefs** file in that directory to include **inp2n.c** in the CFILES definition and to include **inp2n.o** in the COBJS definition. Finally, add the following entry to the list of files to make.

   ```
   inp2n.o: inp2n.c
   ```

6. The following lines are added to **lib/inp/inpdomod.c.** These are the valid model types available for a numerical device. The administrator can eliminate an execution method by not including it in this listing.

   ```
   } else if(      (strcmp(typename,”pis”) == 0)        ||
                   (strcmp(typename,”npis”) == 0)       ||
                   (strcmp(typename,”ppis”) == 0)       ||
                   (strcmp(typename,”pvm”) == 0)        ||
                   (strcmp(typename,”npvm”) == 0)       ||
                   (strcmp(typename,”ppvm”) == 0)       ||
                   (strcmp(typename,”prl”) == 0)        ||
                   (strcmp(typename,”nprl”) == 0)       ||
                   (strcmp(typename,”pprl”) == 0)       ){
       type = INPtypelook(“Npisces”);
       if(type < 0) {
          err=INPmkTemp(“Device type Npisces not available in this binary\n”);
       }
       INPmakeMod(modname,type,image);
   ```

7. The following line identification is added to **lib/inp/inppas2.c**

   ```
   case ‘N’:
       INP2N(ckt,tab,current,gnode);
       break;
   ```

8. Make the directory **lib/dev/npisc** and copy all the files into this directory except for **inp2n.c** which should be placed in **lib/inp**.
9. In the main configuration file, one needs to include “npisc” in the listing of devices to compile. This configuration file is located in the **conf** directory which is at the same level as the **src** directory**.**
10. The user should clean everything and recompile SPICE. There may be dependencies that are not taken into account when certain header files are modified.

17.1.3 Increasing Number of Nodes for a SPICE Device

The released version of SPICE is configured for five nodes per device. For numerical devices, the maximum number of nodes is set to ten (This number corresponds to the maximum number of electrodes that Stanford PISCES allows). Because of the way new devices are interfaced to SPICE, the number of device nodes is set in the code. Fortunately, expanding the number of nodes is trivial [9].

1. Modify the file **lib/ckt/cktbindn.c** to include the cases for nodes six through ten. Follow the pattern already prescribed for nodes one through five.
2. Modify the file **include/gendefs.h** to include the additional nodes six through ten in the pattern prescribed by nodes one through five.
3. The user should clean everything and recompile SPICE from scratch. There may be dependencies that are not taken into account when certain header files are modified.

17.1.4 Configuring NPISC for Use with PVM

In order to use PVM to execute a device simulation, the following changes are implemented. These changes configure the routines that access PVM and compile the slave processes. PVM 3.0 or higher is assumed to be installed on appropriate machines and a working knowledge of PVM is assumed [5].

1. In the **conf/defaults** define the variable PVM_DIR to the PVM home directory. For example:

   ```
   PVM_DIR = /home/spice/pvm3
   ```

2. For each specific architecture file define the variable PVM_ARCH to the correct PVM bin directory. For example, in the **conf/sun4** file one includes the following line.

   ```
   PVM_ARCH = SUN4
   ```

   In addition, the following variables are also defined in the individual architecture configuration files.

   ```
   PVM_BIN = $(PVM_DIR)/bin/$(PVM_ARCH)
   PVM_TYPE = PVM_PUBLICDOMAIN
   ```

   The PVM_TYPE variable is included so that other version of PVM can used. For example, IBM has their own version of PVM. If the mixed-mode code is changed to include this version, all changes should be enclosed by an #ifdef structure using a new name for PVM_TYPE.

3. Copy (or link) the files **npiscslave.c** and **npiscctrl.c** to the directory **src/bin**.

4. The **makedefs** file in the directory **src/bin** has the following lines added so that the **npiscslave** and **npiscctrl** routines are compiled and installed correctly. In the list of CFILES, add the files **npiscslave.c** and **npiscctrl.c**. In the list of INSTALL_TARGETS add the following lines.

   ```
   $(PVM_BIN)/npiscslave \
   $(PVM_BIN)/npiscctrl
   ```

   A variable COBJS is defined below CFILES as follows.

   ```
   COBJS          = npiscslave.o npiscctrl.o
   ```

   Finally, add the following line to the list of object files.

   ```
   npiscslave.o: npiscslave.c
   npiscctrl.o: npiscctrl.c
   ```

5. The **makeops** file in the directory **src/bin** is modified to include the following lines. Around line number 14 add the following.

```
$(PVM_BIN)/npiscslave: npiscslave
                rm -r $@
                cp $? $@
$(PVM_BIN)/npiscctrl: npiscctrl
                rm -r $@
                cp $? $@
```

   Around line number 55 add:

```
npiscslave.o: npiscslave.c
                $(CC) -c $(CFLAGS) -I$(PVM_DIR)/include \
                  -I$(SRC_DIR)/../lib/dev/npisc \
                  -D$(PVM_TYPE) \
                  $(SRC_DIR)/npiscslave.c
npiscctrl.o: npiscctrl.c
                $(CC) -c $(CFLAGS) -I$(PVM_DIR)/include \
                  -I$(SRC_DIR)/../lib/dev/npisc \
                  -D$(PVM_TYPE) \
                  $(SRC_DIR)/npiscctrl.c
```

   In the LIBS declaration around line number 150 add the following to include the PVM library:

```
$(PVM_DIR)/lib/$(PVM_ARCH)/libpvm3.a
```

   Finally, around line 160, add the following compilations:

```
npiscslave: npiscslave.o $(OBJ_TOP)/lib/dev.a \
                -@rm -f npiscslave
                $(CC) -o $@ npiscslave.o $(OBJ_TOP)/lib/dev.a \
                  $(PVM_DIR)/lib/$(PVM_ARCH)/libpvm3.a
npiscctrl: npiscctrl.o $(OBJ_TOP)/lib/dev.a \
                -@rm -f npiscctrl
                $(CC) -o $@ npiscctrl.o $(OBJ_TOP)/lib/dev.a \
                  $(PVM_DIR)/lib/$(PVM_ARCH)/libpvm3.a
```

17.1.5 Different Versions of SPICE

There are some difference between SPICE3f and SPICE3d. This section attempts to clarify as many of those differences as possible; however, it may not be all inclusive and it does not include the changes for configuring the compilation for use with PVM. For versions other than 3f and 3d the administrator should refer to the implementation guides [8][9]. In addition, the BJT device provides an excellent basis for how a new SPICE device is installed.

The first and most obvious difference is that SPICE3d capitalizes the identifier of each module. As a result, anywhere npisc is specified, it becomes NPISC. For example **npiscload.c** becomes **NPISCload.c** and DEV_npisc becomes DEV_NPISC.

Following are the differences between SPICE3f and SPICE3d given in the same sequence as the original installation instructions.

1. Instead of changing the **bin/config.c** file, one needs to make the given change in the **bin/spice.c** file.
2. Instead of changing the **bin/config.c** file one needs to make the change in the **include/devices.h** file.
3. Instead of changing the **include/inpdefs.h** file, one needs to make the change in the **include/INPdefs.h** file.
4. One needs to change the **lib/FTE/subckt.c** file instead of the **lib/fte/subckt.c** file.
5. The file **INP2N.c** should be added to the **lib/INP** directory and the appropriate change should be made to the **Makefile** in that directory.
6. Change the **lib/INP/INPdomodel.c** instead of the **lib/inp/inpdomod.c** file.
7. Add the new line to **lib/INP/INPpas2.c** instead of **lib/INP/INPpas2.c**.
8. The directory for the numerical device is called **lib/DEV/NPISC**, the files should have “npisc” changed to “NPISC”, and **makefile.3d2** is renamed to **Makefile**. In addition, there are a few changes to the code as outlined here. First, the include filenames need to be changed since SPICE3d uses a different

naming convention as described previously. Second, **NPISC.c** and **NPISCitf.h** are not in the correct form. The user should examine **BJT.c** and **BJTitf.h** for reference, but the following paragraphs point out the changes that are required.

In **NPISC.c**, the variables NPISCiSize and NPISCmSize are defined for the size of the instance and model. SPICE3d does not use these variables and consequently, they should be removed from the file. Likewise, in **NPICSitf.h** the extern lines for these variables should be removed and the other reference to these variables should be replaced with an explicit sizeof() statement. The code may work if these variables are in the code, but it has not been tested. Also in **NPISCitf.h**, there is an #ifdef DEV_npisc statement surrounding the code. This statement and the corresponding #endif at the end of the code should be removed.

SPICE3f has a location for an additional function called DEVunsetup. This function is not in SPICE3d and is not used by the DEVice NPISC. Therefore, this line is removed from the function list in **NPISCitf.h**. This function is defined as NULL, between the two definitions for NPISCsetup. Refer to the **BJTitf.h** file for a comparison. Also in **NPISCitf.h**, there is a line DEV DEFAULT which is deleted. Finally, in **NPISCdefs.h**, the NPISCstate variable in the NPISCinstance structure is moved from its location at the beginning and placed at the end of the structure. SPICE3f requires it to be there while SPICE3d requires it not be there.

If SPICE3d does not work correctly after making these changes, refer to the equivalent files in the **lib/DEV/BJT** directory. The critical files and ones most likely to have an error are **NPISCitf.h**, **NPISCdefs.h**, and **NPISC.c**.

9. For SPICE3d, the NPISC device must be defined in the variables SUBDIRS and MSCSUBDIRS in **lib/DEV/Makefile**.

10. The user should clean everything and recompile SPICE to make sure that all dependencies are taken into account.

11. PVM has not been tested with SPICE3d and the procedure for adding the compilation of **npiscslave.c** and **npicsctrl.c** is unknown. One can attempt to define the appropriate variables as discussed in Section 17.1.4 on page 306 and compile the routines manually.

17.2 Adding Another Device Simulator

This section describes how to add another device simulator such that it can be used within the mixed-mode environment.

17.2.1 Changes to Device Simulator

This section discusses the changes required for the device simulator. SPICE determines a bias and the mixed-mode interface writes an input deck for the device simulator. The interface executes the device simulator using that input deck. In return, the device simulator must perform the following.

1. It must be able to load a mesh and a previous solution at some DC point or at some time instant. An important advantage is to be able to load the last two solutions and then use a projection to find an initial guess at the new bias point.
2. Given a DC bias, a transient bias and a Δt, or an ac frequency the device simulator needs to solve for the current vector and conductance (susceptance) matrix. The conductance matrix is for the frequency given or at some low frequency value. Refer to “Equations for Device Simulation” on page 269 for more information.

Most simulators posses these attributes except for finding the low frequency conductance matrix in DC and transient analysis. The changes to obtain this information is minimal and should not be a major problem.

17.2.2 Changes to Mixed-Mode Interface

As a basis, one can follow the currently configured Stanford version of PISCES. There are a few small changes in a couple of routines and a number of new routines that must be written. All changes are specific to the device simulator.

npiscdefs.h

This file is used to define the new device simulator and to give it an appropriate identification number. This number corresponds to the **level** given on the **.model** card. All code specific to that device simulator should be surrounded by an #ifdef structure so that it may easily be excluded in a compilation.

npiscexec.c

The purpose of this routine is to execute the device simulations. SPICE passes this routine the voltage boundary conditions on each electrode of the numerical device. This routine returns the solution for the current vector and admittance matrix.

npiscexec calls two subroutines that are specific for the device simulator as determined by the **level** value. One routine writes the input deck for the device simulator and the other reads the solution from the files created by the device simulator. For Stanford PISCES, these routines are as follows and are located in a "case" structure in the file.

```
int NPISCwriteStanford( NUMDEVbias *)
int NPISCreadStanford( NUMDEVbias *, NUMDEVsoln *)
```

npiscwriteMYSIMULATOR.c

This routine is responsible for writing the device simulator input deck. A pointer to a structure containing the information about the bias is passed to the routine. An error code or 0 is returned from this subroutine call.

The bias structure contains information about the SPICE instance name, SPICE model name, voltage bias, time step, frequency, temperature, etc. (Refer to **numdevdefs.h** for all the information contained within the structure.) This information should be sufficient to write an input deck for the device simulator.

There are a couple of requirements on file naming conventions.

1. The solution for the current bias must be saved as:

   [Instance Name].solution.prev0

Refer to the files **npiscmvfiles.c**, **npiscwritestanford.c**, and **npiscexec.c** for descriptions of how these files are manipulated and how to utilize these solution files. All routines are well documented.

2. Every file created must start with the instance name. This provides a unique way for identifying each file and avoids clobbering other files.
3. In “Running a Mixed-Mode Simulation” on page 286, a file naming convention is presented. Any new device simulator should follow the given convention to avoid confusion.

The **npiscwritestanford.c** file can be used as a starting point and modified for the needs of the new device simulator. The code is well documented and should not be difficult to follow.

npiscreadMYSIMULATOR.c

This routine is responsible for reading the results of the device simulation. When creating the input deck, one should have specified output files for saving the current vector and conductance matrix of the device solution. Given the pointer to the bias data structure which contains the instance name for identification and given a pointer to a solution data structure, this routine reads the simulation results and stores them in the solution data structure. It must also multiply by the **area** factor if it is not done during the device simulation.

The **npiscreadstanford.c** file can be used as a starting point and modified for the needs of the new device simulator. The code is well documented and should not be difficult to follow.

npiscmethod.c

In PISCES, there is a card called “method” that is used to define the numerical methods during the device simulation. This method is the same for all device simulations for a specific model. Hence, the mixed-mode interface has been designed to accept this method card in two formats (Refer to “Method and Model Parameters” on page 288 for more information.), store in its internal structures, and pass it to the **npiscwriteMYSIMULATOR.c** routine through the bias data structure. Since other device simulators have something similar, this routine is modified to take that into account.

There are two ways to determine the method line. The easiest way to implement the method card is to read it from a file named **[Model Name].method** which is the default in **npiscmethod.c**.

In order to eliminate an extra file, a binary encoded **method** parameter is added to the SPICE **.model** parameters. Essentially, a binary number (written in decimal format on the **.model** card) determines whether a logical expression on the PISCES method line is true or false. The **numdevdefs.h** file contains defined variables for all the possible actions. Each action has a number in base two that corresponds to a bit in a long integer.

The **npiscmethod.c** routine takes the integer value given on the **.model** card and converts that to character string for the numerical device input deck. This file and the **numdevdefs.h** file are well documented and the programmer should not have a problem understanding how to modify this routine. If there is no method line, everything is initialized to NULL.

npiscmodel.c

This file is analogous to the **npiscmethod.c** file except it does the processing for the model card in the numerical device input deck. Refer to this routine and the **numdevdefs.h** file for more information.

After all changes have been made, the new files can be added to the **makedefs** file so that they are compiled and linked when SPICE is built. Once included, the new device simulator can be used in a mixed-mode simulation simply by specifying the proper **level** on the **.model** card in the SPICE netlist.

References

[1] Yu, Zhiping, Robert W. Dutton, and Hui Wang. “A Modularized, Mixed IC Device/ Circuit Simulation System.” *Proceedings of the Synthesis and Simulation Meeting and International Exchange.* Kobe, Japan. April 1992.

[2] McCalla, William J. *Fundamentals of Computer-Aided Circuit Simulation.* Boston: Kluwer Academic Publishers, 1993.

[3] Mayaram, Kartikeya and Donald O. Pederson. “Coupling Algorithms for Mixed-Level Circuit and Device Simulation.” *IEEE Transactions on Computer Aided Design.* Vol. II, No 8. August 1992. pp 1003-1012.

[4] Rollins, Gregory J. and John Choma. “Mixed-Mode PISCES-SPICE Coupled Circuit and Device Solver.” *IEEE Transactions on Computer Aided Design.* Vol. 7, No. 8, August 1988. pp 862-867.

[5] Geist, Al, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam. *PVM 3 User’s Guide and Reference Manual.* Oak Ridge: Oak Ridge National Laboratory, 1993.

[6] Johnson, B., T. Quarles, A. R. Newton, D.O Pederson, and A. Sangiovanni-Vincentelli. *SPICE3 Version 3f User’s Manual.* Berkeley: Regents of the University of California, 1992.

[7] Dutton, R. W., F. Rotella, Z. Sahul, L. So, and Z. Yu. “Integrated TCAD for OEIC Applications.” International Symposium on Optoelectronics for Information and Microwave Systems. *Proceedings of the SPIE - The International Society for Optical Engineering.* Los Angeles, CA. January 1994.

[8] Quarles, Thomas L. “Adding Devices to SPICE3.” Memorandum No. UCB/ERL M89/45. Berkeley: University of California, 1989.

[9] Quarles, Thomas L. “SPICE3 Implementation Guide.” Memorandum No. UCB/ERL M89/44. Berkeley: University of California, 1989.