A THREE DIMENSIONAL NUMERICAL SIMULATION OF SHORT CHANNEL MOSFET'S WITH THE EFFECTS OF GATE OXIDE CHARGE

By

Russel Jacob Baker, B.S.E.

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science in Engineering, M.S.E.

in

Electrical Engineering

Department of Computer Science and Electrical Engineering
University of Nevada, Las Vegas
May, 1988

The thesis of Russel Jacob Baker for the degree of Master of Science in Engineering in Electrical Engineering is approved.

[illegible]

Chairperson, [Abdol Rahim Khoie]

[illegible]

Examining Committee Member, [Dwight S. Heim]

[illegible]

Examining Committee Member, [Ashok Iyer]

[illegible]

Graduate Faculty Representative, [Sadanand Verma]

[illegible]

Graduate Dean, [Ronald W. Smith]

University of Nevada
Las Vegas, Nevada
May, 1988

ABSTRACT

A THREE DIMENSIONAL NUMERICAL SIMULATION OF SHORT CHANNEL MOSFET'S WITH THE EFFECTS OF GATE OXIDE CHARGE*

Russel Jacob Baker, M.S.E.

University of Nevada, Las Vegas, 1988

A three dimensional model is described for a short channel MOSFET with the effect of trapped oxide charge. The model uses the finite difference scheme to solve the system of coupled nonlinear partial differential equations describing the transport of carriers in a semiconductor. A field dependent mobility is used. A MOSFET with an oxide charge of 10^{10}cm^{-2} is found to have a drain current of 1.13224 mA at a gate voltage of 0.4 volts, while an oxide charge of 10^{11}cm^{-2} gives a current of 1.13232 mA which is not significant under high drain bias. The short channel effects such as, failure of current saturation due to punch-through, are illustrated with I-V plots. The 3-dB frequency for a short channel MOSFET is found to be 4.2 MHZ, although the reduction of channel length does increase 3 dB frequency the punch through effect puts a limit on channel length.

* This work was supported by the United States Army Research Office Grant # DAAL 03-87-G0004.

ACKNOWLEDGEMENTS

The author wishes to acknowledge the United States Army Research Office for providing the computation environment that made the numerical modeling of this Thesis possible. I am grateful to my committee members Dr. D. Heim, Dr. A. Iyer and Dr. S. Verma for reviewing my Thesis. I would like to extend my deepest appreciation to my advisor Dr. A. Rahim Khoie for his guidance and support during the progress of this Thesis and for his invaluable advise throughout the course of my undergraduate and graduate education.

Last but not least I would like to thank my wife Julie for her patience, understanding and love during the long hours I spent working on this thesis.

TABLE OF CONTENTS

	Page
ABSTRACT	iii
ACKNOWLEDGMENTS	iv
LIST OF SYMBOLS	viii
LIST OF FIGURES	xiii
CHAPTER 1 - INTRODUCTION	1
1.1 Short Channel MOSFET's	1
1.2 Trapped Oxide Charge	2
1.3 Organization of this Thesis	2
CHAPTER 2 - LITERATURE REVIEW	4
CHAPTER 3 - THE BASIC SEMICONDUCTOR EQUATIONS	7
3.1 Equation Formation	7
3.1.1 Poisson's Equations	9
3.1.2 Current Density Equations	10
3.1.3 Current Continuity Equations	12
3.2 Boundary Conditions	13
3.2.1 Source and Drain Contacts	13
3.2.2 Gate Contact and Interface Trapped	

Charge 14

3.2.3 Artificial boundaries 16

CHAPTER 4 - THE PHYSICAL PARAMETERS 18

4.1 Carrier Mobility 18

4.2 Carrier Generation-Recombination 21

CHAPTER 5 - THE DISCRETIZATION OF THE SEMICONDUCTOR EQUATIONS 26

5.1 Finite Differences 27

5.1.1 Poisson Equation 28

5.1.2 Current Continuity Equations 30

5.1.3 Boundary Conditions 35

5.1.4 Scaling 36

5.2 Designing a Mesh 38

CHAPTER 6 - SOLUTION OF THE DISCRETE SEMICONDUCTOR EQUATIONS 39

6.1 Newtons Method 39

6.2 Numerical Instability and Convergence Problems 42

6.3 Initial Guess 43

CHAPTER 7 - RESULTS OF SIMULATIONS 45

7.1 Simulation of an Unbiased MOSFET 45

7.2 Simulation of an Biased MOSFET 47

7.3 Effects of Gate Oxide Charge 50

7.4 MOSFET Characterization 50

CHAPTER 8 - DISCUSSIONS AND CONCLUSIONS 52

8.1 The Instabilities 52

8.2 Distribution of Potential, Carrier Concentrations and Current Densities 53

8.3 I_D-V_D Characteristics 55

8.4 Effects of Oxide Charge 57

8.5 Transconductance and Input Capacitance 58

8.6 Future Work 60

TABLES 61

FIGURES 62

REFERENCES 89

LIST OF SYMBOLS

$A_{n,p}$	Coefficient for determining μ^{LIE}.
$B_{n,p}$	Coefficient for determining μ^{LIE}.
$B(x)$	Bernoulli function, $x \cdot (e^x - 1)^{-1}$.
C	$N_D - N_A$.
C_n^{ref}	Coefficient of scattering for electrons.
C_p^{ref}	Coefficient of scattering for holes.
C_o	Maximum doping value in simulation domain.
$\mathbf{D}$	Electric displacement vector.
D_n	Electron Diffusivity.
D_p	Hole diffusivity.
D_o	Maximum diffusion coefficient in domain.
$\mathbf{E}$	Electric field vector.
E_{Fn}	Fermi energy for electrons.
E_{Fp}	Fermi energy for holes.
$E_{n,p}^{crit}$	Critical E field for Auger process to begin.

E	Magnitude of electric field vector.
$F_{n,p}$	Coefficient for determining μ^{LIE}.
$F_1(\psi, n, p)$	Poisson's equation.
$F_2(\psi, n, p)$	Current continuity equation for electrons.
$F_3(\psi, n, p)$	Current continuity equation for holes.
G^{II}	Generation due to impact ionization.
$\mathbf{J}_n$	Current density vector for electrons.
$\mathbf{J}_p$	Current density vector for holes.
k	Boltzmann constant, $1.38 \cdot 10^{-23}$VAsK^{-1}.
L	Channel length.
n	Electron concentration.
N_D	Number of donor atoms.
N_A	Number of acceptor atoms.
n_i	Intrinsic carrier concentration.
n_{po}	Equilibrium value of electrons in p type

p_{po}	Equilbrium value of holes in p type semiconductor.
q	Electronic Charge, $1.6 \cdot 10^{-19}$C.
Q_{int}	Charge at semiconductor/oxide interface.
R	Recombination/Generation difference.
R^{AU}	Recombination due to Auger process.
rj	Junction depth.
S_n	Scattering probability for electrons.
S_p	Scattering probability for holes.
T	Temperature in Kelvin.
t_{ins}	Thickness of the oxide.
Vg	Gate potential.
x_o	Debye length.
Z	Channel width.
$\alpha_{n,p}$	Ionization coefficients for Auger process.
$\alpha_{n,p}^{inf}$	Ionization coefficients for infinite E field.
$\delta \psi^r$	Difference in successive solutions of F_1.

ϵ_{sem}	Semiconductor permittivity.
ϵ_{ins}	Insulator permittivity.
λ^2	Scaling multiplier of Poissons equation.
Φ_F	Fermi potential under equilibrium.
Φ_{Fn}	Quasi Fermi potential for electrons.
Φ_{Fp}	Quasi Fermi potential for holes.
ρ	Total space charge.
$\sigma_{p,n}^{SRH}$	Shockley-Read-Hall capture rates.
$\sigma_{n,p}^{AU}$	Auger coefficients.
$\tau_{p,n}$	Carrier lifetimes.
μ_n	Electron Mobilty.
μ_p	Hole mobility.
μ_n^L	Lattice scattering mobility for electrons.
μ_p^L	Lattice scattering mobility for holes.
$\mu_{n,p}^{LI}$	Ionized and lattice scattering mobility.

ψ^{xx}	Second order derivative of ψ in x direction.
ψ	Electrostatic potential.
ψ_a	Applied potential at ohmic contact.
ψ_{bi}	Built in potential.
ψ_D	Effective potential of drain at the gate.
ψ^{r+1}	Updated solution for Possion's equation.
ψ^r	Previous solution of Poisson's equation.
ω	Successive under relaxation coefficient.

LIST OF FIGURES

Figure 1	MOSFET geometry used in simulation.
Figure 2	Simulation domain.
Figure 3	Doping profile for device 1.
Figure 4	Electrostatic potential for device 1.
Figure 5	Concentration of electrons for device 1.
Figure 6	Concentration of holes for device 1.
Figure 7	Electron mobility for device 1.
Figure 8	Hole mobility for device 1.
Figure 9	Net generation/recombination rate device 1.
Figure 10	Magnitude of electron current density for device 1.
Figure 11	Doping concentration for device 2.
Figure 12	Electrostatic potential for device 2.
Figure 13	Concentration of electrons for device 2.
Figure 14	Concentration of holes for device 2.
Figure 15	Electron Mobility for device 2.
Figure 16	Hole mobility for device 2.
Figure 17	Net recombination/generation rate for device 2.
Figure 18	Magnitude of electron current density for device 2.

Figure 19	Doping concentration for device 3.
Figure 20	Electrostatic potential for device 3.
Figure 21	Electron concentration for device 3.
Figure 22	I_D vs. V_D for varying V_G.
Figure 23	I_D vs. V_D for varying $V_G$without recombination term.
Figure 24	I_D vs. V_G with effects of oxide charge.
Figure 25	Transconductance vs. drain voltage.
Figure 26	Input capacitance vs. drain voltage.
Figure 27	Diverging Solution.

CHAPTER 1 - INTRODUCTION

This thesis deals with the modeling of short channel metal oxide semiconductor field effect transistors (MOSFET's) with the effects of gate oxide charge. Present day VLSI presents the problem of smaller and smaller device geometries with higher density. The "trial and error" method of the past is even less desirable due to the increase cost of production and development time. For these reasons computer aided design is essential to the device engineer.

1.1 Short Channel MOSFET's

The short channel effect comes from reducing the channel length making it comparable to the source and drain depletion layer widths. This reduction in turn causes the potential distribution in the channel to depend on both the transverse field, controlled by the gate voltage, and the longitudinal field, controlled by the drain bias. In this situation the transverse field is no longer much greater than the longitudinal field. This two dimensional effect results in a degradation of the subthreshold behavior, dependence of the threshold voltage on channel length and biasing voltages, and failure of current saturation due to punch-through.

1.2 Trapped Oxide Charge

The three dimensional model's of MOSFET's developed in the past have not taken the trapped oxide charge into consideration. The effect of this trapped oxide charge is very important when dealing with short channel MOSFET's due to the small surface area under the gate region. Inclusion of this effect has made the modeled device behave closer to the actual device.

1.3 Organization of this Thesis

Chapter 3 of this thesis develops the basic transport equations of carriers in the semiconductor using a semiclassical approach. These equations together with the appropriate boundary conditions, also discussed in this section, give a complete description of carrier motion.

Chapter 4 discusses the physical parameters, carrier mobility and carrier generation - recombination. The effects of lattice and ionized impurity scattering as well as the carrier velocity saturating is used in the modeling of mobility. The modeling of carrier generation - recombination takes under consideration Shockley - Read - Hall, Auger and impact ionization processes.

The method of finite differences used to solve the system of coupled nonlinear differential equations is discussed in chapter 5. The transport equations are transformed from continuous to discrete form.

The solution of these discretized semiconductor equations is discussed in chapter 6. A Newton iteration is used with successive under relaxation on a reduced problem set. This method reduces and simplifies the large number of equations that needs to be solved for each iteration of Newton's method.

The results of three simulations are given in chapter 7. The results are discussed in chapter 8. For each MOSFET under consideration plots of electrostatic potential, electron and hole concentrations as well as several other plots of interest to the device engineer are given.

Possible future work in this area would be in the simulation of transient behavior. The model that has been developed could be used, with little modification, for this type of simulation. Reference words: MOSFET, Short Channel, Numerical Simulation, Jacobian, Oxide Trapped Charge, Three-Dimensional, Semiconductor, Solid State.

CHAPTER 2 - LITERATURE REVIEW

Semiconductor device modeling started with Gummel[7] who modeled a one-dimensional bipolar transistor. The algorithm which is called Gummels method has been used extensively throughout semiconductor device modeling history. The advantage of this method is that it decouples the semiconductor equations. Gummel's method proceeds as follows; given an initial guess for the electron and hole concentrations Poisson's equation is solved for potential. The solution of Poisson's equation is then used to solve the current continuity and current density equations for an improved estimate of the electron and hole concentrations. The new updated carrier concentrations are then used to solve Poisson's equation. This loop continues until the difference between successive solutions is below the accuracy level desired. Gummel's method has been used in two dimensions by Slotboom[8],[9], Mock[10],[11], Heinier[12], Manck[13] and others. The main difference between these attempts to solve the semiconductor equations is in the choice of varibles and the treatment of the carrier continuity equations. Fitchner et. al. [3] have used the exponentials of the quasi Fermi levels as the dependent variables in the current density equations while others[34], [35] have used the quasi Fermi potentials. A comparison of the two sets is done in [36]. Some authors[37], [38] have used the stream function as a variable in the continuity equations. The main assumption that exists is that the recombination term is zero.

There has been many 2-D simulations of MOS devices [14],[15],[16],[17]. Each of these papers focuses on a different aspect of MOSFET's. Power MOSFET's are discussed in [14], a finite element analysis of a MOSFET in [15], analysis of breakdown phenomena in MOSFET's in [16] and modeling of the avalanche effect in MOSFET's in [17].

Three dimensional simulation has been carried out by [18], [19],[20]. It is stated in these papers that the main reason for three-dimensional simulation is due to short channel effects. Until recently three dimensional simulations were not considered practical due to the large memory requirement and accuracy needed during the solution of the large system of algebraic equations, but with advances in computer resources these requirements have been met. A three-dimensional finite element simulation of various semiconductor devices was discussed in [18], three-dimensional simulation of VLSI MOSFET's in [19] and a three-dimensional simulation of inverse narrow channel effect in [20].

This thesis is a three-dimensional numerical modeling of a short-channel MOSFET including the effect of trapped oxide charge. The oxide charge trapped in the boundary of the semiconductor and oxide has been reported in [33], and is believed to have a serious effect on the capacitance-voltage and current-voltage characterics of short channel MOSFET's. This work is an investigation of this effect in a numerical modeling enviornment.

CHAPTER 3 - THE BASIC SEMICONDUCTOR EQUATIONS

In order to simulate any semiconductor device the first step is to derive the partial differential equations which describe the transport of the carriers in the semiconductor while under the influence of external fields. The equations that will be derived are valid for all semiconductor devices although the only concern is with silicon MOSFET's. The equations can be used for any type of semiconducting material with the appropriate change of physical parameters. Also in this section the boundary conditions which will be used in the simulation are derived.

3.1 Equation Formation

To arrive at the equations which describe transport of carriers in the semiconductor device the Boltzmann Transport Equation (BTE) is used. The BTE is given by:

$$(n\cdot\mathbf{v}) + (q\cdot n\cdot\mathbf{E})/m^* + \nabla(n\cdot k\cdot T)/m^* = -(n\cdot\mathbf{v})/\tau$$

n is the carrier concentration, **v** is the drift velocity, **E** is the electric field, k is Boltzmann's contant, T is temperature, m^* is the effective mass of the carrier and τ is the average collision time. Solving this equation will give the carrier concentrations and drift velocities.

To use the BTE the motion of carriers has to be treated as semiclassical[2]. Semiclassical in this context means the carrier motion in a semiconductor with an externally applied field is regarded as a series of acceleration (treated by classical mechanics) and scattering (treated by quantum mechanics) events[3]. The assumption of a semiclassical nature accounts well for the transport of carriers in small silicon devices[2].

Fichtner et. al.[3] discussed the applicability of the BTE for the description of small semiconductor structures and concluded that silicon devices with active dimensions of 0.1 μm or larger can be described well with the semiclassical BTE approach.

The operation of a semiconductor device with dimensions greater than 0.1 μm can therefore be described by solving the BTE. However solving the BTE directly is impractical, therefore several simplifying assumptions must be made. The first is the quasi static local potential approximation[4]. This approximation changes the problem from space and momentum to space alone. Other approximations are; collisions are instantaneous, carrier-carrier interaction is negligible and the scattering probability is independent of external forces. Using these approximations the current density equations are derived.

The basic semiconductor equations needed for modeling short channel MOSFET's are Possion's equation, current density equations and the current continuity equations.

3.1.1 Poisson's Equation

Poisson's equation is:

$$\nabla \cdot \mathbf{D} = \rho \tag{3-1}$$

and it describes the electrostatic potential in a semiconductor.

The electric displacement vector **D** is related to the electric field vector **E** by:

$$\mathbf{D} = \epsilon_s \mathbf{E} \tag{3-2}$$

where ϵ_s is the semiconductor permittivity. If the permittivity is time independent this relation is valid for all materials. In a semiconductor with uniform composition ϵ_s is constant and the Poisson equation becomes:

$$\epsilon_s \nabla \cdot \mathbf{E} = - \epsilon_s \nabla^2 \psi = \rho \tag{3-3}$$

ψ is the electrostatic potential and is related to the electric field as given by:

$$\mathbf{E} = -\nabla\psi \tag{3-4}$$

and ρ the total space charge, assuming complete ionization at room temperature is given by;

$$\rho = -q(n - p + C) \tag{3-5}$$

where

$$C = N_D - N_A \tag{3-6}$$

N_D is the number of donor atoms, N_A is the number of acceptor atoms and p and n are the hole and electron carrier densities, respectively.

3.1.2 The Current Density Equations

The current density equations for electrons and holes, derived from the BTE with several assumptions, are given by:

$$\mathbf{J}_n = q\mu_n n\mathbf{E} + qD_n\nabla n \tag{3-7}$$

$$\mathbf{J}_p = q\mu_p p\mathbf{E} - qD_p\nabla p \tag{3-8}$$

where D_n and D_p are the electron and hole diffusivities respectively, and μ_n and μ_p are the electron and hole mobilities. The diffusion coefficients and mobilities are dependent on the electric field **E**, which is discussed in chapter 4.

Assuming a nondegenerate condition, the diffusion coefficients and mobilities are related by the Einstein relation[5]:

$$D_n = \mu_n kT/q \qquad (3\text{-}9)$$

$$D_p = \mu_p kT/q \qquad (3\text{-}10)$$

where k is Boltzmanns constant and T is the carrier temperature in Kelvin.

To describe carrier densities the Boltzmann approximation will be used. This approximation is valid for nondegenerate materials and is given by:

$$n = n_i \exp[q(\psi - \Phi_F)/kT] \qquad (3\text{-}11)$$

$$p = n_i \exp[q(\Phi_F - \psi)/kT] \qquad (3\text{-}12)$$

where n_i is the intrinsic carrier density and Φ_F is the Fermi potential under equilibrium conditions.

Under non-equilibrium conditions Fermi potential is replaced by Φ_{Fn} and Φ_{Fp} which are the quasi Fermi potentials for electrons and holes respectively. Equations (3-11) and (3-12) then become:

$$n = n_i \exp[q(\psi - \Phi_{Fn})/kT] \qquad (3\text{-}13)$$

$$p = n_i \exp[q(\Phi_{Fp} - \psi)/kT] \qquad (3\text{-}14)$$

The product of (3-13) and (3-14), n and p, is

$$np = n_i^2 \exp[(E_{Fn} - E_{Fp})/kT] \qquad (3\text{-}15)$$

This equation shows that the difference in the quasi Fermi level indicates how the pn product varies from its equilibrium value n_i^2.

3.1.3 The Current Continuity Equations

The current continuity equations are given by:

$$\nabla \cdot \mathbf{J}_n - q\frac{\partial n}{\partial t} = q \cdot R(\psi, n, p) \qquad (3\text{-}16)$$

$$\nabla \cdot \mathbf{J}_p + q\frac{\partial p}{\partial t} = -q \cdot R(\psi, n, p) \qquad (3\text{-}17)$$

$R(\psi,n,p)$ is the recombination and generation difference. This term takes into account different phenomena such as thermal generation, generation due to impact ionization and recombination due to traps.

Using equations (3-3), (3-7), (3-8), (3-16) and (3-17) the current flow and potential within a device can be described. This simulation will be steady state, although it could be adapted to a transient simulation, so that all derivatives with respect to time are set to zero. Substituting (3-7) and (3-8) into (3-16) and (3-17) and using (3-8) the following equations are realized

$$\nabla\cdot(D_n\nabla n - \mu_n n\nabla\psi) - R(\psi,n,p) = 0 \qquad (3\text{-}18)$$

$$\nabla\cdot(D_p\nabla p + \mu_p p\nabla\psi) - R(\psi,n,p) = 0 \qquad (3\text{-}19)$$

and the Poisson equation is

$$\nabla^2\psi - q/\epsilon_s(n - p + C) = 0 \qquad (3\text{-}20)$$

Equations (3-18), (3-19) and (3-20), are a system of coupled nonlinear partial differential equations in terms of n,p, and ψ, with the appropriate boundary conditions this system describes the behavior of short channel MOSFET's.

3.2 Boundary Conditions

The set of equations given by (3-18), (3-19) and (3-20) subject to the following boundary conditions are solved.

3.2.1 Source and Drain Contacts

The source and drain contacts are assumed to be ohmic contacts. In most simulation programs these contacts are assumed to be at thermal equilibrium with a vanishing space charge region. This implies:

$$n \cdot p - n_i^2 = 0 \tag{3-21}$$

$$n - p - C = 0 \tag{3-22}$$

These two equations can be put into the form of a Dirichlet boundary condition for the electrons and holes, as given by (3-23) and (3-24).

$$n = [\sqrt{(C^2 + 4 \cdot n_i^2)} + C]/2 \tag{3-23}$$

$$p = [\sqrt{(C^2 + 4 \cdot n_i^2)} - C]/2 \tag{3-24}$$

The boundary condition for ψ is given by

$$\psi = \psi_a + \psi_{bi} \tag{3-25}$$

where ψ_a is the applied voltage at the contact and ψ_{bi} is the built in voltage at the contact. ψ_a is 0 V for source contact and ψ_a is the drain voltage at the drain contact. The built in voltage is given by

$$\psi_{bi} = (kT/q) \cdot \ln(N_D/n_i) \tag{3-26}$$

3.2.2 Gate Contact and Iterface Trapped Charge

At this boundary there is an oxide-semiconductor interface. At this interface fixed charges exist. These are the interface charges and the fixed oxide charges.

The interface charges are due to Si-SiO_2 interface properties and are dependent on the chemical composition of this interface. The value of the interface trapped charge is on the order of $10^{10}cm^{-2}$[6].

The fixed oxide charge is located within approximately 30 Angstrom of the interface and is treated as if it were at the interface[33]. This charge is fixed and cannot be charged or discharged over a wide variation of surface potential. The value of the fixed oxide charge is on the order of $10^{10}cm^{-2}$[6].

At this oxide-semiconductor interface Gauss's law in differential form must be obeyed or:

$$\epsilon_{sem} \frac{\partial \psi}{\partial \mathbf{n}}\bigg|_{sem} - \epsilon_{ins} \frac{\partial \psi}{\partial \mathbf{n}}\bigg|_{ins} = Q_{int} \qquad (3\text{-}27)$$

ϵ_{sem} and ϵ_{ins} are the semiconductor and insulator permittivities respectively and **n** is a vector normal to the surface. Q_{int} is the trapped interface charge at the semiconductor - insulator interface which is assumed to be the sum of the interface charge and fixed charge($2 \cdot 10^{10}$ cm^{-2}). Gauss's law states that the electric flux(displacement vector) passing through any closed surface is equal to the total charge enclosed by that surface. From Gauss's law the boundary condition at the gate for electrostatic potential, ψ, can be determined. The boundary conditions for the electron and hole densities are given by[6]:

$$n = n_{po} \cdot \exp[q\,(\psi - \psi_D) / kT] \qquad (3\text{-}28)$$

$$p = p_{po} \cdot \exp[-q\,\psi / kT] \qquad (3\text{-}29)$$

where n_{po} and p_{po} are the equilibrium values of electron and hole concentrations respectively and ψ_D is the potential due to the drain bias.

3.2.3 Artificial Boundaries

Artificial boundaries are imaginery boundaries which are used to isolate the simulation domain, refer to figure 1. The artificial boundaries are chosen such that the device is selfcontained which is equivalent to assuming that the current density normal to the surface is zero. In other words, at these boundaries the following hold:

$$\frac{\partial \psi}{\partial \mathbf{n}} = 0 \tag{3-30a}$$

$$\frac{\partial n}{\partial \mathbf{n}} = 0 \tag{3-30b}$$

$$\frac{\partial p}{\partial \mathbf{n}} = 0 \tag{3-30c}$$

where **n** is the vector normal to the boundary. It is assumed that there is no variation of ψ, n, or p from the chosen boundary to the edge of the device.

CHAPTER 4 - THE PHYSICAL PARAMETERS

The system of nonlinear partial differential equations discussed in chapter 3 must be solved in order to simulate the behavior of the device. The purpose of this chapter is to elaborate on the physical parameters of these equations; the carrier mobility and the recombination-generation terms. Any simulation of a device relies heavily on the available models for these physical parameters and the accuracy of these models.

4.1 Carrier Mobility

Mobility is defined as the velocity of a carrier divided by the electric field applied to that carrier. There are many different types of scattering processes, lattice scattering, ioionized impurity scattering and saturation of drift velocity with high electric field, which contribute to the overall mobility. Mobilities are described in terms of a relaxation time. The relaxation time is a measure of the rate of return to the state of equilibrium from a disturbed state. The mobilities from each process add inversely to give the overall mobility.

One dominant scattering mechanism is interaction of carriers with the thermally generated phonons which are vibrations of the atoms in the crystal. These thermally generated vibrations are dependent on temperature. A model given by Sah et al is claimed to give accurate values of mobility in silicon in the temperature range of 4.2 to 600 degrees Kelvin[22], as given by equations (4-1) and (4-2).

$$\mu_n{}^L = 1/(1/(4195*(T/300)^{-1.5})+1/(2153*(T/300)^{-3.13})) \tag{4-1}$$

$$\mu_p{}^L = 1/(1/(2502*(T/300)^{-1.5})+1/(591*(T/300)^{-3.25})) \tag{4-2}$$

The symbols $\mu_n{}^L$ and $\mu_p{}^L$ donote mobilty due to lattice scattering for electrons and holes respectivley. It is stated in [22] that the additional effort for more elaborate formulae based on complicated theoretical models are not justified. The superscript L denotes lattice scattering.

The next scattering mechanism that will be used in the model of carrier mobility is the ionized impurity scattering. There has been a few models published for this type of scattering[23], [24]. These models are not independent from the lattice scattering, therefore a model that takes into account both of these scattering mechanisms must be used. One such model has been proposed by Scharfetter and Gummel[25]. This model predicts a combined mobility due to lattice vibrations and ionized impurities as given by:

$$\mu_{n,p}^{LI} = \mu_{n,p}^{L}/\sqrt{(1+N_D/(C_{n,p}^{ref} + N_D/S_{n,p}))}$$

$$C_n^{ref} = 3\cdot10^{16}\text{cm}^{-3}, \; S_n = 350 \qquad (4\text{-}3)$$

$$C_p^{ref} = 4\cdot10^{16}\text{cm}^{-3}, \; S_p = 81$$

This model gives an accurate account of the carrier mobility taking into account the lattice and impurity scattering mechanisms.

The final effect that will be considered is the saturation of the drift velocity for high electric fields. When this occurs the mobility will decrease with increasing electric field. The mobility taking into account lattice scattering, ionized impurity scattering and high electric fields has been approximated by [25] and is given as:

$$\mu^{LIE} = \mu^{LI}/\sqrt{(1+CI/(C^{ref}+CI/S)+(E/A)^2/(E/A+F)+(E/B)^2)} \qquad (4\text{-}4)$$

The varibles in (4-4) have to be imagined with the subscripts n or p. For the varibles A,B and F, [25] recommends the following values for silicon at 300 K:

$$A_n=3.5\cdot10^3\text{V/cm},\ B_n=7.4\cdot10^3\text{V/cm},\ F_n=8.8 \qquad (4\text{-}5)$$

$$A_p=6.1\cdot10^3\text{V/cm},\ B_p=2.5\cdot10^4\text{V/cm},\ F_p=1.6$$

There are two other basic types of scattering mechanisms that are not included in this model for mobility, carrier-carrier and neutral impurity scattering. These scattering effects are neglible compared to the three that have already been discussed and the inclusion of their effects would only increase the complexity of the model not the accuracy. In making this statement it should be remembered that one of the assumptions that was used to derive equations (3-18), (3-19) and (3-30) was that the carrier-carrier interaction is neglible.

4.2 Carrier Generation-Recombination

The importance of carrier generation-recombination phenomena is dependent on the particular device under consideration. For a bipolar device recombination is very important in determining the current gain whereas in a unipolar device it is of small importance. But if the device will be operating under a high field condition this term becomes very important. This is due to the generation due to impact ionization and the generation-recombination due to Auger phenomena.

The most dominant recombination mechanism in silicon is an indirect process involving a trap center somewhere in the energy bandgap. This mechanism is the Shockley-Read-Hall generation - recombination. There are four partial processes involved[26];

1. electron capture: an electron from the conduction band is trapped by an unoccupied defect which becomes occupied.
2. hole capture: an electron from an occupied trap moves to the valence band and neutralizes a hole. The trap becomes unoccupied.
3. hole emission: an electron from the valence band is trapped by a defect, thus leaving a hole in the valence band and an occupied trap.
4. electron emission: an electron from an occupied trap moves to the conduction band. The trap becomes unoccupied.

The Shockley-Read-Hall generation - recombination rate is given by [29], [30]:

$$R^{SRH} = (np - n_i^2)\cdot(\tau_p\cdot(n+n_1) + \tau_n\cdot(p+p_1))^{-1} \qquad (4\text{-}6)$$

with

$$\tau_p = 1/\sigma_p^{SRH} \qquad (4\text{-}7)$$

$$\tau_n = 1/\sigma_n^{SRH} \qquad (4\text{-}8)$$

τ_p and τ_n are the carrier lifetimes and σ_n^{SRH} and σ_p^{SRH} are defined as the capture rates[26].

The next type of recombination - generation that will be added in the simulation is Auger or three particle transistions. The Auger partial processes can be listed as follows[26];

1. electron capture: an electron from the conduction band moves to the valence band, transmitting the excess energy to another electron in the conduction band. In the valence band the electron recombines with a hole.
2. hole capture: an electron from the conduction band moves to the valence band transmitting the excess energy to a hole in the valence band, which moves away from the valence band edge. The electron recombines with a hole.
3. electron emission: an electron from the valence band moves to the conduction band by consuming the energy of a high energetic electron in the conduction band and leaving a hole

in the valence band.

4. hole emission: an electron from the valence band moves to the conduction band by consuming the energy of a high energetic hole in the valence band. A hole is left at the valence band edge.

In any of these transistions three particles are involved. The total net generation - recombination rate is given by[31]:

$$R^{AU} = (\sigma_n^{AU} \cdot n + \sigma_p^{AU} \cdot p) \cdot (n \cdot p - n_i^2) \qquad (4\text{-}9)$$

σ_n^{AU} and σ_p^{AU} are called the Auger coefficients and have the values:

$$\sigma_n^{AU} = 2.8 \cdot 10^{-31} \qquad (4\text{-}10a)$$

$$\sigma_p^{AU} = 9.9 \cdot 10^{-31} \qquad (4\text{-}10b)$$

The last type of generation that will be used in the model is impact ionization. Impact ionization is the most important phenomena in junction breakdown. There are two partial processes involved[26];

1. electron emission: an electron from the valence band moves to the conduction band by consuming the energy of a high energetic electron in the conduction band and leaving a hole

in the valence band.

2. hole emission: an electron from the valence band moves to the conduction band by consuming the energy of a high energetic hole in the valence band. A hole is left at the valence band edge.

Impact ionization generation can be expressed as:

$$G^{II} = 1/q\cdot(\alpha_n|\mathbf{J}_n| + \alpha_p|\mathbf{J}_p|) \qquad (4\text{-}11)$$

where α_n and α_p are the ionization coefficients and $\mathbf{J}_n$ and $\mathbf{J}_p$ are the current densities. The ionization coefficients are given by:

$$\alpha_n = \alpha_n^{inf}\cdot\exp(-E_n^{crit}/E) \qquad (4\text{-}12a)$$

$$\alpha_p = \alpha_p^{inf}\cdot\exp(-E_p^{crit}/E) \qquad (4\text{-}12b)$$

$$\alpha_n^{inf} = 1.0\cdot10^6\,\text{cm}^{-1},\ E_n^{crit} = 1.66\cdot10^6\,\text{V/cm} \qquad (4\text{-}12c)$$

$$\alpha_p^{inf} = 2.0\cdot10^6\,\text{cm}^{-1},\ E_p^{crit} = 1.98\cdot10^6\,\text{V/cm} \qquad (4\text{-}12d)$$

Equation (4-11) together with (4-12) is used for simulation of impact ionization.

CHAPTER 5 - THE DISCRETIZATION OF THE SEMICONDUCTOR EQUATIONS

Analytical solutions for the set of differential equations described in chapter 4, if not impossible, is extremely difficult to obtain. The first step in obtaining the solution of such system of equations is discretization of this system. To do so the differential equations are approximated by finite differences. This changes the system of equations from nonlinear coupled partial differential equations to a set of discretized, nonlinear coupled equations. The nonlinear discretized system is then linearized using Taylor's series for the functions describing the discretized equations. This forms the basis for Newton iteration scheme. The coupling of these equations is dealt with in the iteration cycles of the Newton iteration scheme which is described in chapter 6.

In the finite difference scheme the simulation domain will be rectangular with meshlines parallel to the three coordinate axises. At each intersection of these meshlines the three equations, (3-18), (3-19) and (3-20), is written in discrete form. Together all of these equations for each point will be used to solve iteratively for the values of ψ, n and p.

5.1 Finite Differences

In the rectangular domain the number of meshlines in each direction will be labeled NX, NY and NZ in the x direction, y direction and z direction respectively with a uniform mesh, refer to Figure 2. The discretization will be the classical seven point method which is a good compromise between complexity of discretization and the accuracy of the result. The distance x_i is assumed to be the distance from the origin to the i meshline parallel to the y axis. A similiar description can be made for y_j and z_m. The following abbreviations will be used:

$$h_i = x_{i+1} - x_i \tag{5-1}$$

$$k_j = y_{j+1} - y_j \tag{5-2}$$

$$l_m = z_{m+1} - z_m \tag{5-3}$$

$$u_{i,j,m} = u(x_i, y_j, z_m) \tag{5-4}$$

$$u_{i+1/2,j,m} = u((x_{i+1} + x_i)/2, y_j, z_m) \tag{5-5}$$

$$u_{i,j+1/2,m} = u(x_i, (y_{j+1} + y_j)/2, z_m) \tag{5-6}$$

$$u_{i,j,m+1/2} = u(x_i, y_j, (z_{m+1} + z_m)/2) \qquad (5\text{-}7)$$

This discretization scheme is the simplest with the local error being controlled by the mesh spacing.

5.1.1 Poissons Equation

Poissons equation is discretized first. The poisson equation can be written explicitly as:

$$\lambda^2(\psi^{xx} + \psi^{yy} + \psi^{zz}) - n + p + C = 0 \qquad (5\text{-}8)$$

where ψ^{xx}, ψ^{yy} and ψ^{zz} are second order partial derivatives in the x, y and z directions respectively. The first order partial derivatives can be replaced with[28]:

$$\psi^{x}\big|_{i,j,m} = (\psi_{i+1/2,j,m} - \psi_{i-1/2,j,m})\cdot(2/(h_i + h_{i-1})) + O(h)\cdot\psi^{xx}\big|_{i,j,m} \qquad (5\text{-}9)$$

$$\psi^{y}\big|_{i,j,m} = (\psi_{i,j+1/2,m} - \psi_{i,j-1/2,m})\cdot(2/(k_j + k_{j-1})) + O(k)\cdot\psi^{yy}\big|_{i,j,m} \qquad (5\text{-}10)$$

$$\psi^{z}\big|_{i,j,m} = (\psi_{i,j,m+1/2} - \psi_{i,j,m-1/2})\cdot(2/(l_m + l_{m-1})) + O(l)\cdot\psi^{zz}\big|_{i,j,m} \qquad (5\text{-}11)$$

The truncation error is controlled by the mesh spacing.

Substituting (5-9), (5-10) and (5-11) into the Poisson equation the following is realized:

$$\lambda^2 \cdot [\, (\psi^x_{i+1/2,j,m} - \psi^x_{i-1/2,j,m}) \cdot (2/(h_i + h_{i-1})) \quad +$$

$$(\psi^y_{i,j+1/2,m} - \psi^y_{i,j-1/2,m}) \cdot (2/(k_j + k_{j-1})) \quad +$$

$$(\psi^z_{i,j,m+1/2} - \psi^z_{i,j,m-1/2}) \cdot (2/(l_m + l_{m-1})) \quad -$$

$$n_{i,j,m} + p_{i,j,m} + C_{i,j,m} = 0 \qquad (5\text{-}12)$$

neglecting the higher order terms. Replacing the first order partial derivatives in (5-12) with:

$$\psi^x_{i+1/2,j,m} = (\, \psi_{i+1,j,m} - \psi_{i,j,m})/h_i + O(h^2) \cdot \psi^{xxx}\big|_{i,j,m}$$

$$(5\text{-}13)$$

$$\psi^y_{i,j+1/2,m} = (\, \psi_{i,j+1,m} - \psi_{i,j,m})/k_j + O(k^2) \cdot \psi^{yyy}\big|_{i,j,m}$$

$$(5\text{-}14)$$

$$\psi^z_{i,j,m+1/2} = (\, \psi_{i,j,m+1} - \psi_{i,j,m})/l_m + O(l^2) \cdot \psi^{zzz}\big|_{i,j,m}$$

$$(5\text{-}15)$$

the Poisson equation in difference form can then be written as:

$$\lambda^2 \cdot [\, ((\psi_{i+1,j,m} - \psi_{i,j,m})/h_i - (\, \psi_{i,j,m} - \psi_{i-1,j,m})/h_{i-1}) \cdot (2/(h_i + h_{i-1})) +$$

$$((\psi_{i,j+1,m} - \psi_{i,j,m})/k_j - (\psi_{i,j,m} - \psi_{i,j-1,m})/k_{j-1}) \cdot (2/(k_j + k_{j-1})) +$$

$$((\psi_{i,j,m+1} - \psi_{i,j,m})/l_m - (\psi_{i,j,m} - \psi_{i,j,m-1})/l_{m-1}) \cdot (2/(l_m + l_{m-1})) +$$

$$n_{i,j,m} + p_{i,j,m} + C_{i,j,m} = 0 \qquad (5\text{-}16)$$

Equation (5-16) is the discretized form of Poisson's equation that is used in the simulation.

5.1.2 Current Continuity Equations

The current continuity equations with the current density equations are written as:

$$(D_n n^x - \mu_n n \psi^x)^x + (D_n n^y - \mu_n n \psi^y)^y + (D_n n^z - \mu_n n \psi^z)^z -$$

$$R(\psi, n, p) = 0 \qquad (5\text{-}17)$$

$$(D_p p^x - \mu_p p \psi^x)^x + (D_p p^y - \mu_p p \psi^y)^y + (D_p p^z - \mu_p p \psi^z)^z -$$

$$R(\psi, n, p) = 0 \qquad (5\text{-}18)$$

Using the approximation for first order partial derivatives given by (5-9), (5-10) and (5-11) the three dimensional current continuity equation for electrons can be written as (5-19).

$$[(-J_{nx})\big|_{i+1/2,j,m} - (-J_{nx})\big|_{i-1/2,j,m}] \cdot (2/(h_i + h_{i-1}))$$

$$+ O(h) \cdot J_{nx}{}^{xx} +$$

$$[(-J_{ny})\big|_{i,j+1/2,m} - (-J_{ny})\big|_{i,j-1/2,m}]\cdot(2/(k_j + k_{j-1}))$$

$$+\ O(k)\cdot J_{ny}^{yy}\ +$$

$$[(-J_{nz})\big|_{i,j,m+1/2} - (-J_{nz})\big|_{,j,m-1/2}]\cdot(2/(l_m + l_{m-1}))$$

$$+\ O(l)\cdot J_{nz}^{zz}\ -\ R(\psi, n, p)\ = 0 \qquad (5\text{-}19)$$

J_{nx}, J_{ny} and J_{nz} are the scaled electron current densities in the x, y and z directions respectivley. The current densities in each direction may be written in terms of first order derivatives as[28]:

$$J_{nx} = \mu_n\cdot n\cdot\psi^x - D_n\cdot n^x \qquad (5\text{-}20)$$

$$J_{ny} = \mu_n\cdot n\cdot\psi^y - D_n\cdot n^y \qquad (5\text{-}21)$$

$$J_{nz} = \mu_n\cdot n\cdot\psi^z - D_n\cdot n^z \qquad (5\text{-}22)$$

To discretize the continuity equation for electrons a Taylors series is used on the current density. This yields the following:

$$J_{nx}(x\in[x_i, x_{i+1}], y_j, z_m) = J_{nx}\big|_{i+1/2,j,m} + (x - x_i - h_i/2)\cdot$$

$$J_{nx}^{x}\big|_{i+1/2,j,m} +\ O(h^2)\cdot\ J_{nx}^{xx}\big|_{i+1/2,j,m} \qquad (5\text{-}23)$$

$$J_{ny}(\ x_i, y\in[y_j, y_{j+1}], z_m) = J_{ny}\big|_{i,j+1/2,m} + (y - y_j - k_j/2)\cdot$$

$$J_{ny}^{y}\big|_{i,j+1/2,m} +\ O(k^2)\cdot\ J_{ny}^{yy}\big|_{i,j+1/2,m} \qquad (5\text{-}24)$$

$$J_{nz}(x_i, y_j, z\in[z_m, z_{m+1}]) = J_{nz}\big|_{i,j,m+1/2} + (z - z_m - l_m/2)\cdot$$

$$J_{nz}{}^{z}\big|_{i,j,m+1/2} + O(l^2)\cdot J_{nz}{}^{zz}\big|_{i,j,m+1/2} \qquad (5\text{-}25)$$

The higher order terms $O(h^2)$, $O(k^2)$ and $O(l^2)$ are ignored and the following equality is made on the interval $[x_i, x_{i+1}]$:

$$\mu_n\cdot n\cdot\psi^x - D_n\cdot n^x = J_{nx}\big|_{i+1/2,j,m} + (x - x_i - h_i/2)\cdot J_{nx}{}^{x}\big|_{i+1/2,j,m}$$

$$(5\text{-}26)$$

$$n(x_i, y_j, z_m) = n_{i,j,m} \qquad (5\text{-}27a)$$

$$n(x_{i+1}, y_j, z_m) = n_{i+1,j,m} \qquad (5\text{-}27b)$$

The assumptions used in equation (5-26) are that the partial derivative of the electrostatic potential between two mesh points is constant and the scaled Einstein relation is assumed to hold for the scaled carrier diffusitivities and mobilities.

Equation (5-26) is a first order differential equation with two boundary conditions(5-27). The solution of (5-26) with boundary conditions given by (5-27) is:

$$J_{nx}\big|_{i+1/2,j,m} = (D_n\big|_{i+1/2,j,m}\cdot B(\psi_{i,j,m} - \psi_{i+1,j,m})\cdot n_{i,j,m} -$$

$$B(\psi_{i+1,j,m} - \psi_{i,j,m})\cdot n_{i+1,j,m})/h_i + h_i(0.5\cdot\coth((\psi_{i,j,m}-\psi_{i+1,j,m})/2)$$

$$- 1/(\psi_{i,j,m} - \psi_{i+1,j,m}))\cdot J_{nx}{}^{x}\big|_{i+1/2,j,m} \qquad (5\text{-}28)$$

B(x) is the Bernoulli function[28] which is defined as:

$$B(x) = x/(\exp(x) - 1) \qquad (5\text{-}29)$$

Under the assumtion that $\psi_{i,j,m}$ only differs by O(h) from its nearest neighbor the last term in (5-28) is $O(h^2)$. Using equations (5-28) and (5-19) and ignoring the last term in (5-28) the discrete form of the continuity equation for electrons can be written in the x direction. The discretized continuity equation equation in the y and z directions are completely analogous. The descretized continuity equation in three dimensions is then given by:

$$n_{i+1,j,m}\cdot D_n\big|_{i+1/2,j,m}\cdot B(\psi_{i+1,j,m}-\psi_{i,j,m})\cdot(k_{j-1} + k_j)(l_{m-1}+l_m)/h_i +$$

$$n_{i,j+1,m}\cdot D_n\big|_{i,j+1/2,m}\cdot B(\psi_{i,j+1,m}-\psi_{i,j,m})\cdot(h_{i-1} + h_i)(l_{m-1}+l_m)/k_j +$$

$$n_{i,j,m+1}\cdot D_n\big|_{i,j,m+1/2}\cdot B(\psi_{i,j,m+1}-\psi_{i,j,m})\cdot(h_{i-1} + h_i)(k_{j-1}+k_j)/l_m +$$

$$n_{i-1,j,m}\cdot D_n\big|_{i-1/2,j,m}\cdot B(\psi_{i-1,j,m}-\psi_{i,j,m})\cdot(k_{j-1} + k_j)(l_{m-1}+l_m)/h_{i-1} +$$

$$n_{i,j-1,m}\cdot D_n\big|_{i,j-1/2,m}\cdot B(\psi_{i,j-1,m}-\psi_{i,j,m})\cdot(h_{i-1} + h_i)(l_{m-1}+l_m)/k_{j-1} +$$

$$n_{i,j,m-1}\cdot D_n\big|_{i,j,m-1/2}\cdot B(\psi_{i,j,m-1}-\psi_{i,j,m})\cdot(h_{i-1} + h_i)(k_{j-1}+k_j)/l_{m-1} -$$

$$n_{i,j,m}\cdot(D_n\big|_{i+1/2,j,m}\cdot B(\psi_{i,j,m}-\psi_{i+1,j,m})\cdot(k_{j-1} + k_j)(l_{m-1}+l_m)/h_i +$$

$$D_n|_{i,j+1/2,m} \cdot B(\psi_{i,j,m} - \psi_{i,j+1,m}) \cdot (h_{i-1} + h_i)(l_{m-1} + l_m)/k_j +$$

$$D_n|_{i,j,m+1/2} \cdot B(\psi_{i,j,m} - \psi_{i,j,m+1}) \cdot (h_{i-1} + h_i)(k_{j-1} + k_j)/l_m +$$

$$D_n|_{i-1/2,j,m} \cdot B(\psi_{i,j,m} - \psi_{i-1,j,m}) \cdot (k_{j-1} + k_j)(l_{m-1} + l_m)/h_{i-1} +$$

$$D_n|_{i,j-1/2,m} \cdot B(\psi_{i,j,m} - \psi_{i,j-1,m}) \cdot (h_{i-1} + h_i)(l_{m-1} + l_m)/k_{j-1} +$$

$$D_n|_{i,j,m-1/2} \cdot B(\psi_{i,j,m} - \psi_{i,j,m-1}) \cdot (h_{i-1} + h_i)(k_{j-1} + k_j)/l_{m-1}) -$$

$$R|_{i,j,m}(\psi, n, p) \cdot (h_{i-1} + h_i)(k_{j-1} + k_j)(l_{m-1} + l_m)/8$$

$$= 0 \qquad (5\text{-}30)$$

Similarly the discretized continuity equation for holes can be written as:

$$p_{i+1,j,m} \cdot D_p|_{i+1/2,j,m} \cdot B(\psi_{i,j,m} - \psi_{i+1,j,m}) \cdot (k_{j-1} + k_j)(l_{m-1} + l_m)/h_i +$$

$$p_{i,j+1,m} \cdot D_p|_{i,j+1/2,m} \cdot B(\psi_{i,j,m} - \psi_{i,j+1,m}) \cdot (h_{i-1} + h_i)(l_{m-1} + l_m)/k_j +$$

$$p_{i,j,m+1} \cdot D_p|_{i,j,m+1/2} \cdot B(\psi_{i,j,m} - \psi_{i,j,m+1}) \cdot (h_{i-1} + h_i)(k_{j-1} + k_j)/l_m +$$

$$p_{i-1,j,m} \cdot D_p|_{i-1/2,j,m} \cdot B(\psi_{i,j,m} - \psi_{i-1,j,m}) \cdot (k_{j-1} + k_j)(l_{m-1} + l_m)/h_{i-1} +$$

$$p_{i,j-1,m} \cdot D_p|_{i,j-1/2,m} \cdot B(\psi_{i,j,m} - \psi_{i,j-1,m}) \cdot (h_{i-1} + h_i)(l_{m-1} + l_m)/k_{j-1} +$$

$$p_{i,j,m-1} \cdot D_p|_{i,j,m-1/2} \cdot B(\psi_{i,j,m} - \psi_{i,j,m-1}) \cdot (h_{i-1} + h_i)(k_{j-1} + k_j)/l_{m-1} -$$

$$p_{i,j,m} \cdot (D_p|_{i+1/2,j,m} \cdot B(\psi_{i+1,j,m} - \psi_{i,j,m}) \cdot (k_{j-1} + k_j)(l_{m-1} + l_m)/h_i +$$

$$D_p\big|_{i,j+1/2,m}\cdot B(\psi_{i,j+1,m}-\psi_{i,j,m})\cdot(h_{i-1}+h_i)(l_{m-1}+l_m)/k_j +$$

$$D_p\big|_{i,j,m+1/2}\cdot B(\psi_{i,j,m+1}-\psi_{i,j,m})\cdot(h_{i-1}+h_i)(k_{j-1}+k_j)/l_m +$$

$$D_p\big|_{i-1/2,j,m}\cdot B(\psi_{i-1,j,m}-\psi_{i,j,m})\cdot(k_{j-1}+k_j)(l_{m-1}+l_m)/h_{i-1} +$$

$$D_p\big|_{i,j-1/2,m}\cdot B(\psi_{i,j-1,m}-\psi_{i,j,m})\cdot(h_{i-1}+h_i)(l_{m-1}+l_m)/k_{j-1} +$$

$$D_p\big|_{i,j,m-1/2}\cdot B(\psi_{i,j,m-1}-\psi_{i,j,m})\cdot(h_{i-1}+h_i)(k_{j-1}+k_j)/l_{m-1}) -$$

$$R\big|_{i,j,m}(\psi,n,p)\cdot(h_{i-1}+h_i)(k_{j-1}+k_j)(l_{m-1}+l_m)/8$$

$$= 0 \qquad (5\text{-}31)$$

Eauations (5-31) and (5-32) are the discretized form of the continuity equations with the current density equations substituted in.

5.1.3 Boundary Conditions

The discretization of the boundary conditions at the source and drain is simple because values at one point are involved only. From equations (3-18), (3-19) and (3-20):

$$n_{i,NY,m} = [\sqrt{(C_{i,NY,m}{}^2 + 4n_i^2)} + C_{i,NY,m}]/2 \qquad (5\text{-}32)$$

$$p_{i,NY,m} = [\sqrt{(C_{i,NY,m}{}^2 + 4n_i^2)} - C_{i,NY,m}]/2 \qquad (5\text{-}33)$$

$$\psi_{i,NY,m} = \psi_{bi}\big|_{i,NY,m} + \psi_a\big|_{i,NY,m} \qquad (5\text{-}34)$$

The mesh line NY is the point where drain, source and gate contacts are located.

At the gate:

$$\epsilon_{sem} \cdot (\psi_{i,NY,m} - \psi_{i,NY-1,m})/(y_{NY} - y_{NY-1}) - \epsilon_{ins} \cdot (Vg - \psi_{i,NY,m})/t_{ins} = Q_{int} \qquad (5\text{-}35)$$

where Vg is the applied gate voltage and t_{ins} is the oxide thickness, and $\psi_{i,NY,m}$ is the potential under the oxide at the boundary of the simulation.

The artificial boundaries are at i=1, i=NX, j=1, m=1, m=NZ. At these boudaries ψ, n and p are equal to there closest neighboring mesh point. This is the same as saying the derivative normal to the boundary is zero.

5.1.4 Scaling

In order to get faster convergence scaling the dependent varibles ψ, n and p is necessary. These dependent varibles are of greatly differing orders of magnitude. DeMari[27] gave a standard way of scaling. This method of scaling was discussed and compared with a "better" method in [28]. The distance is scaled with the Debye length x_o, potential is scaled with kT/q (ψ_o), the carrier concentrations are scaled with the maximum doping concentration C_o, the diffusion coefficients are scaled with the maximum diffusion coefficient in the simulation domain D_o, the mobility is scaled with the D_o/ψ_o, and the recombination - generation term is scaled with $D_o \cdot C_o / x_o^2$. So now equations (3-23), (3-24), (3-25), the basic semiconductor equations scaled, look like:

$$\lambda^2 \cdot \nabla^2 \psi - (n - p - C) = 0 \qquad (5\text{-}36)$$

$$\nabla \cdot (D_n \nabla n - \mu_n n \nabla \psi) - R(\psi, n, p) = 0 \qquad (5\text{-}37)$$

$$\nabla \cdot (D_p \nabla p - \mu_p p \nabla \psi) - R(\psi, n, p) = 0 \qquad (5\text{-}38)$$

where

$$\lambda^2 = \psi_o \epsilon_s / (x_o^2 \cdot q \cdot C_o) \qquad (5\text{-}39)$$

The biggest effect this scaling has is on the carrier concentrations and doping concentrations. The poisson equation is multipied with a factor approximately 10^{-14} depending on the device and operating conditions while the continuity equations with the current densities substituted in are multiplied with approximately 10^{-10}.

5.2 Designing a Mesh

Figure 1 shows the structure of a short channel MOSFET with channel length L, width Z and source and drain junction depth of rj. The source and substrate bias are connected to ground(0V).

Figure 2 shows a representation of the mesh used in this simulation. When designing a mesh for a simulation, the number of mesh points should be greater in the regions of the device where varibles are changing in a short distance, such as, in the vicinity of drain, source and surface region of the channel. The main requirement of the mesh spacing is that it gives the desired accuracy for such regions. After using a nonuniform mesh in this simulation it was changed to a uniform mesh for reasons of simplicity.

CHAPTER 6 - SOLUTION OF THE DISCRETE SEMICONDUCTOR EQUATIONS

The Poisson and continuity equations for electrons and holes have been discretized. The next step in the simulation is to solve for the potential, electron and hole concentrations at each point. The method of solution is an iterative technique called Newton's method. Section 6.1 of this section discusses the Newton iteration applied to the semiconductor equations. Section 6.2 discusses the numerical instabilities. Section 6.3 discusses the initial guess needed for quicker convergence of the iteration.

6.1 Newton's Method

To explain Newtons method applied to the semiconductor equations the notation will be defined first.

$$F_1(\psi, n, p) = 0 \tag{6-1}$$

$$F_2(\psi, n, p) = 0 \tag{6-2}$$

$$F_3(\psi, n, p) = 0 \tag{6-3}$$

F_1 denotes the discrete Poisson equation, F_2 and F_3 are the discretized continuity equations for electrons and holes, respectivley.

Using Taylors theorem and neglecting the higher order terms the following may be written[39]:

$$(F_1^{\psi}\delta\psi + F_1^{n}\delta n + F_1^{p}\delta p)^{r+1} = -F_1(\psi^r, n^r, p^r)$$

(6-4)

$$(F_2^{\psi}\delta\psi + F_2^{n}\delta n + F_2^{p}\delta p)^{r+1} = -F_2(\psi^r, n^r, p^r)$$

(6-5)

$$(F_3^{\psi}\delta\psi + F_3^{n}\delta n + F_3^{p}\delta p)^{r+1} = -F_3(\psi^r, n^r, p^r)$$

(6-6)

The correction vector for the r-th iterative is given by:

$$\delta\psi^r = \psi^{r+1} - \psi^r \quad (6\text{-}7)$$

$$\delta n^r = n^{r+1} - n^r \quad (6\text{-}8)$$

$$\delta p^r = p^{r+1} - p^r \quad (6\text{-}9)$$

The unknowns $\delta\psi$, δn and δp at each point are found and added to ψ, n and p to give the updated solution or in other words, if ψ^r is a solution then ψ^{r+1} is a better solution if $\psi^{r+1} = \psi^r + \delta\psi^r$. Rearranging terms in equations (6-4) thru (6-6) yeilds:

$$F_1^{\psi}\delta\psi^{r+1} = -F_1(\psi^r, n^r, p^r) - F_1^{n}\delta n^r - F_1^{p}\delta p^r$$

(6-10)

$$F_2^n \delta n^{r+1} = -F_2(\psi^r, n^r, p^r) - F_2^\psi \delta \psi^r - F_2^p \delta p^r \tag{6-11}$$

$$F_3^p \delta p^{r+1} = -F_3(\psi^r, n^r, p^r) - F_3^\psi \delta \psi^r - F_3^n \delta n^r \tag{6-12}$$

or to resubstitute the Taylors series expansion and combine like terms on the right hand side of (6-10), (6-11) and (6-12) yields[39]:

$$F_1^\psi \delta \psi^{r+1} = -\omega F_1(\psi^r, n^r + \delta n^r, p^r + \delta p^r) \tag{6-13}$$

$$F_2^n \delta n^{r+1} = -\omega F_2(\psi^r + \delta \psi^{r+1}, n^r + \delta n^r, p^r + \delta p^r) \tag{6-14}$$

$$F_3^p \delta p^{r+1} = -\omega F_3(\psi^r + \delta \psi^{r+1}, n^r + \delta n^{r+1}, p^r + \delta p^r) \tag{6-15}$$

The factor ω is the under relaxtion parameter which is between zero and one. The term ω should be different for each equation[39]. The parameter slows the convergence of the overall solution so that the equations will not diverge due to a poor initial guess. Since the number of equations that is solved is so large rounding errors become significant and can effect the updating of the solution. For this reason the under relaxation is necessary. In other words, using under relaxation the updated solution does not change radically at the price of slower convergence. This system of equations is solved iteratively until the desired accuracy is attained.

6.2 Numerical Instability and Convergence Problems

The Newton method in conjunction with under relaxation discussed in the last section worked well for low bias conditions, but not for higher bias conditions. If the biasing conditions are such that the device is punched through convergence will be even more difficult to obtain. A simple method for determining if the device is punched through is given by Sze[6]. He uses a abrupt junction approximation to determine the width of the drain depletion layer. This width is given by (6-16).

$$W_D=\sqrt{((2\epsilon_s/qN_A)\cdot(V_D + V_{bi}))} \tag{6-16}$$

With a shorter channel length a smaller drain voltage can cause punch through.

Another reason for the poor convergence of Newton's method was due to the greatly differing values of the dependent varibles. To overcome these problems a linearization scheme of the following was used[7],[28]:

$$\nabla^2 \psi^{r+1} - \omega\cdot(n+p)\cdot(\psi^{r+1} - \psi^r) - (n - p + C) = 0$$

(6-17)

Similarily the current continuity equations become:

$$\nabla\cdot\mathbf{J}_n{}^{r+1} - \omega\cdot R\cdot(n^{r+1} - n^r) - R = 0 \qquad (6\text{-}18)$$

$$\nabla\cdot\mathbf{J}_p{}^{r+1} - \omega\cdot R\cdot(p^{r+1} - p^r) - R = 0 \qquad (6\text{-}19)$$

When the solution converges, $x^{k+1} = x^k$, the added term becomes zero. This type of scheme was found to work very well for the discretized semiconductor equations. The method used to solve the equations after this linearization is a strongly implicit procedure. This method for the solution of very large sparse matrices was developed by Stone[40]. The Numerical Algorithm Group, NAG, routine D03ECF is used. This NAG routine uses Stones method. The method of linearization disscussed in conjunction with the NAG routine worked well at high and low biasing voltages.

6.3 Initial Guess

For quicker convergence and usually a requirement for convergence at all, an initial guess must be supplied in conjunction with Newton's method. The particular model which is used for the initial guess of this simulation is given by Yau[32]. The drain depetion layer width is determined by abrupt junction approximation given by:

$$W_D = \sqrt{((2\epsilon_s/qN_A)\cdot(V_D + V_{bi}))} \qquad (6\text{-}20)$$

and the source depletion layer width is given by:

$$W_S = \sqrt{((2\epsilon_s/qN_A)\cdot(V_{bi}))} \qquad (6\text{-}16)$$

The n doped regions under the drain and source are assumed to be equipotential areas so that the electron and hole concentrations are constant. For finding the initial guess in the drain region equation (6-17) is used.

$$\psi = \psi_o \exp((x_1^2 + y_1^2)/W_D) \qquad (6\text{-}17)$$

ψ_o is the potential at the drain substrate interface, x_1 and y_1 are the x and y distances from the drain substrate interface. For n and p use equation (6-17) with ψ_o replaced with the electron or hole density at the interface. Simiarly in the source region use equation (6-17) with W_S replacing W_D. This initial guess is used to generate values for the mobility and recombination.

CHAPTER 7 - RESULTS OF SIMULATIONS

In this chapter the results of three simulations are presented. The silicon MOSFET's used in the simulation have been constructed from typical devices being fabricated in industry at this time[6]. In section 7.1 a MOSFET with no external biasing conditions will be modeled. This is followed by two simulations with external biasing conditions. At the end of this chapter a comparison is made between a MOSFET with and without oxide charge to demonstrate how this charge effects the drain current. Also at the end of this chapter are plots of input capacitance vs. drain voltage, transconductance vs. drain voltage and drain current vs. drain voltage, which are used to investigate the dependency of cutoff frequency of the device on the channel length and doping concentration of the channel.

7.1 Simulation of Unbiased MOSFET

The MOSFET that is simulated has the source, gate and drain contacts connected to ground, 0V. The channel length of device 1 is one micrometer, substrate doping of $10^{15}cm^{-3}$ and drain and source doping of $10^{17}cm^{-3}$. Table 1 lists the parameters for device 1. The operating temperature is assumed to be 300 degrees Kelvin.

Figure 3 shows the log of the doping profile for the device. Constant surface diffusion process has been simulated within the simulation program. The inclusion of surface diffusion process in the simulation helps give more accurate results.

Figure 4 shows the electrostatic potential distribution for the device. The potential distribution consists only of the built in potential within the device due to the p-n junctions at the source and drain contacts.

Figure 5 shows the concentration of electrons in the device with no applied potential. The z axis in the Figure is the logrithm of the electron concentration. The distribution of electrons is only affected by the built in potential of the device. Figure 6 shows the hole concentration which is also affected by the built in potential.

Figure 7 and 8 show the electron and hole mobility respectively. As was mentioned earlier these mobilities are field dependent which takes into account the effects of scattering mechanisms such as lattice, impurity scattering and velocity saturation due to high electric field. The mobilities become less in the region of the source and drain due to the higher doping concentration and electric fields in that region. The mobilities in the bulk of the device are approximately 1375 $cm^2V^{-1}s^{-1}$ and 465 $cm^2V^{-1}s^{-1}$ for electrons and holes respectively.

The net recombination/generation term shown in Figure 9 has been transformed using:

$$z = \mathrm{Log}_{10}(1 + rg \cdot 10^{-18}) \qquad (7\text{-}1)$$

where rg is the net recombination/generation term. As would be expected the only recombination is in the area of the source and drain regions. This small amount of recombination is negligible and had very little effect on the simulation results. Figures 22 and 23 show a comparison of I-V characteristics with and without recombination. At this point it should be pointed out that the main process of current transport in a MOSFET is drift. The diffusion process is usually negligible and some simultation programs for MOSFET's don't even take it under consideration[38].

The final plot, Figure 10, in this simulation is of the magnitude of the current density. In determining the drain current the current density parallel to and directly under the gate region is used[3].

7.2 Simulation of a Biased MOSFET

In the preceeding section an unbiased MOSFET was presented. The plots given are of interest to the device engineer but of no importance to the applications engineer. The applications engineer is interested in how the device operates under certain applied voltages. The MOSFET simulated in this section has the same device parameters as the previous section with the exception of the substrate doping which is $7 \cdot 10^{15}$, refer to Table 1. The applied voltage at the drain is 0.5 volts and at the gate is 0.2 volts. The doping profile is shown in Figure 11 with the same description as was previously given.

The electrostatic potential for device 2 is shown in Figure 12. The saddle point between the drain and gate has been shifted towards the source. This shift is due to the increase in the drain potential. The force on carriers in the drain region is greater than in the source region. This inequality causes the electrons to migrate towards the drain.

The electron concentration for device 2 is shown in Figure 13. The z axis is logarithmic. The channel region has an accumulated layer of charge which is approximately one million times smaller then the concentration of electrons in the drain and source areas.

This layer of electrons is the path by which the drain current flows. From this plot it can be concluded that the gate surface is only mildly inverted. Figure 14 shows the corresponding concentration of holes. Figures 15 and 16 show the electron and hole mobilities under biased conditions respectively. The carrier mobilities under biased conditions, Figures15 and 16, are lower than that of the unbiased condition, Figures 6 and 7. This is due to the higher potential distribution for device 2. This higher potential increases the scattering of carriers and thus lowers the mobility.

The recombination/generation term for device 2 shown in figure 17 has the same transformation to the z axis as was given earlier. The recombination in this device has one significant point which lies in the region of the drain channel region. This plot illustrates that the recombination takes place mainly in the drain region. This recombination is due to the large potential gradient that exists in this area.

The magnitude of the current density is given in Figure 18. The current density in the direction from the source to drain determines the amount of drain current[3].

7.3 Effects of Gate Oxide Charge

Figure 24 shows drain current versus gate voltage for device 2 with differing values of oxide charge. The drain voltage is set at 0.5 volts.

It can be seen from this Figure that the value of the oxide charge does have an effect on the drain current of the device. With increasing positive charge the drain current increases. It should be noted that this is an n-channel device operating in the enhancement mode. With no applied gate potential(V_g=0V) the leakage current that flows is increased with the increase in positive charge at this surface. The magnitude of current shift is on the order of microamps. This small amount of shift would be insignificant in most cases of design, but when designing low biased, low noise MOSFET's it could become very significant.

7.4 MOSFET Characterization

To characterize a MOSFET plots of drain current vs. drain voltage with varying gate voltage, transconductance vs. drain voltage and input capacitance vs. drain voltage must be given for the device. This has been done for a device, device 3, with parameters given in

Table 1. The channel length for device 3 is 0.73 μm and the drain and substrate dopings are $10^{20}cm^{-3}$ and $7 \cdot 10^{16}cm^{-3}$, respectively. Figure 19 shows the doping profile for device 3. Figure 20 shows electrostatic potential for the device with drain bias of 2 volts and gate bias of 1 volt, with the corresponding electron concentraion shown in figure 21.

Figures 22 and 23 show the I-V characteristics for device 3. The pinch off voltage of the device is approximately 4.0 volts. There is transistor action beyond the pinch off voltage although it is not linear. Figures 25 and 26 are the transconductance and input capacitance vs. drain voltage.

CHAPTER 8 - DISCUSSIONS AND CONCLUSIONS

The previous section gave the results of three simulations. The first two simulations were carried out using the Newton iteration discussed in Chapter 6. The number of mesh lines used were 35, 30 and 15 in the x, y and z directions, respectively. This number of mesh lines translates into a matrix size of approximately 15,000 by 15,000 or 15,000 unknowns. This large of a matrix must be solved very carefully due to the fact that rounding errors become significant. The Newton iteration worked at very low bias but not at a higher bias. For this reason the linearization scheme discussed in chapter 6 was used. The instability that was encountered using Newton's method was not seen using the linearization technique in conjunction with a strongly implicit procedure for solving the large matrix.

8.1 The Instability

The Newton underelaxation scheme employed here showed a relatively poor performance in converging to an accurate solution under high bias conditions. This was due to the fact that the Jacobian matrix was not diagonally dominant. A sample unstable convergence of electrostatic potential is shown in Figure 27. After modifying the

Jacobian matrix entries, as discussed in section 6.2, the solution converged more readily.

8.2 Distribution of Potential, Carrier Concentrations and Current Densities

The electrostatic potentials shown in Figures 4, 12 and 20 clearly illustrate the effect of biasing on the potential distribution. The shift in the saddle point is due to the doping concentrations in the substrate and the applied bias. The higher the substrate doping, still assuming an abrupt drain-substrate junction, the less the potential distribution will protrude into the substrate region for a given applied voltage. For device 2 with $L = 1.0$ µm, $N_D = 10^{17}\text{cm}^{-3}$ and $N_A = 7 \cdot 10^{15}\text{cm}^{-3}$, the applied drain bias of 0.5 volts is almost enough to cause the MOSFET channel to punch through. The doping concentration changes the value of drain voltage needed to achieve punch through as well. For device 1 with a substrate doping of 10^{15} an applied drain potential of 0.5 volts would cause punch through. The same device with the substrate doping of $7 \cdot 10^{15}$, device 2, does not cause punch through.

The distribution of electrons in the channel under different biasing conditions are presented in Figures 5, 13 and 21. Figure 5 corresponds to no inversion, device 1, Figure 13 to weak inversion, device 2 and Figure 21 to strong inversion device 3. For a device to have its channel region strongly inverted the applied gate potential must be high enough to draw the carriers to the oxide interface. For exact poptential required to bring on strong inversion the device geometries must be known. For device 3, stong inversion occurs at a gate potential of approximately 1.0 volt while weak inversion occurs at 0.5 volts.

The net recombination/generation for devices 1 and 2 are shown in Figures 9 and 17, respectively. Although the magnitude of this term changed significantly, the overall effect on the I_D-V_D characteristics is not significant, as shown in Figure 22 which should be compared to Figure 23. The greatest change in these I-V characteristics is approximately 10 μA. This result agrees with those reported by others[3],[21] and [28].

The magnitude of the current density is shown in Figures 10 and 18. When determining the drain current only the component of the current density in the x direction is used. For the discrete case the current density in the x direction passes through a plane defined by a constant x and varying y and z. The area of this plane is multiplied by the current density flowing through this plane.

With this program different parameters can be changed in order to find the optimum design. For example, if a high transconductance is sought while the only varible in the device that is flexible is channel length, the program can simulate the device parameters with different channel lengths until the highest transconductance is found. This allows creativity when designing the device.

8.3 I_D-V_D Characteristics

Figure 22 shows the I-V characteristics for device 3. The pinch off voltage is about 4.0 volts for this device. There is transistor action beyond the pinch off voltage although it is not very linear.

The short channel effects of device 3 shown in Figure 22 can be illustrated by considering the curve for a gate voltage of 3.0 volts. Beyond the drain voltage of 4.0 volts the device has punched through. With increasing drain voltage an increase in the drain current occurs. This effect, which was mentioned earlier, is failure of current saturation due to punch through. The drain current of a long channel MOSFET in the saturation region changes very little with drain voltage. For this short channel device a drain voltage of 4.5 volts corresponds to a drain current of approximately 10 mA while at a drain voltage of 5.5 volts the drain current is 13.5 mA. The difference in drain voltage of 1.0 volt gives a difference in drain current of 3.5 mA. This significant change in drain current while the device is in the saturation region is a very good example of the short channel effects.

If the gate voltage is increased beyond 3.0 volts the drain current starts to increase in an almost ohmic fashion. This is due to the fact that the channel has punched through. Increasing the channel length will give better characteristics in the saturation region. If the channel length is increased the applied voltage needed to create the punch through effect will increase.

To accurately model this short channel behavior a field dependent mobility and three dimensional simulation must be used. As the electric field is increased the carrier velocity in the channel saturates. This effect must be modeled with a field dependent mobility. As the source and drain depletion layer widths become comparable to the channel length the transverse field, controlled by the gate bias, becomes comparable to the longitudinal field controlled by the drain bias. To take into account this two dimensional effect all three dimensions must be included in the simulation.

8.4 Effects of Oxide Charge

The oxide charge affects how the gate potential changes the channel region in the MOSFET. A positive charge at the oxide semiconductor interface causes a greater current to flow for a lesser gate bias. The positive charge at the interface attracts a negative layer of charge under the interface. This layer of negative charge thus increases the amount of leakage current when the device is off.

Figure 24 shows how the gate oxide changes the drain current for device 2. For plotting purposes the drain current presented in this Figure has been normalized by subtracting 1.132 mA from it. For an oxide oxide charge of 10^{10}cm^{-2} and a gate voltage of 0.4 volts the drain current is 1.13224 mA, while for a oxide charge of 10^{10}cm^{-2} the drain current is 1.13233. This change in drain current is not significant although it could be if low biasing conditions are needed. The I-V characteristics shown in this Figure are not very smooth although the trend is increasing upward. This nonsmoothness is due to numerical computation error.

8.5 Transconductance and Capacitance

Figure 25 is a plot of transconductance vs. drain voltage. The transconductance is a measure of how the drain current changes for a change in gate voltage. As the drain bias increases the transconductance of the device increases as shown in Figure 25. The transconductance at a drain bias of 3.0 volts is 0.4 millimhos while the transconductance is 4.0 millimhos at a drain bias of 5.0 volts. This is due to the fact at a drain bias of 5.0 volts the device is in saturation while at the drain bias of 3.0 volts the device is in the subthreshold region. This fact agrees well with results obtained for the pinch off voltage of the device (see Section 8.3).

The input capacitance vs. drain voltage is shown in Figure 26. This capacitance increase with drain voltage and thus also with drain current. This increase comes from the increase in drain current. With more drain current the charge in the channel is greater and thus the channel capacitance is greater. For example a drain bias of 2.0 volts corresponds to an input capacitance of 30 pF while at a drain bias of 5.0 volts the capacitance is 150 pF. This is do to the fact that 4.0 volts of drain voltage is indeed the pinch off voltage of the device and the input capacitance of the device is expected to increase sharply. The input capacitance is the parallel combination of the oxide capacitance and the channel capacitance. The input capacitance behaved similar to the transconductance. If the input capacitance is the major concern in the device the program can simulate different designs until the optimum, or desired, design is found.

The transconductance and input capacitance can be used to determine the maximum operating frequency of a device. This frequency is simply the transconductance divided by the input capacitance. For device 3, the maximum operating frequency is 4.2 MHz. This operating frequency is dependent on the input capacitance which is a function of the channel length. With smaller channel length the input capacitance decreases and correspondingly the maximum operating frequency increases. For device 3 with L=0.73 μm the operating frequency of 4.2 MHz can be increased by decreasing the channel length. The problem in doing this, as has been discussed before, is the short channel effects start to become very noticeable.

8.6 Future Work

This thesis is a three dimensional simulation of a short channel MOSFET's with the effects of the trapped oxide charge. Future work that may also be done in this area is a transient simulation of short channel MOSFET's, carrier heating in the channel region and different types of doping and fabrication structures.

TABLE 1

	DEVICE 1	DEVICE 2	DEVICE 3
Channel length	1.0 μm	1.0 μm	0.73 μm
Channel width	1.5 μm	1.5 μm	10.0 μm
Drain and source lengths	0.3 μm	0.3 μm	0.13 μm
Junction depth	0.3 μm	0.3 μm	0.13 μm
Oxide charge	10^{10} cm^{-2}	10^{10} cm^{-2}	10^{10} cm^{-2}
Oxide thickness	260 Å	260 Å	260 Å
Drain and source doping	10^{17}cm^{-3}	10^{17}cm^{-3}	10^{20}cm^{-3}
Substrate doping	10^{15}cm^{-3}	$7{\cdot}10^{15}$cm^{-3}	$7{\cdot}10^{16}$cm^{-3}
Height of device	1.5 μm	1.5μm	0.8 μm

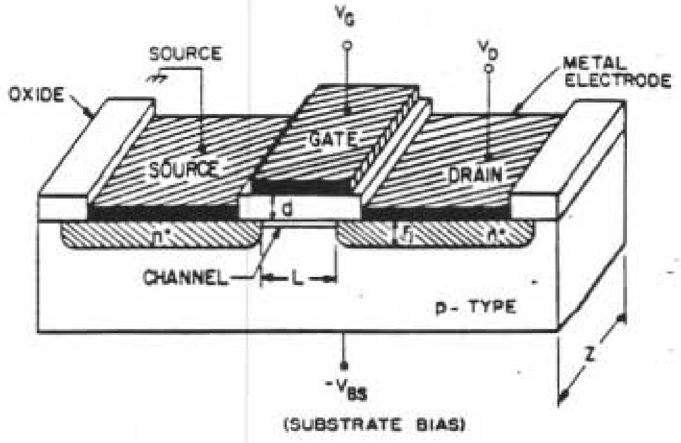

Figure 1. Representation of MOSFET geometry used in simulation.

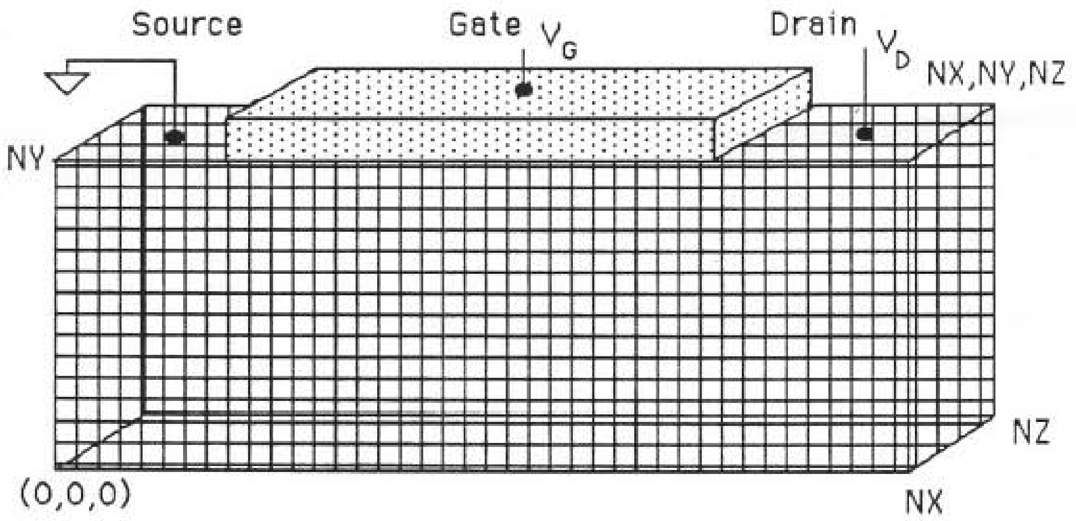

Figure 2. Simulation Domain.

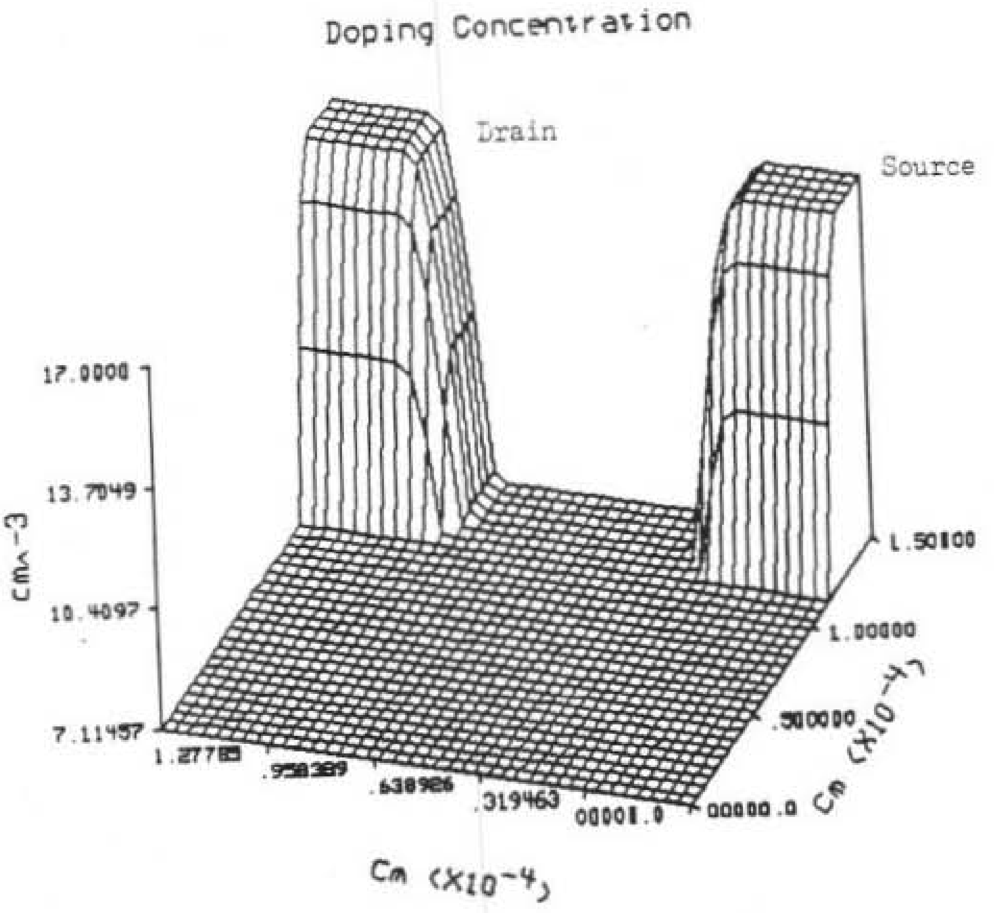

Figure 3. Doping profile [log., cm^{-3}] for device 1.

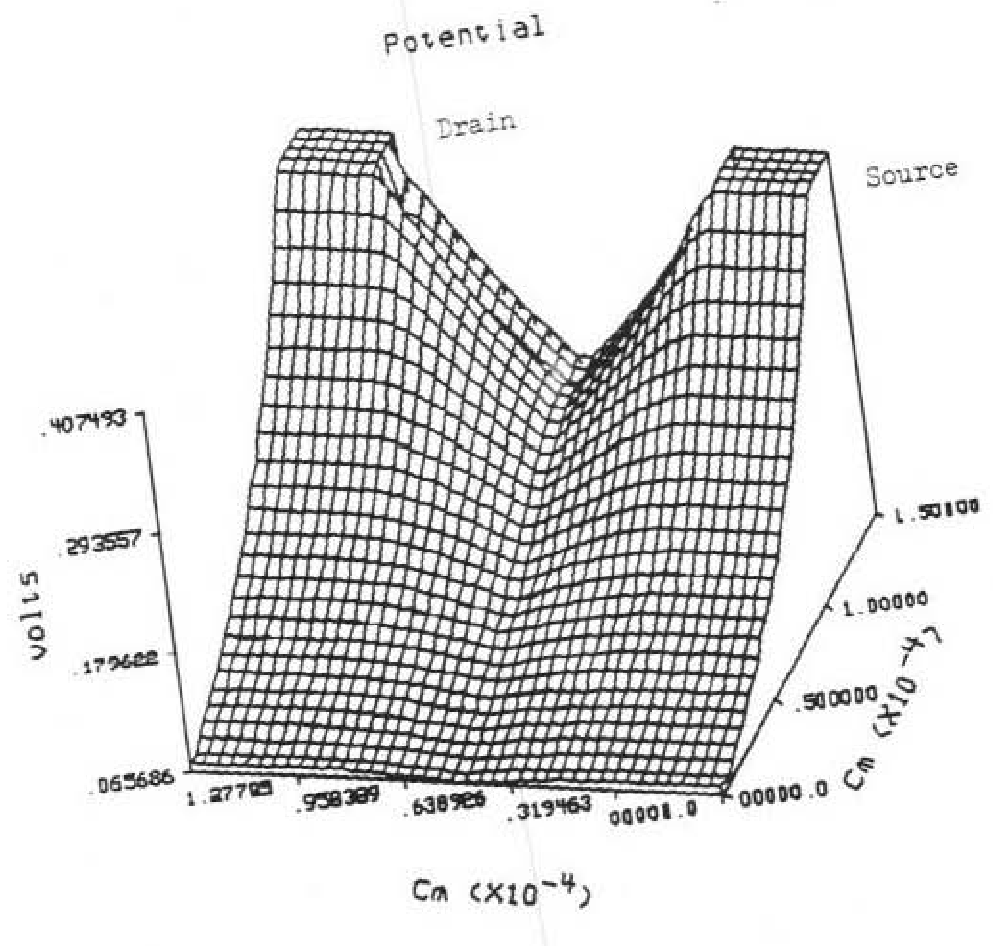

Figure 4. Electrostatic potential [V] for device 1 ($V_{gs}=V_{ds}=0$).

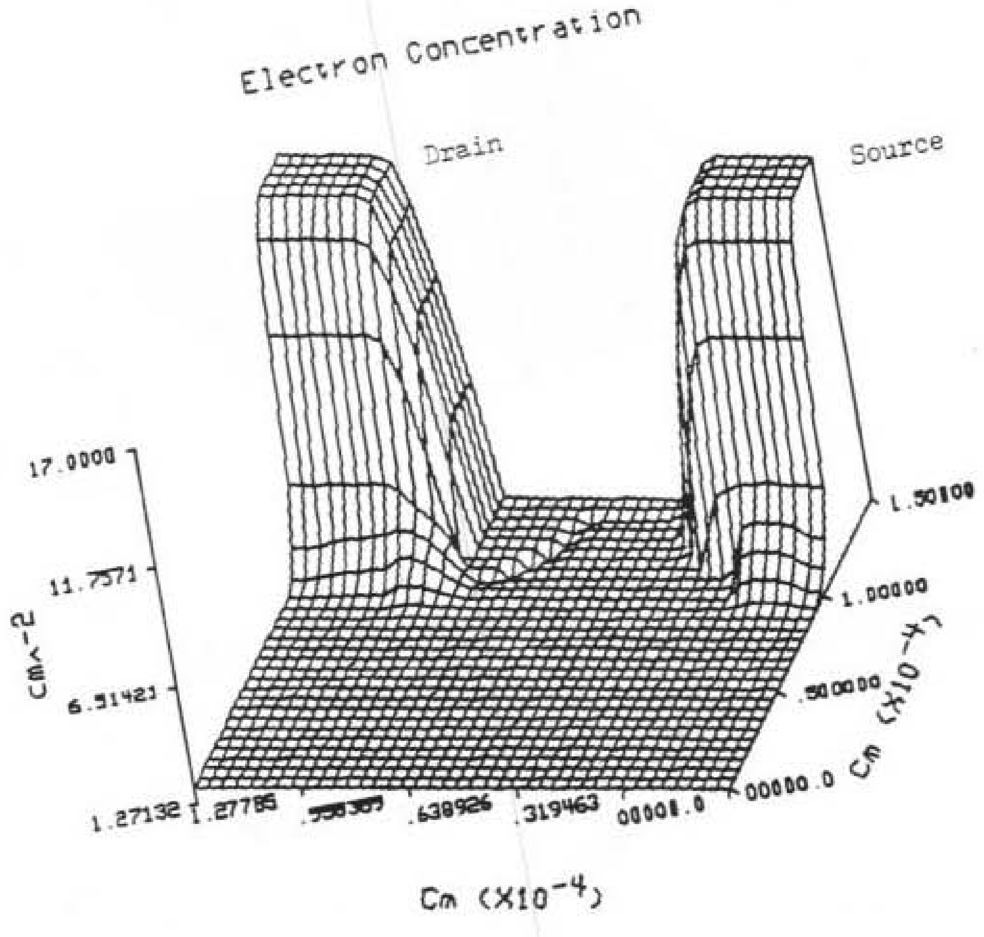

Figure 5. Concentration of electrons [log., cm^{-3}] for device 1.

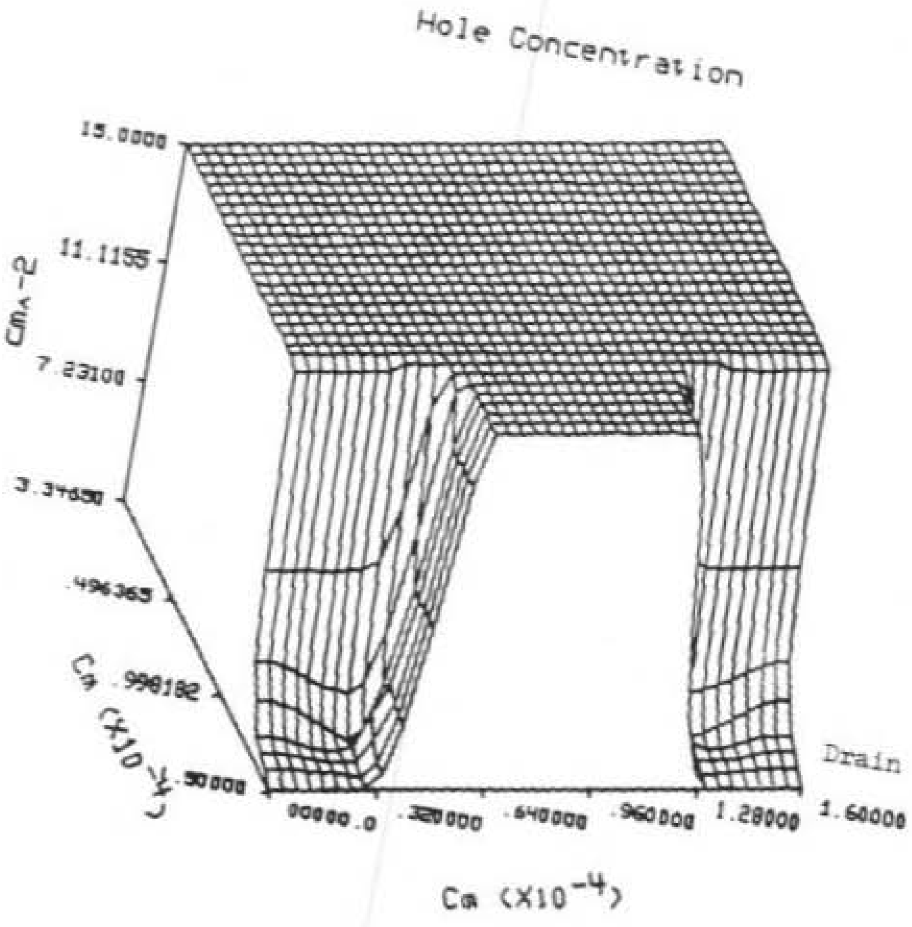

Figure 6. Concentration of holes for device 1 [log., cm^{-3}].

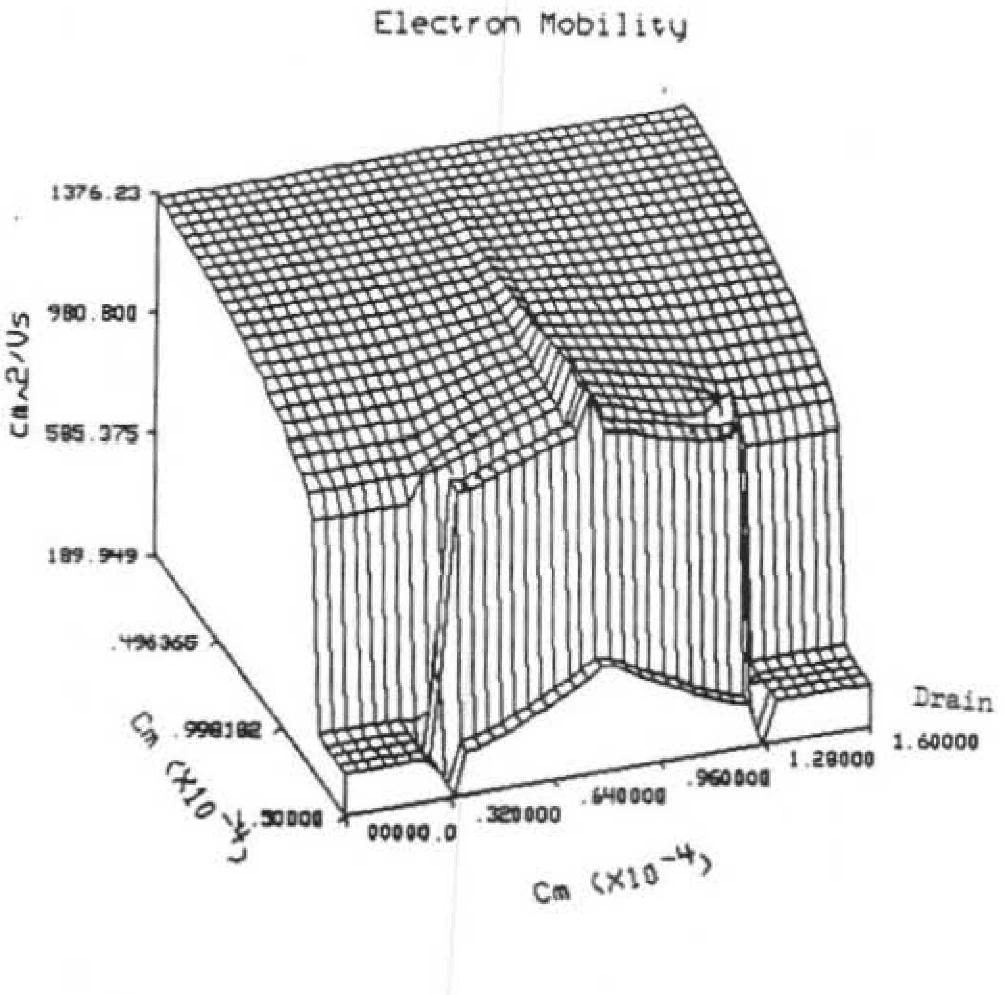

Figure 7. Electron mobility for device 1 [cm^2/Vs].

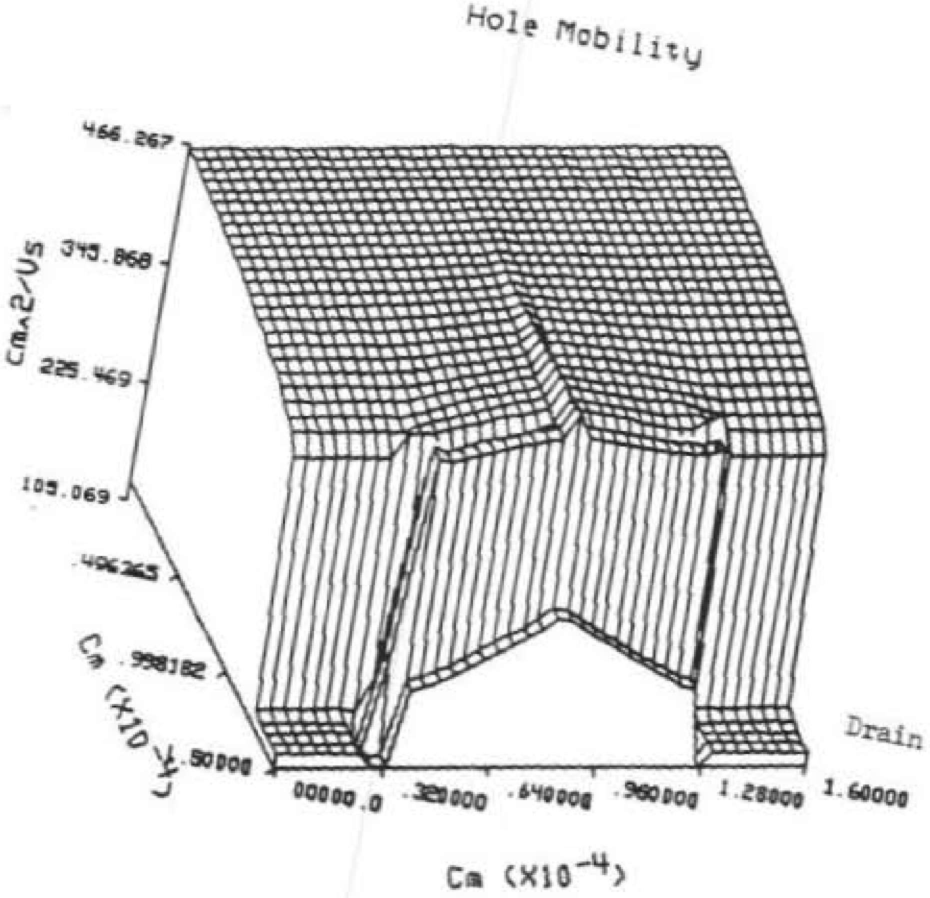

Figure 8. Hole mobility for device 1 [cm^2/Vs].

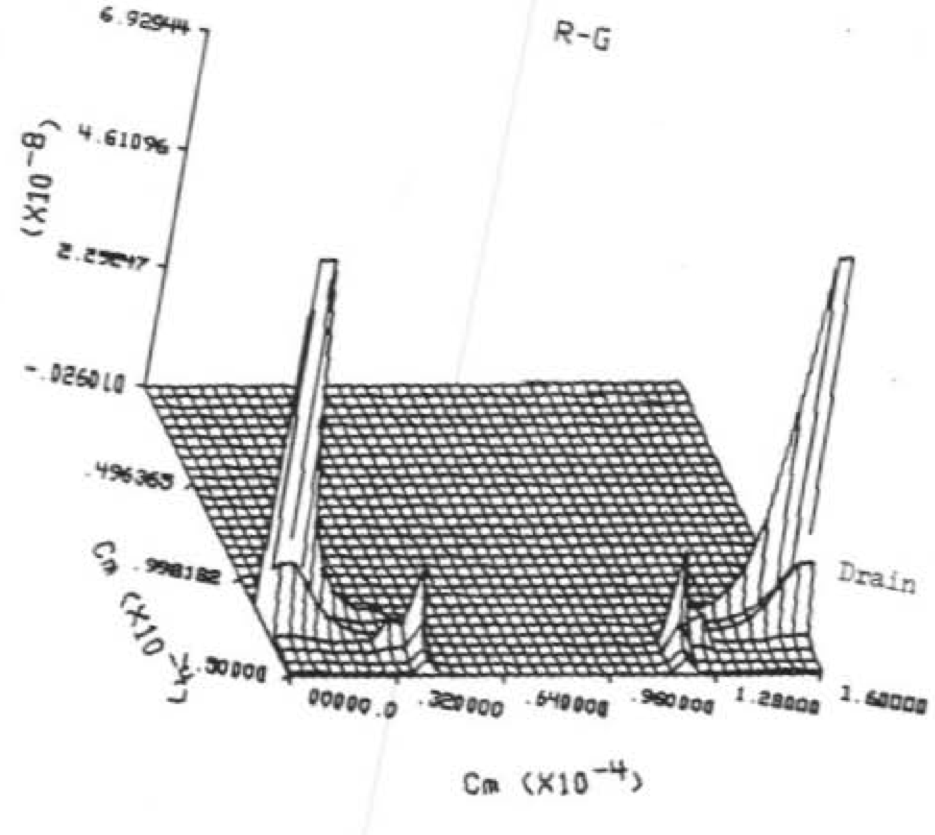

Figure 9. Net generation/recombination rate.

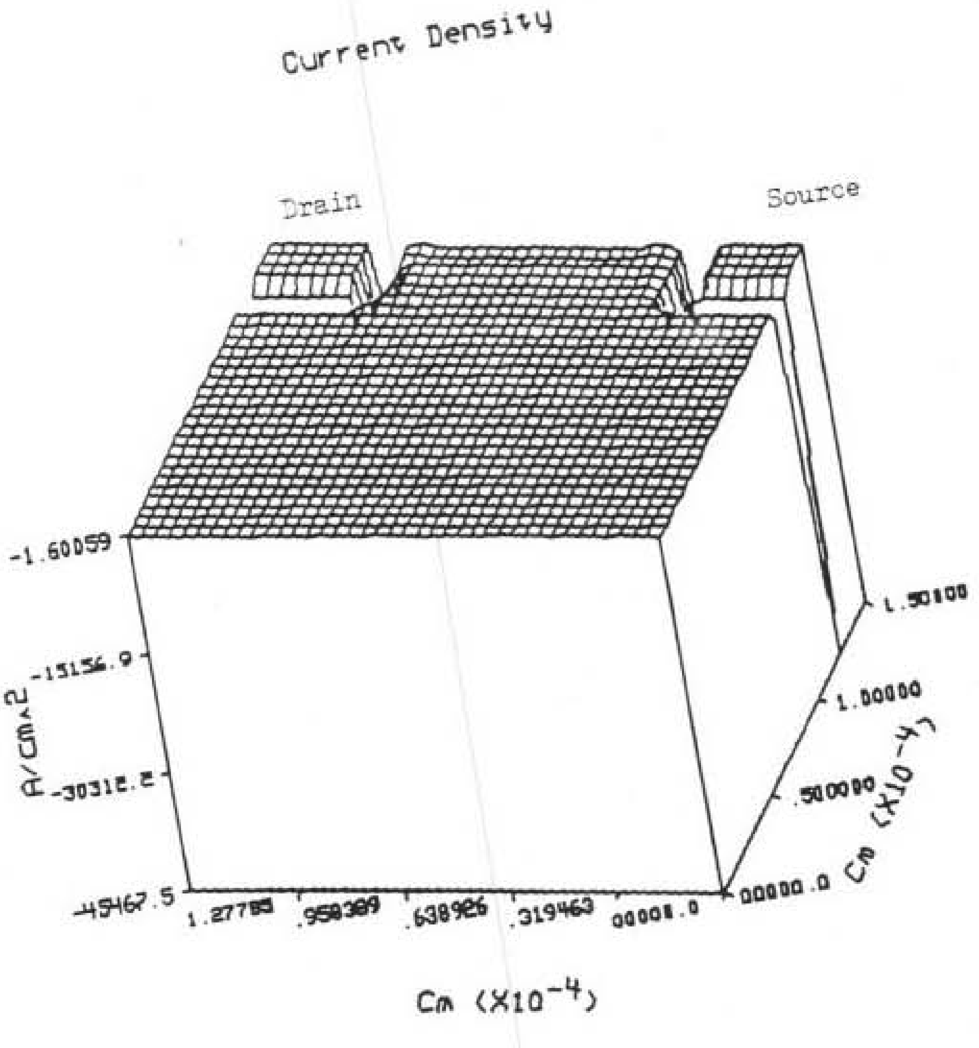

Figure 10. Magnitude of electron current density [Acm^{-2}] for device

1.

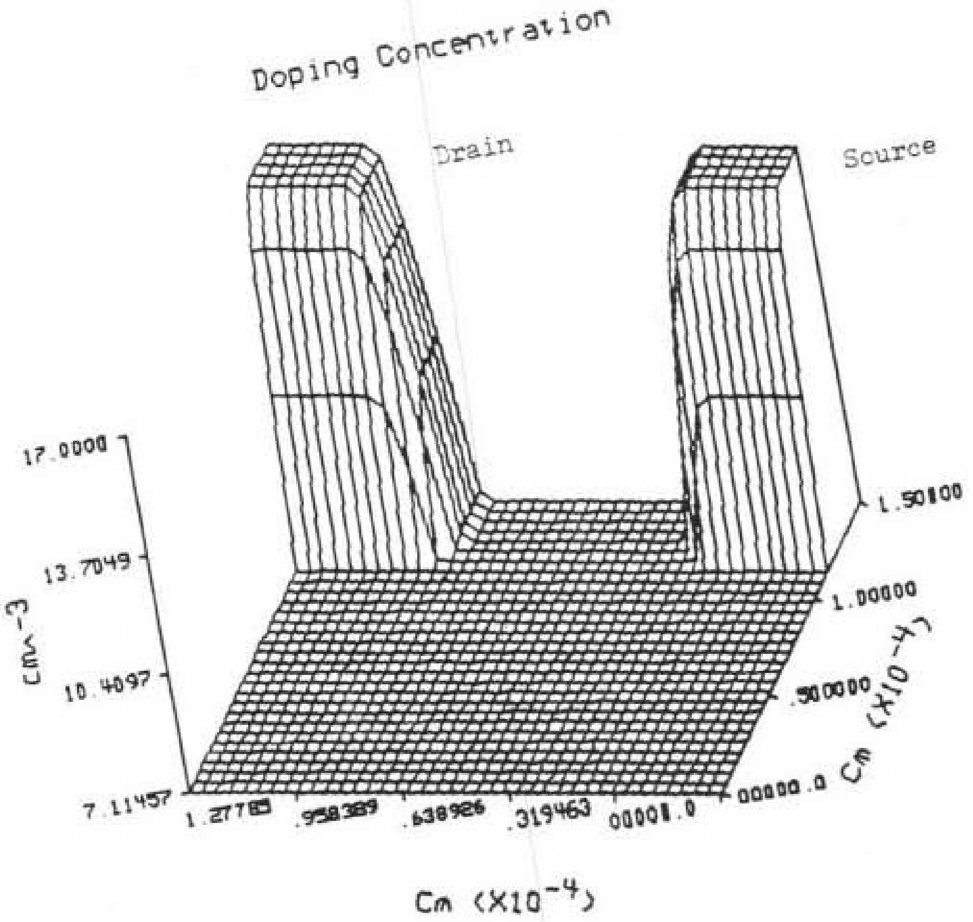

Figure 11. Doping concentration [log., cm^{-3}] for device 2.

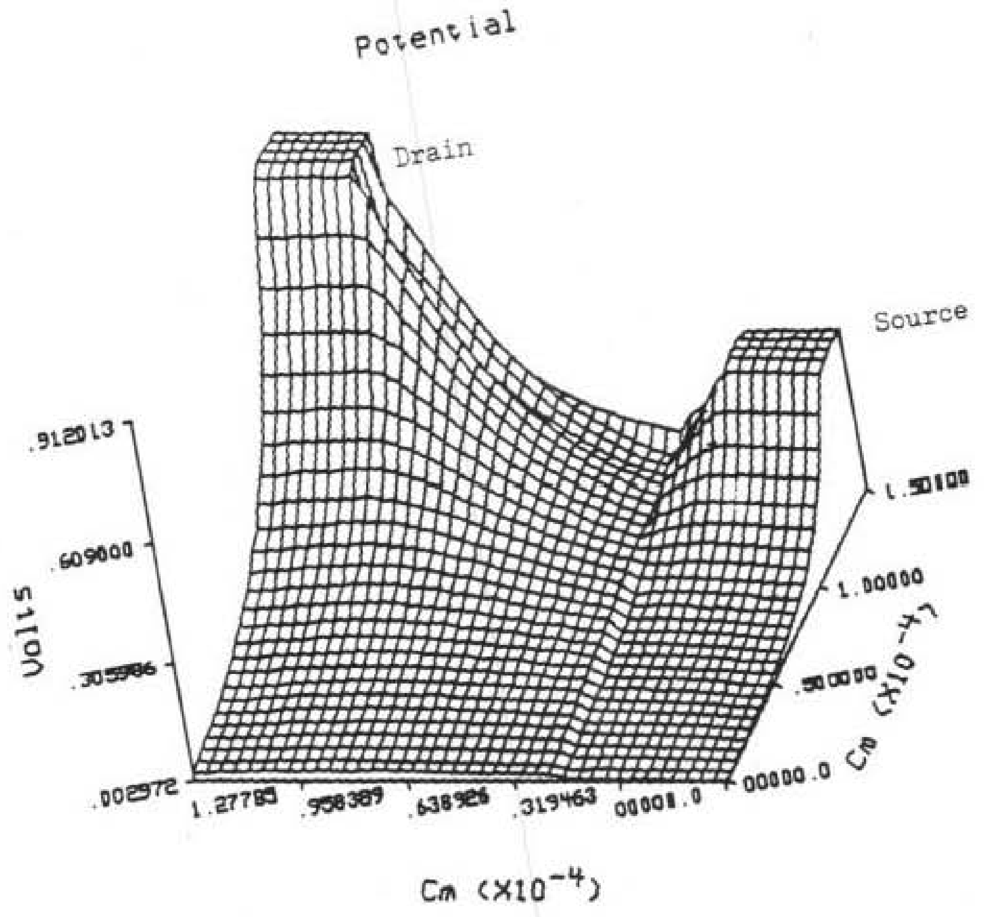

Figure 12. Electrostatic potential [V] for device 2 (V_{gs}=.2V, V_{ds}=.5V).

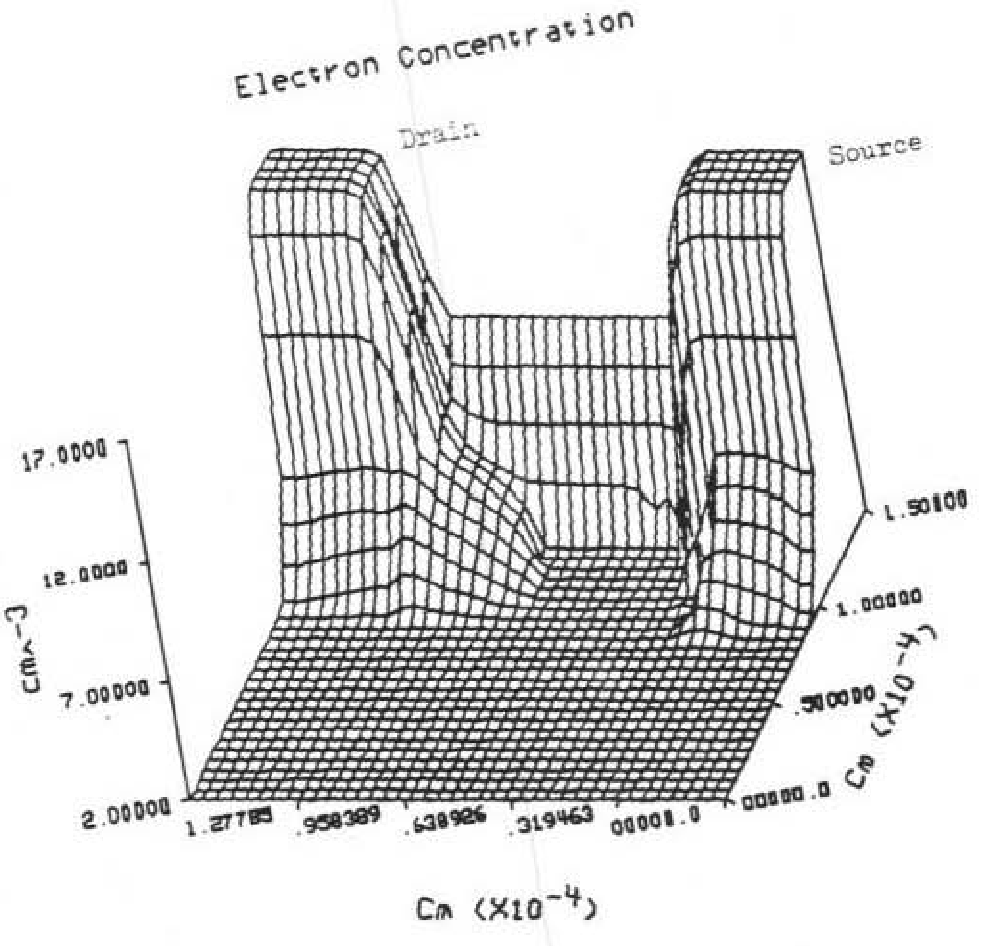

Figure 13. Concentration of electrons [log., cm^{-3}] for device 2

(V_{gs}=0.2V, V_{ds}=0.5V).

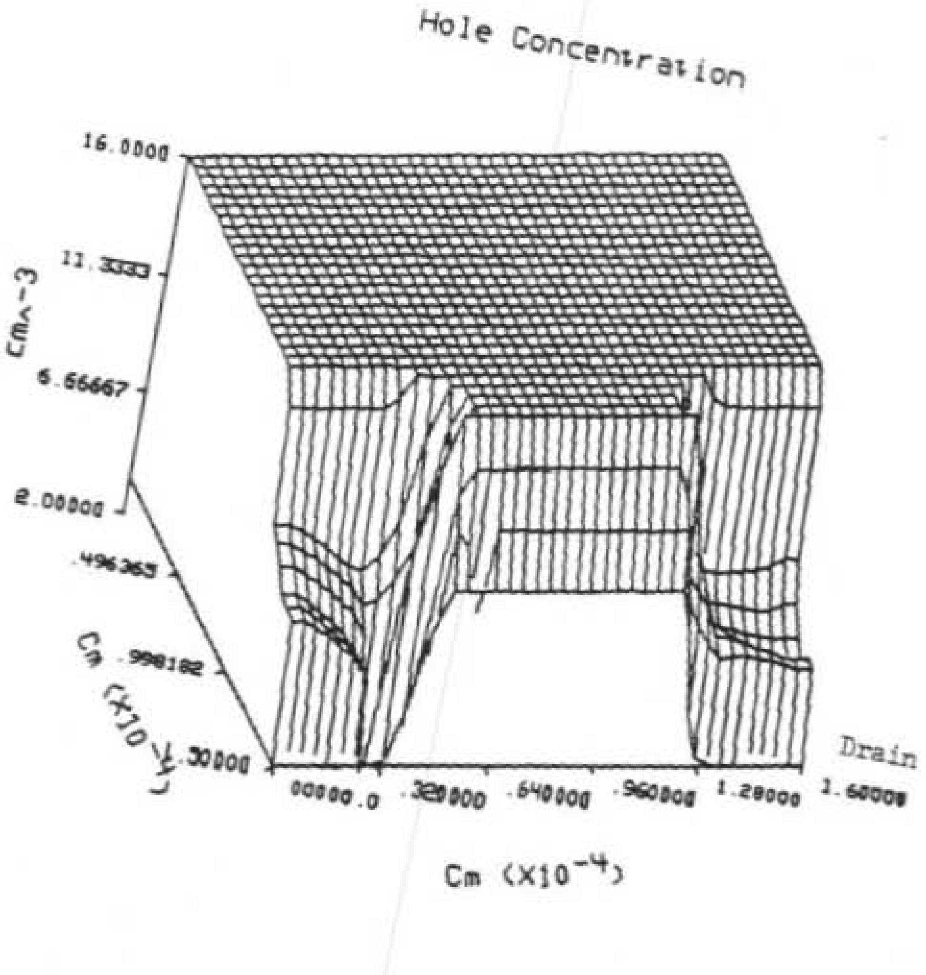

Figure 14. Concentration of holes [log., cm^{-3}] (V_{gs}=0.2V, V_{ds}=0.5V).

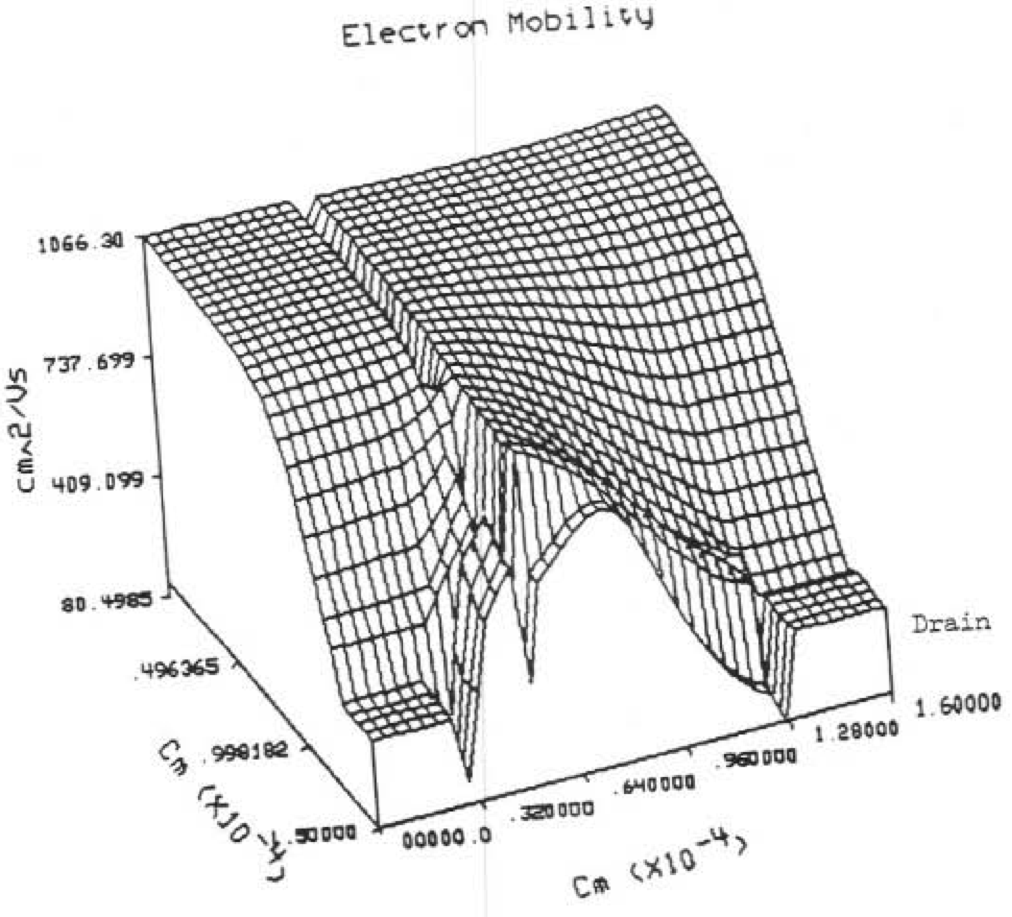

Figure 15. Electron Mobility for device 2.

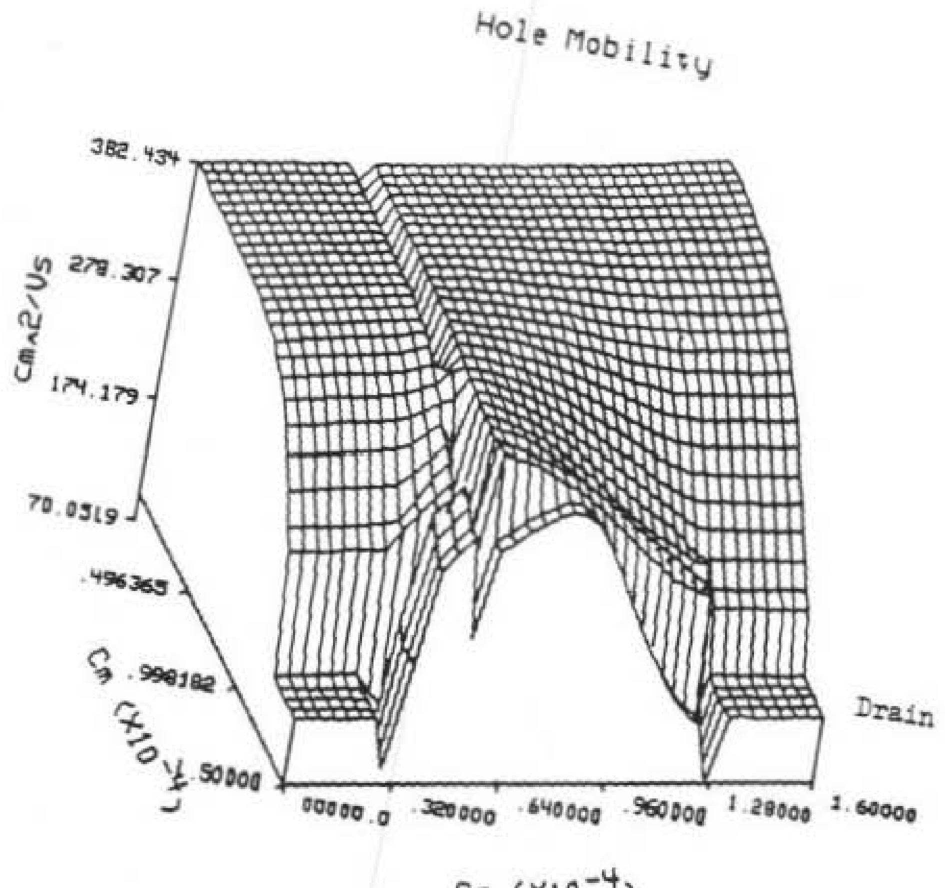

Figure 16. Hole mobility for device 2.

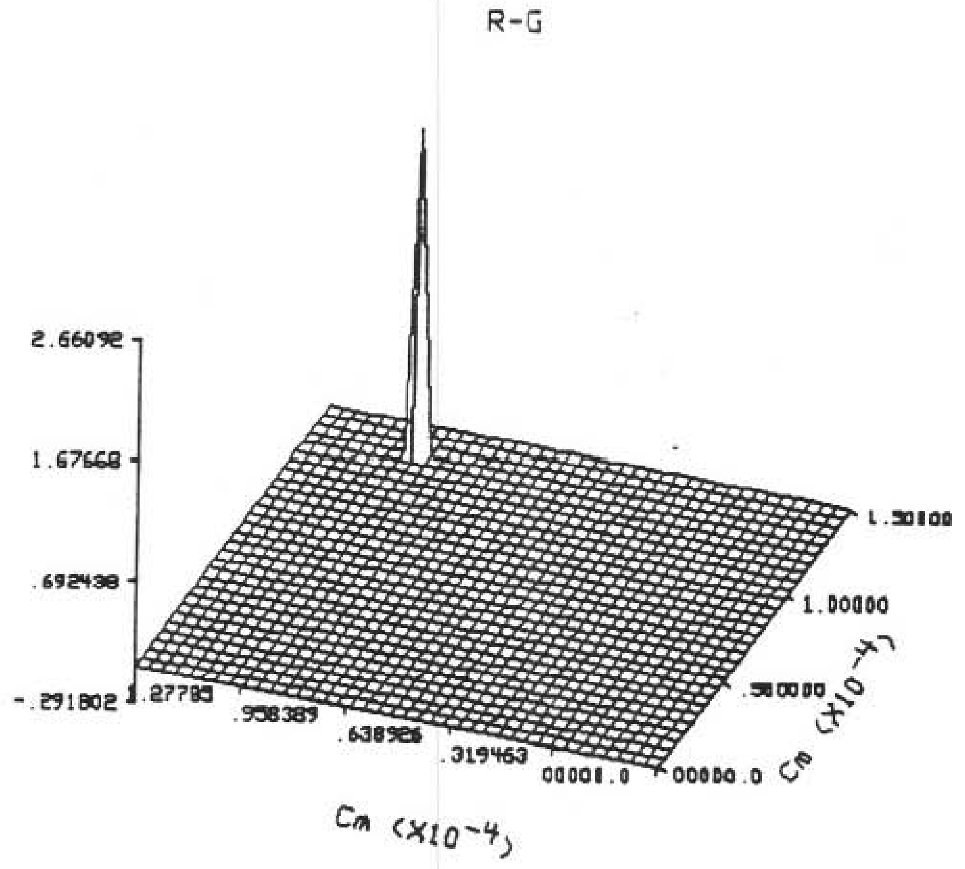

Figure 17. Net generation/recombination rate [cf. description of fig.]

(V_{gs}=0.2V, V_{ds}=0.5V).

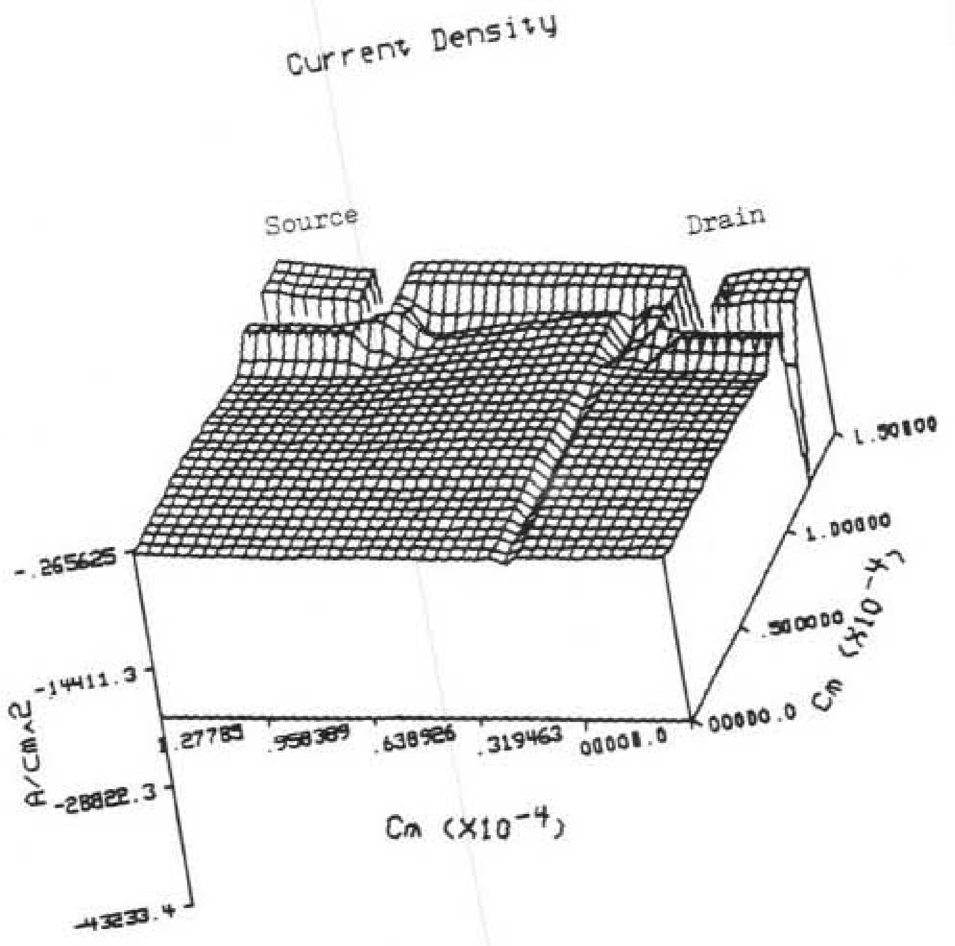

Figure 18. Magnitude of electron current density [Acm^{-2}] for device 2 (V_{gs}=0.2V, V_{ds}=0.5V).

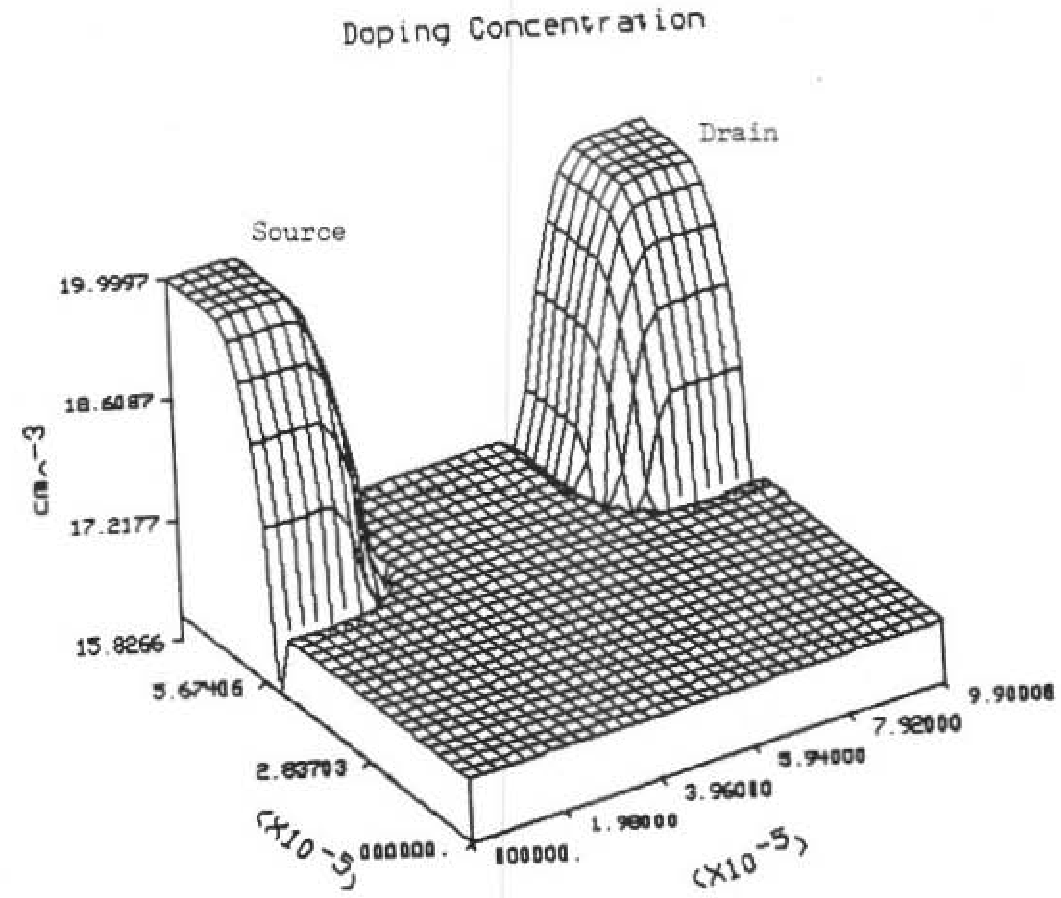

Figure 19. Doping profile [log.,cm-3] for device 3.

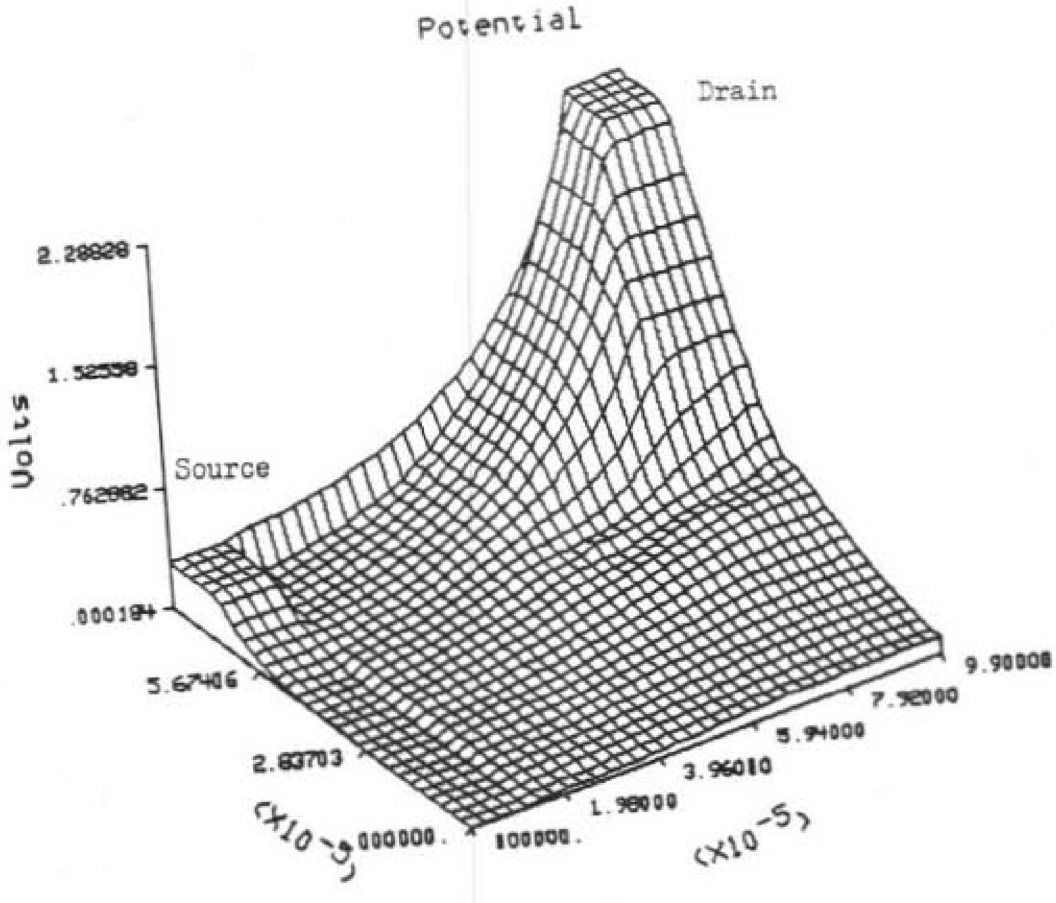

Figure 20. Electrostatic potential for device 3 (V_{gs}=1V, V_{ds}=2V).

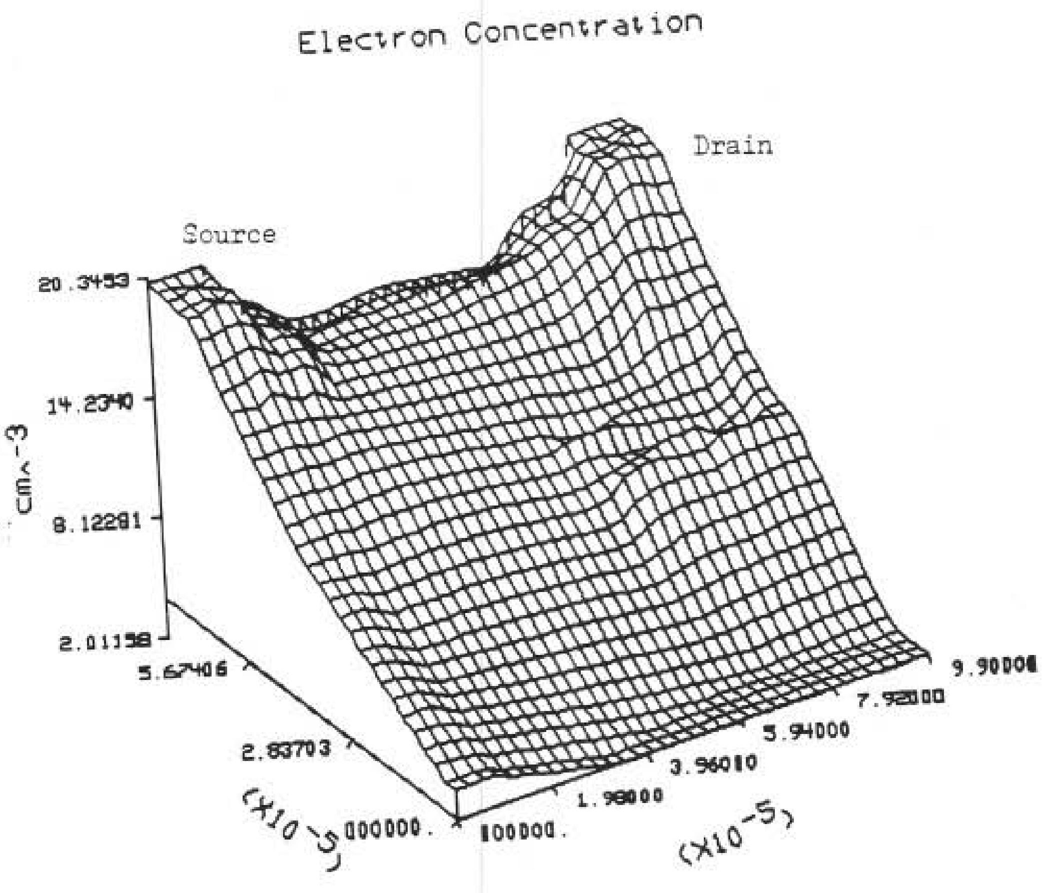

Figure 21. Concentration of electrons [log., cm^{-3}] for device 3.

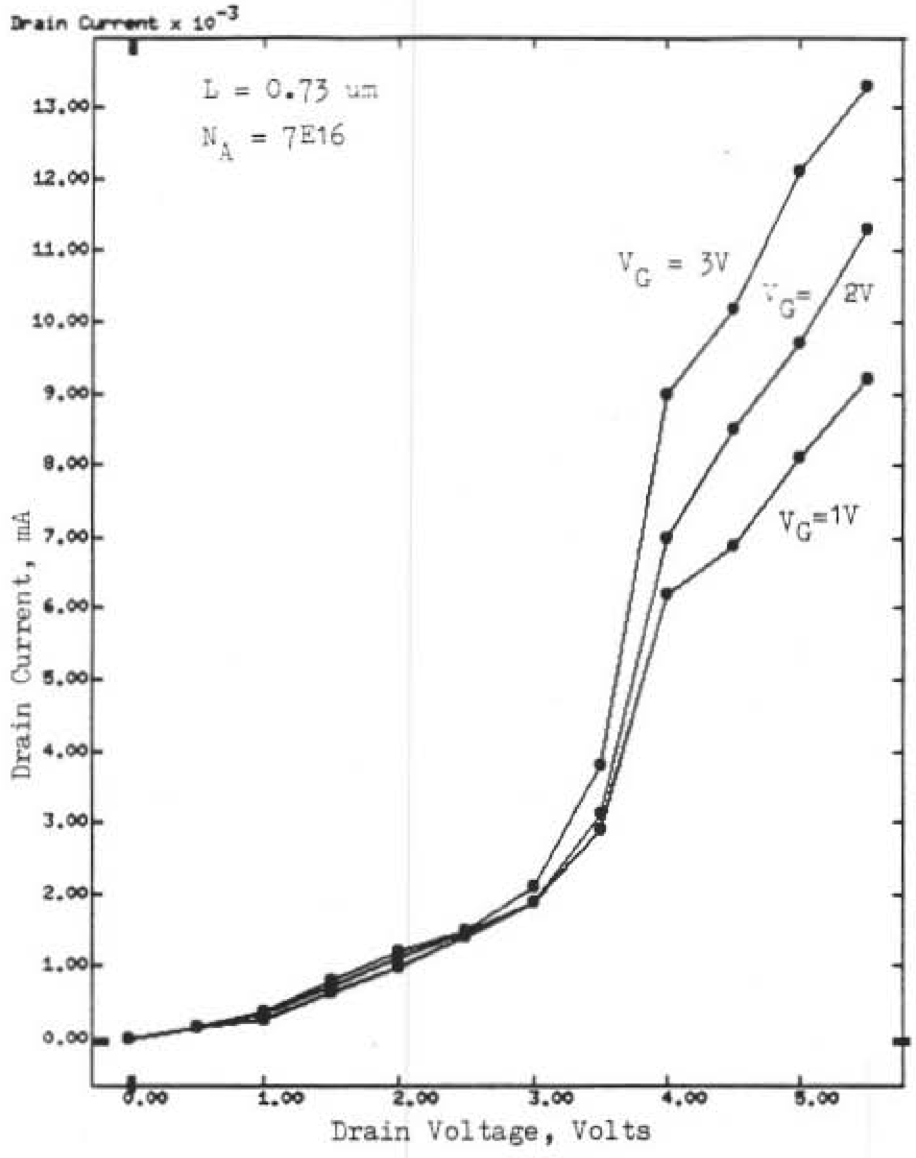

Figure 22. I_D vs. V_D for varying V_G.

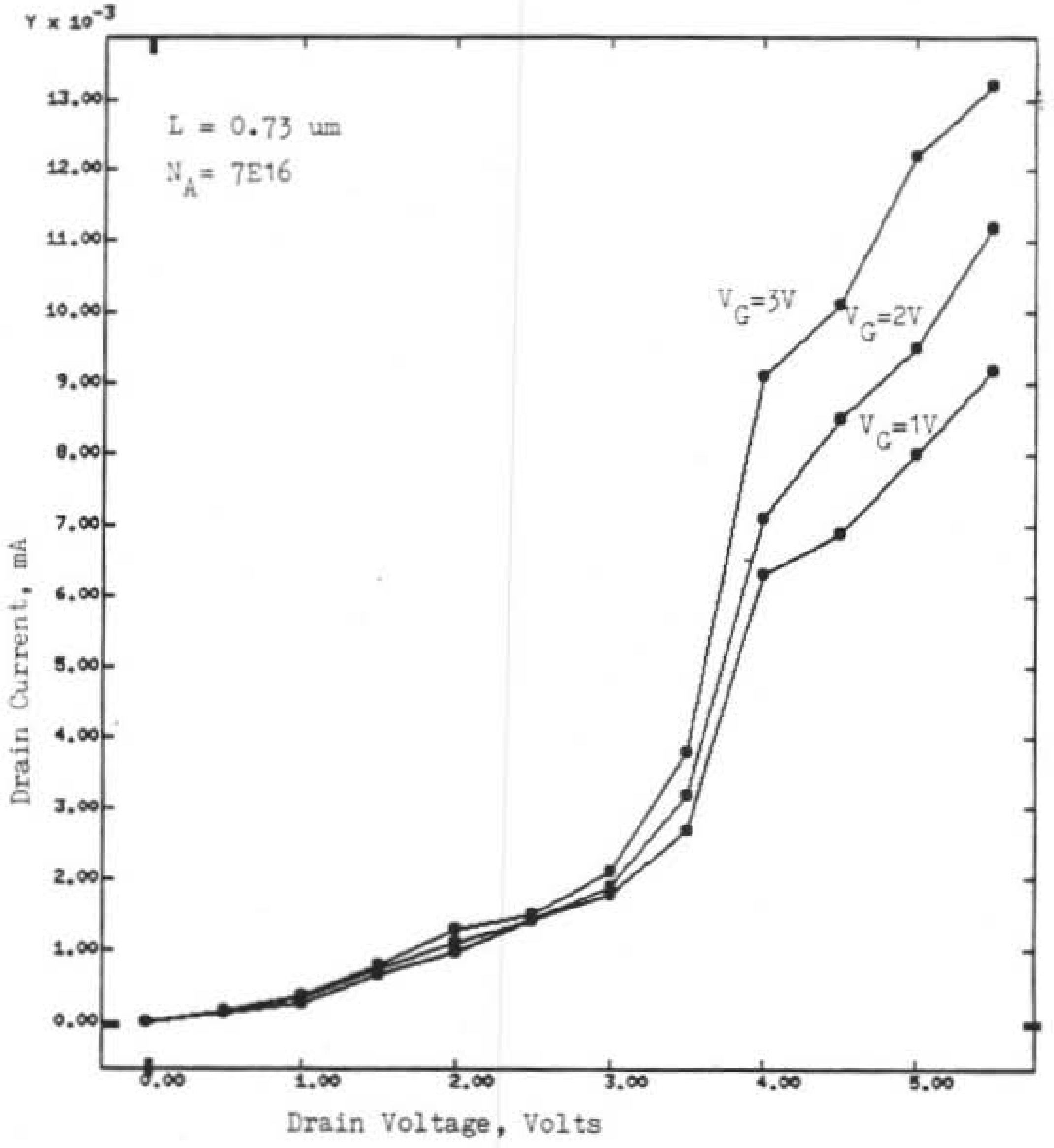

Figure 23. I_D vs. V_D for varying V_G without recombination term.

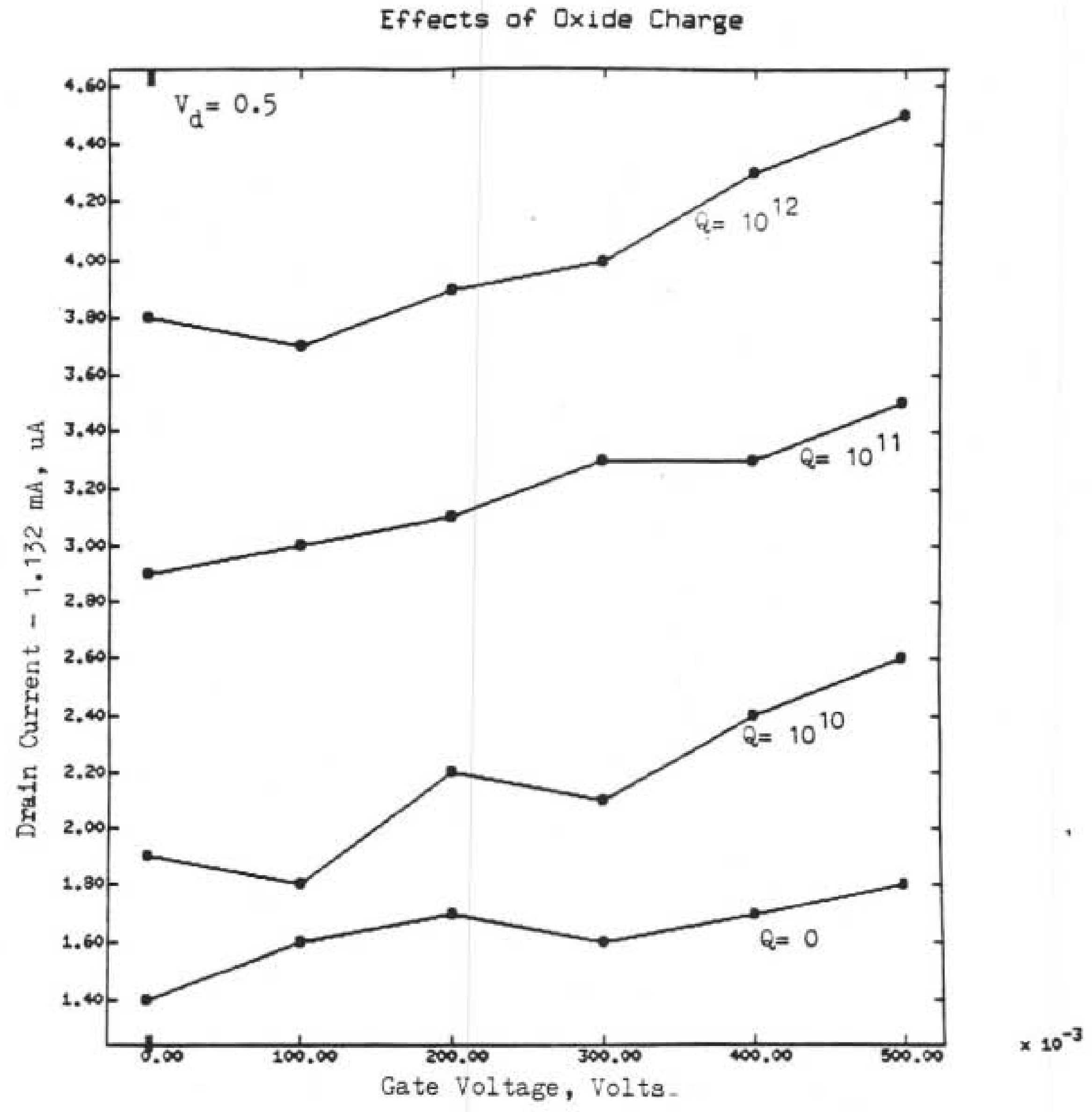

Figure 24. I_D vs. V_G with effects of oxide charge.

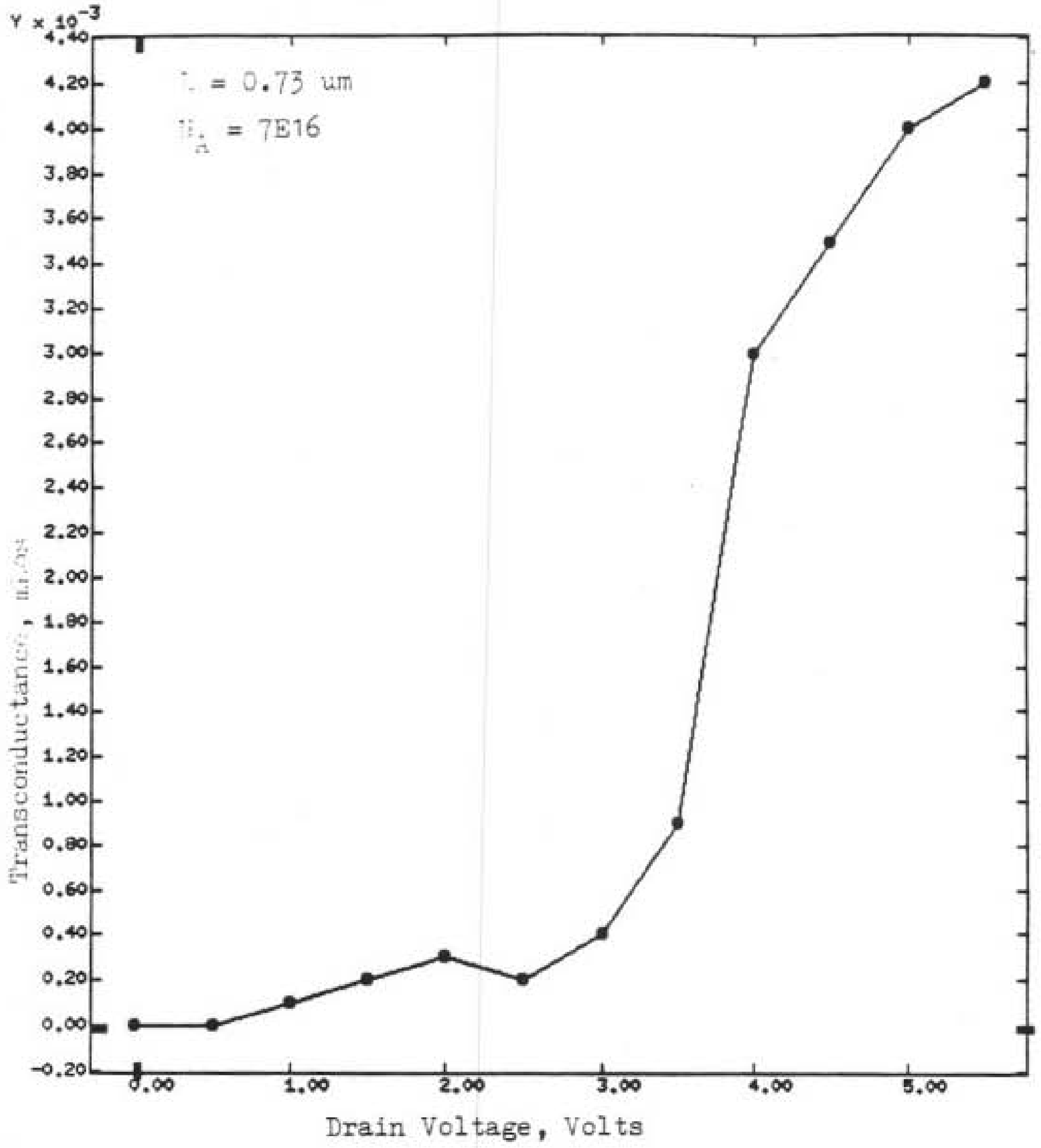

Figure 25. Transconductance vs. drain voltage.

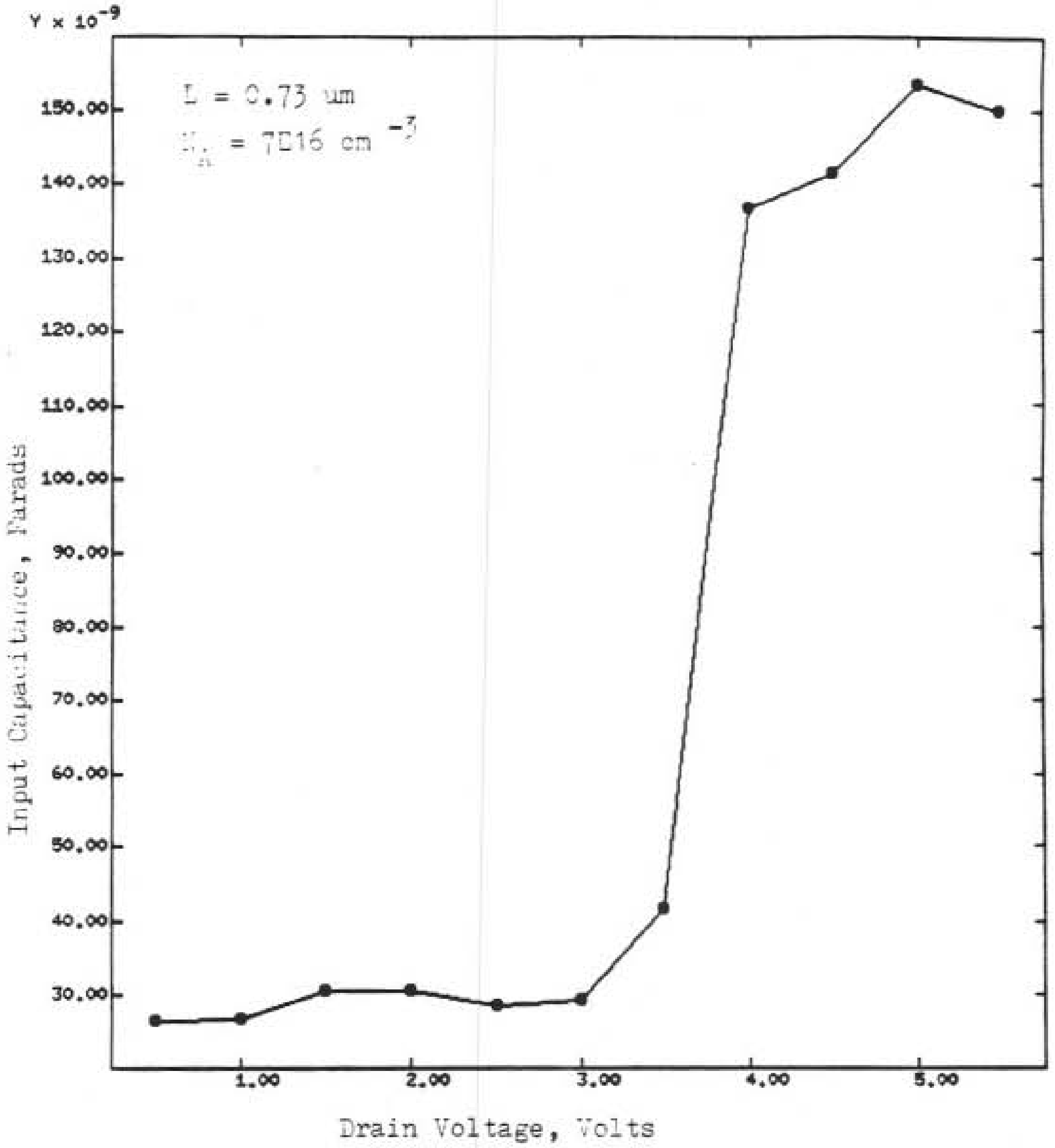

Figure 26. Input capacitance vs. drain voltage.

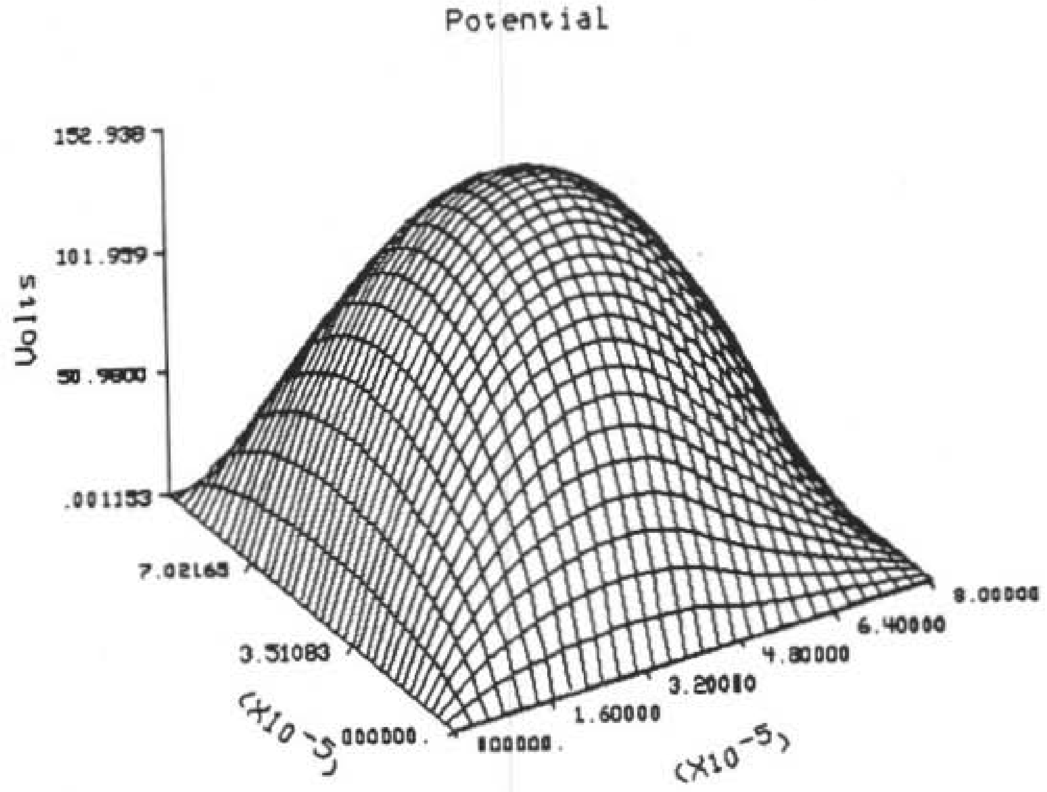

Figure 27. Diverging Solution.

REFERENCES

[1] Bank, R.E., Rose, D.J., and Fitchner, W., "Numerical methods for semiconductor device simulation," *IEEE Trans. Electron Devices*, vol. ED-30, pp.1031-1041, 1983.

[2] Conwell, E., *High Field Transport in Semiconductors*, New York: Academic Press, 1967.

[3] Fitchner, W., Rose, D.J., and Bank, R.E., "Semiconductor Device Simulation," *IEEE Trans. Electron Devices*, vol. ED-30, pp. 1018-1030, 1983.

[4] Selberherr, S., *Analysis and Simulation of Semiconductor Devices*, New York: Springer-Verlag Wein, p. 19, 1984.

[5] Navon, D. H., *Semiconductor Microdevices and Materials*, New York: CBS College Publishing, p. 87, 1986.

[6] Sze, S.M., *Physics of Semiconductor Devices*, New York: John Wiley and Sons, Inc., 1981.

[7] Gummel, H.K., *IEEE Trans. ElectronDevices*, vol. ED-11, p. 445, 1964.

[8] Slotboom, J.W., *Electron Lett.*, vol. 5, p. 667, 1969.

[9] Slotboom, J.W., *IEEE Trans. Electron Devices*, vol. ED-20, p.

669, 1973.

[10] Mock, M.S., *Solid-State Electron*, vol. 16, p. 601, 1973.

[11] Mock, M.S., *Journ. Eng. Math*, vol 7, p. 193, 1973.

[12] Heimeier, H. H., *IEEE Trans. Electron Devices*, vol. ED-20, p.669, 1973.

[13] Manck, O., Engl, W.E., *IEEE Trans. Electron Devices*, vol ED-22, p. 339, 1975.

[14] Navon, D.H., Wang, C.T.,"Numerical Modeling of Power MOSFET's", *Solid State Electron.* vol. 26, No. 4, pp.287-290, 1980.

[15] Wilson, C.L., Blue, J.L.,"Two-Dimensional Finite Element Charge-Sheet Model of a Short Channel MOS transistor.", *Solid State Electron. vol.* 25, No. 6, pp. 461-477, 1982.

[16] Schutz, A., Selberherr, S., Potzl, H.W.,"Analysis of Breakdown Phenomena in MOSFET's", *IEEE Trans. Computer-Aided-Design of Integrated Circuits CAD-1*, pp. 77-85,1982.

[17] Schutz, A., Selberherr, S., Potzl, H.W., "A Two-Dimensional Model of the Avalanche Effect in MOS Transistors", *Solid State Electron.* vol. 25, pp. 177-183, 1982.

[18] Buturla, E.M., Cottrell, P.E., Grossman, B.M., Salsburg, K.A., Lawlor, M.B., McMullen, C.T.,"Three-Dimensional Finite Element Simulation of Semiconductor Devices", *Proc. Int. Solid State Circuits Conf.*, pp. 76-77, 1980.

[19] Chamberlain, S.G., Husain, A.,"Three-Dimensional Numerical Simulation of VLSI MOSFET's", *Proc. Int. Electron Devices Meeting*, pp. 592-595, 1981.

[20] Shigyo, N., Konaka, M., Dang, R.L.M., "Three-Dimensional Simulation of inverse Narrow Channel Effect", *Electron. Lett.*, vol. 18, No. 6, pp. 274-275, 1982.

[21] Hayt, W.H., *Engineering Electromagnetics*, New York: McGraw-Hill Publishing, p. 62, 1981.

[22] Sah, C.T., Chan, P.C.H., Wang, C.K., Sah, R.L.Y., Yamakawa, K.A., Lutwack, R., "Effect of Zinc Impurity in Silicon Solar Cell Efficiency ",*IEEE Trans. Electron Devices* , vol ED-28, No. 3, pp. 304-313,1981.

[23] Conwell, E., Weisskopf, V.F.,"Theory of Impurity Scattering in Semiconductors", *Physical Review 77*, no. 3, pp. 388-390, 1950.

[24] Brooks, H., "Scattering by ionized Impurities in Semiconductors", *Physical Review 83*, p. 879, 1951.

[25] Scharfetter, D.L., Gummel, H.K., "Large Signal Analysis of a Silicon Read Diode Oscillator", *IEEE Trans. Electron Devices*, vol. ED-16, pp. 64-77, 1969.

[26] Selberherr, S., *Analysis and Simulation of Semiconductor Devices*, New York: Springer-Verlag Wein, pp.107-118, 1984.

[27] DeMari, A., "An Accurate Numerical Steady State One Dimenaional Solution of the P-N Juntion", *Solid State Electron*, vol. 11, pp. 33-58, 1968.

[28] Selberherr, S., *Analysis and Simulation of Semiconductor Devices*, New York: Springer-Verlag Wein, pp. 1984.

[29] Shockley, W., Read, W.T., "Statistics of the Recombination of Holes and Electrons", *Physical Review* 87, No. 5, 835-842, 1952.

[30] Hall, R.N., "Electron-Hole Recombination in Germanium", *Physical Review* 87, 387, 1952.

[31] Dzeiwior, J., Schmid, W., "Auger Coefficients for Highly Doped and Highly Excited Silicon", *Applied Physics Letters*

31, 346-348, 1977.

[32] Yau, L.D., "Simple I/V Model For Short-Channel I.G.F.E.T.S In The Triode Region", *Electronic Letters*, Vol. 11, No. 2, 1975.

[33] Kim, C.K., "The Physics of Charge-Coupled Devices", *Charge -Coupled Devices and Systems*, Wiley, New York, 1979, p.1.

[34] Hachtel, G.D., Mack, M.H., O'Brien, R.R., Speelpennin,B., "Semiconductor Analysis Using Finite Elements- Part 1", *Computational Aspects*, IBM J. Res. Dev. 25, 232-245, 1981.

[35] Hachtel, G.D., Mack, M.H., O'Brien, R.R., Speelpennin,B., "Semiconductor Analysis Using Finite Elements- Part 2", *IGFET and BJT Case Studies*, IBM J. Res. Dev. 25, 246-260, 1981.

[36] Fontana, T.P., Mcgregor, D.M., Lowther, R.P., *DEFINES: A Semiconductor Device Finite Element Simulation.* Electocon International Inc. 1982.

[37] Mock, M.S., *Analysis of Mathematical Models of Semiconductor Devices*, Dublin, Boole Press 1983.

[38] Sutherland, A.D., "On the Use of Overalaxation in Conjunction

with Gummel's Algorithm to Speed Convergence in a Two-Dimensional Computer Model for MOSFET's", *IEEE Trans. Electron Devices*, ED-27, 1297-1298, 1980.

[39] Franz, A.F., Franz, G.A., Selberherr, S., Ringhofer, C., Markowich, P., "Finite-Difference Method Suitable for Semiconductor Device Simulation", *IEEE Trans. Electron Devices*, ED-30, 1070-1082,1983.

[40] Stone, H.L., "Iterative Solution of Implicit Approximations of Multidimensional Partial Differential Equations", *S.I.A.M.J. Numerical Analysis* 5, 3, 530-558, 1968.

```
C-----------------------------------------------------------------------
C   Jake Baker
C
C   Thesis program
C
C-----------------------------------------------------------------------
C
          real*8   l
          real*8   xdist(60),ydist(60),zdist(60)
          real*8   si(35,30,15),n(35,30,15),p(35,30,15)
          real*8   mun(35,30,15),mup(35,30,15),doping(35,30,15)
          real*8    rg(35,30,15),jtot(35,30,15)
          real*8   xlambda2
          real*8   ersi(35,30,15),ern(30,25,3),erp(30,25,3)
C
          open(unit=14,file='ipsub.in',status='old')
          open(unit=15,file='total.out',status='old')
C
          call ipsub(temp,l,z,dsl,rj,vg,vd,qox,dopds,ssd,h,tox)
          write(6,*) ' Input parameters completed'
C
C         do 17 vg=1,3
C         do 16 vd=0.,5.5,.5
          call initgues1(temp,l,z,dsl,rj,vg,vd,qox,dopds,ssd,h,tox,
     +          wd,ws,wm,ys,yd,vbi,sis)
C         write(6,*) ' Initgues1 completed'
C
          call mesh(nx,ny,nz,xdist,ydist,zdist,l,z,dsl,rj,h,wd,ws,wm,vbi,sis,
     +          yd,doping,dopds,ssd)
C         write(6,*) ' Mesh routine completed',nx,ny,nz
C
          call diffuse(nx,ny,nz,xdist,ydist,zdist,rj,l,dsl,h,ssd,dopds,doping,
     +          n,p,si)
C         write(6,*) 'Diffuse completed'
C
          call initgues2(nx,ny,nz,xdist,ydist,zdist,rj,dsl,l,sis,wd,ws,wm,
     +          vbi,ssd,dopds,temp,si,n,p,vd,h)
C         write(6,*) ' Initgues2 completed'
C
C         do 10 iit=1,3
C         if(iit.eq.1)go to 18
             call boundcond(si,n,p,nx,ny,nz,xdist,ydist,zdist,temp,qox,l,
     +          z,dsl,vg,vd,tox,rg,ssd,mun,mup,doping,jtot)
C               write(6,*) ' Boundcond routine completed'
C
C
          call scaleit(si,n,p,doping,mun,mup,rg,xlambda2,temp,nx,ny,nz,
     +          xdist,ydist,zdist)
C         write(6,*) ' Scaleit routine completed'
C
          call soljacsi(nx,ny,nz,xdist,ydist,zdist,si,n,p,doping,xlambda2,
     +          mup,mun,rg,ersi)
C         write(6,*) ' Soljacsi routine completed'
C
          call unscaleit(si,n,p,doping,mun,mup,rg,xlambda2,temp,nx,ny,nz,xdist,
     +          ydist,zdist)
C         write(6,*) ' Unscaleit completed'
C
          go to 66
18        continue
          call boundcond(si,n,p,nx,ny,nz,xdist,ydist,zdist,temp,qox,l,
     +          z,dsl,vg,vd,tox,rg,ssd,mun,mup,doping,jtot)
C         write(6,*) ' Boundcond routine completed'
C
70        call mobility(mun,mup,doping,si,n,p,nx,ny,nz,temp,xdist,ydist,zdist)
C         write(6,*) ' Mobility routine completed'
```

```
C
C          call recgen(si,n,p,nx,ny,nz,temp,xdist,ydist,zdist,rg,doping,mun,
C     +          mup,jtot)
C          write(6,*) ' Recgen routine completed'
C
           call boundcond(si,n,p,nx,ny,nz,xdist,ydist,zdist,temp,qox,l,
      +          z,dsl,vg,vd,tox,rg,ssd,mun,mup,doping,jtot)
C          write(6,*) ' Boundcond routine completed'
C
           call scaleit(si,n,p,doping,mun,mup,rg,xlambda2,temp,nx,ny,nz,
      +          xdist,ydist,zdist)
C          write(6,*) ' Scaleit routine completed'
C
           call soljacn(nx,ny,nz,xdist,ydist,zdist,si,n,p,doping,xlambda2,
      +          mup,mun,rg,ern,dopds)
C          write(6,*) ' Soljacn routine completed'
C
           call unscaleit(si,n,p,doping,mun,mup,rg,xlambda2,temp,nx,ny,nz,xdist,
      +          ydist,zdist)
C          write(6,*) ' Unscaleit completed'
C
           call boundcond(si,n,p,nx,ny,nz,xdist,ydist,zdist,temp,qox,l,
      +          z,dsl,vg,vd,tox,rg,ssd,mun,mup,doping,jtot)
C          write(6,*) ' Boundcond routine completed'
C
           call mobility(mun,mup,doping,si,n,p,nx,ny,nz,temp,xdist,ydist,zdist)
C          write(6,*) ' Mobility routine completed'
C
           call recgen(si,n,p,nx,ny,nz,temp,xdist,ydist,zdist,rg,doping,mun,
      +          mup,jtot)
C          write(6,*) ' Recgen routine completed'
C
           call boundcond(si,n,p,nx,ny,nz,xdist,ydist,zdist,temp,qox,l,
      +          z,dsl,vg,vd,tox,rg,ssd,mun,mup,doping,jtot)
C          write(6,*) 'Boundcond completed'
C
           call scaleit(si,n,p,doping,mun,mup,rg,xlambda2,temp,nx,ny,nz,
      +          xdist,ydist,zdist)
C          write(6,*) ' Scaleit routine completed'
C
           call soljacp(nx,ny,nz,xdist,ydist,zdist,si,n,p,doping,xlambda2,
      +          mup,mun,rg,erp,ssd)
C          write(6,*) ' Soljacp routine completed'
C
           call unscaleit(si,n,p,doping,mun,mup,rg,xlambda2,temp,nx,ny,nz,xdist,
      +          ydist,zdist)
C          write(6,*) ' Unscaleit completed'
10         continue
C
           call boundcond(si,n,p,nx,ny,nz,xdist,ydist,zdist,temp,qox,l,
      +          z,dsl,vg,vd,tox,rg,ssd,mun,mup,doping,jtot)
C          write(6,*) 'Boundcond completed'
C
                current=0.0
C
           do 101 iz=2,nz-1
                do 102 iy=2,ny-1
                   current=jtot(25,iy,iz)*(ydist(iy+1)-ydist(iy)
      +            )*(zdist(iz+1)-zdist(iz))+current
102             continue
101        continue
           write(6,*)vg,current,qox
           current=abs(current)
           write(15,*)vd,current
16         continue
           write(15,*)
```

```
17      continue
66      call printr(si,n,p,xdist,ydist,zdist,nx,ny,nz,doping,mun,mup,rg,temp
     +              ,jtot)
        write(6,*) ' Print to file routine completed'
        stop
        end
C
C-----------------------------------------------------------------------
C   subroutine  initguess calulates the initial guess for the solution
C-----------------------------------------------------------------------
C
        subroutine initgues1(temp,l,z,dsl,rj,vg,vd,qox,dopds,ssd,h,tox,
     +           wd,ws,wm,ys,yd,vbi,sis)
C
        real*8 l
        vbip=(1.38e-23*temp/1.6e-19)*(log(ssd/1.45e15))
        sis=vbip*2.0
        vbi=(1.38e-23*temp/1.6e-19)*(log(dopds/1.45e15))
        cont=2.0*11.9*8.85e-14/(1.6e-19*ssd)
        wd=sqrt(cont*(vd+vbi))
        ws=sqrt(cont*vbi)
        wm=sqrt(cont*sis)
        go to 10
        write(6,*)
        write(6,*) ' SIS = ',sis,' cm'
        write(6,*) ' VBI = ',vbi,' cm'
        write(6,*) ' CONT =',cont,' cm'
        write(6,*) ' WD = ',wd,' cm'
        write(6,*) ' WS = ',ws,' cm'
        write(6,*) ' WM = ',wm,' cm'
        write(6,*) ' YD = ',yd,' cm'
10      write(6,*)
        return
        end
C
C-----------------------------------------------------------------------
C  subroutine initgues2 puts initial guess into determine mesh points
C-----------------------------------------------------------------------
C
        subroutine initgues2(nx,ny,nz,xdist,ydist,zdist,rj,dsl,l,sis,wd,ws,wm,
     +           vbi,ssd,dopds,temp,si,n,p,vd,h)
        real*8  xdist(60),ydist(60),zdist(60),l
        real*8  si(35,30,15),n(35,30,15),p(35,30,15)
C
        do 10 iz=1,nz
           do 20 iy=1,ny
                do 30 ix=1,nx
C
                   if (xdist(ix).le.dsl) then
                        if (ydist(iy).ge.(h-rj)) then
                           si(ix,iy,iz)=vbi
                        else
                           si(ix,iy,iz)=vbi*exp(-((h-rj-ydist(iy)))/ws)
                        end if
                   elseif (xdist(ix).ge.(dsl+l)) then
                        if (ydist(iy).ge.(h-rj)) then
                           si(ix,iy,iz)=vbi+vd
                        else
                           si(ix,iy,iz)=(vbi+vd)*exp(-((h-rj-ydist(iy)))
     +                          /wd)
                        end if
                   else
                        if (xdist(ix).le.(dsl)) then
                           if (ydist(iy).ge.(h-rj)) then
                                si(ix,iy,iz)=vbi*exp(-(sqrt((xdist(ix)-
     +                          dsl)**2.0)/ws))
```

```
                              else
                                   si(ix,iy,iz)=vbi*exp(-(sqrt((xdist(ix)-dsl)
     +                             **2.0+(h-rj-ydist(iy))**2.0))/ws)
                              end if
                           else
                              if (ydist(iy).ge.(h-rj)) then
                                   si(ix,iy,iz)=(vbi+vd)*exp(-(sqrt((-xdist(ix)
     +                             +l+dsl)**2.0)/wd))
                              else
                                   si(ix,iy,iz)=(vbi+vd)*exp(-(sqrt((l+dsl-
     +                                     xdist(ix))
     +                                     **2.0+(h-rj-ydist(iy))**2.0))/wd)
                              end if
                           end if
                      end if
C
30                 continue
20            continue
10         continue
           write(6,*) 'ws= ',ws
           write(6,*) 'wd= ',wd
           write(6,*) 'wm= ',wm
           return
           end
C
C-------------------------------------------------------------------------------
C   subroutine ipsub reads input values from a file.
C-------------------------------------------------------------------------------
C
           subroutine ipsub(temp,l,z,dsl,rj,vg,vd,qox,dopds,ssd,h,tox)
           real*8 l
C
           read(14,10) temp,l,z,dsl,rj,vg,vd,qox,dopds,ssd,h,tox
10         format(1215e15)
           write(6,*) ' Temperature: ',temp
           write(6,*) ' Channel length: ',l
           write(6,*) ' Channel width: ',z
           write(6,*) ' Drain and source length: ',dsl
           write(6,*) ' Junction depth: ',rj
           write(6,*) ' Gate voltage: ',vg
           write(6,*) ' Drain voltage: ',vd
           write(6,*) ' Oxide charge: ',qox
           write(6,*) ' Drain and source doping: ',dopds
           write(6,*) ' Substrate doping: ',ssd
           write(6,*) ' Height of device: ',h
           write(6,*) ' Oxide thickness: ',tox
           write(6,*) ' Diffusion process: ',diftype
           write(6,*)
           return
           end
C
C--------------------------------------------------------------------------------
C    subroutine mesh determines the Mesh points necessary for simulation for
C    short channel MOSFET's
C--------------------------------------------------------------------------------
C
           subroutine mesh(nx,ny,nz,xdist,ydist,zdist,l,z,dsl,rj,h,wd,ws,wm,vbi,
     +            sis,yd,doping,dopds,ssd)
           real*8 l,xdist(60),ydist(60),zdist(60),doping(35,30,15)
C
           nx=35
           ny=30
           nz=15
           do 10 iz=1,nz
                  zdist(iz)=(z*(iz-1)/(nz-1))
10         continue
```

```
        do 20 iy=1,ny
                ydist(iy)=h*(iy-1)/(ny-1)
20      continue
        do 50 ix=1,nx
                xdist(ix)=((2.0*dsl+1)*(ix-1)/(nx-1.0))
50      continue
        return
        end
c
c-------------------------------------------------------------------------
c  subroutine diffuse computes the diffusion doping profile.
c-------------------------------------------------------------------------
c
        subroutine diffuse(nx,ny,nz,xdist,ydist,zdist,rj,l,dsl,h,ssd,dopds,
     +          doping,n,p,si)
        real*8 xdist(60),ydist(60),zdist(60),doping(35,30,15),l
        real*8 n(35,30,15),p(35,30,15),si(35,30,15)
c
c   calculate the doping values.
c
        do 15 iz=1,nz
           do 25 iy=1,ny
                do 35 ix=1,nx
         if((xdist(ix).le.dsl).or.(xdist(ix).ge.
     +                  (dsl+l))) then
c
                if(ydist(iy).ge.(h-rj))then
                        doping(ix,iy,iz)=dopds-ssd
c
                else
                        hh=-ydist(iy)+h-rj
c
                        doping(ix,iy,iz)=dopds*exp((-hh**2)/3e-11)-ssd
                end if
c
                else
c
                if(ydist(iy).gt.h-rj)then
                        doping(ix,iy,iz)=dopds*(exp((-(-dsl+xdist(ix))**2
     +                          )/3e-11)+exp((-(dsl+l-xdist(ix))**2
     +                          )/3e-11))-ssd
c
                else
                        doping(ix,iy,iz)=dopds*(exp(-((-dsl+xdist(ix))**2+
     +                          (h-rj-ydist(iy))**2)/3e-11)+exp(-((dsl+
     +                          l-xdist(ix))**2+(h-rj-ydist(iy))**2)
     +                          /3e-11))-ssd
                end if
c
                end if
35              continue
25         continue
15      continue
c
        do 45 iz=1,nz
           do 55 iy=1,ny
             do 65 ix=1,nx
             n(ix,iy,iz)=doping(ix,iy,iz) +ssd +1e2
             p(ix,iy,iz)= 210e18/n(ix,iy,iz)+1e2
c       if (xdist(ix).gt.dsl.and.xdist(ix).lt.dsl+l.and.ydist(iy).gt.h-rj)then
c               n(ix,iy,iz)=doping(ix,iy,iz)+1e2
c               p(ix,iy,iz)=210e18/n(ix,iy,iz)+1e2
c       end if
65           continue
55         continue
45        continue
```

```
        return
        end
C
C
C-------------------------------------------------------------------------------
C  subroutine printr writes data to files inorder to graph on control-c system.
C-------------------------------------------------------------------------------
C
        subroutine printr(si,n,p,xdist,ydist,zdist,nx,ny,nz,doping,mun,
     +          mup,rg,temp,jtot)
        real*8 si(35,30,15),n(35,30,15),p(35,30,15),doping(35,30,15)
        real*8 mun(35,30,15)
        real*8 xdist(60),ydist(60),zdist(60),mup(35,30,15),jtot(35,30,15)
        real*8   rg(35,30,15)
C
        open(unit=1,file='xd.dat',form='formatted')
        open(unit=2,file='yd.dat',form='formatted')
        open(unit=3,file='zd.dat',form='formatted')
        open(unit=4,file='xdist.dat',form='formatted')
        open(unit=8,file='ydist.dat',form='formatted')
        open(unit=29,file='rg.dat',form='formatted')
        open(unit=21,file='mun.dat',form='formatted')
        open(unit=22,file='mup.dat',form='formatted')
        open(unit=23,file='dop.dat',form='formatted')
        open(unit=28,file='jtot.dat',form='formatted')
C
        do 30 i=1,nx
                write(4,20) xdist(i)
30      continue
        do 40 i=1,ny
                write(8,10) ydist(i)
40      continue
        write(8,*)
        do 50 j=1,ny
                do 60 i=1,nx-1
                        write(2,10) log10(p(i,j,2))
                        write(1,10)log10(n(i,j,2))
                        write(29,10)log10(abs(rg(i,j,2)/1e18+1))
                        write(3,10) si(i,j,2)
                        write(21,10) mun(i,j,2)
                        write(22,10) mup(i,j,2)
                        write(23,10) log10(abs(doping(i,j,2)))
                        write(28,10) jtot(i,j,2)
60              continue
                write(2,99) log10(p(nx,j,2))
                write(1,99)log10(n(nx,j,2))
                write(3,99) si(nx,j,2)
                write(29,99)log10(abs(rg(nx,j,2)/1e18+1))
                write(21,99) mun(nx,j,2)
                write(22,99) mup(nx,j,2)
                write(23,99) log10(abs(doping(nx,j,2)))
                write(28,99) jtot(nx,j,2)
50      continue
20      format(e20.8)
99      format(e20.8)
10      format(e20.8,$)
        return
        end
C
C------------------------------------------------------------------------------
C  subroutine mobility
C------------------------------------------------------------------------------
C
        subroutine mobility(mun,mup,doping,si,n,p,nx,ny,nz,temp,xdist,
     +          ydist,zdist)
        real*8 mun(35,30,15),mup(35,30,15),doping(35,30,15)
```

```
      real*8 si(35,30,15),n(35,30,15),p(35,30,15)
      real*8 xdist(60),ydist(60),zdist(60),ex,ey,ez,e
C
      do 10 iz=2,nz-1
         do 20 iy=2,ny-1
            do 30 ix=2,nx-1
C
C  This part of the program predicts acoustic deformation potential lattice
C  scattering.  This particular model has been given by Sah et al.
C
               xmunl=1.0/((1.0/(4195*(temp/300.0)**(-1.5)))+(1.0/(2153*
     +                (temp/300.0)**(-3.13))))
C
C  The next part of the program predicts ionized impurity scattering as well as
C  lattice scattering.
C
               xmunli=xmunl/sqrt(1.0+abs(doping(ix,iy,iz))/((3.0e16)+
     +                abs(doping(ix,iy,iz))/350.0))
C
C  This part of the program takes the gradient of the electric field.
C
               ex=(si(ix+1,iy,iz)-si(ix,iy,iz))/(xdist(ix+1)-xdist(ix))
               ey=(si(ix,iy+1,iz)-si(ix,iy,iz))/(ydist(iy+1)-ydist(iy))
               ez=(si(ix,iy,iz+1)-si(ix,iy,iz))/(zdist(iz+1)-zdist(iz))
               e=sqrt(ex**2+ey**2+ez**2)
C
C  This part takes the consideration of the carrier heating mobility.
C
               mun(ix,iy,iz)=xmunli/sqrt(1.0+abs(doping(ix,iy,iz))/(3e16+
     +                abs(doping(ix,iy,iz))/350)+(e/3.5e3)**2/(e/3.5e3+
     +                8.8)+(e/7.4e3)**2)
C
C  The next part of the program does the same as the proceeding except
C  that holes are now considered.
C
               xmupl=1.0/((1.0/(2502*(temp/300.0)**(-1.5)))+(1.0/(591*
     +                (temp/300.0)**(-3.25))))
               xmupli=xmupl/sqrt(1.0+abs(doping(ix,iy,iz))/((4e16)+
     +                abs(doping(ix,iy,iz))/81))
               mup(ix,iy,iz)=xmupli/sqrt(1.0+abs(doping(ix,iy,iz))/(4e16+
     +                abs(doping(ix,iy,iz))/81)+(e/6.1e3)**2/(e/6.1e3+1.6)+
     +                (e/2.5e4)**2)
30          continue
20       continue
10    continue
      return
      end
C
C-----------------------------------------------------------------------------
C   subroutine recgen
C-----------------------------------------------------------------------------
C
      subroutine recgen(si,n,p,nx,ny,nz,temp,xdist,ydist,zdist,rg,doping,
     +     mun,mup,jtot)
      real*8 doping(35,30,15),mun(35,30,15),mup(35,30,15)
      real*8 xdist(60),ydist(60),zdist(60),p(35,30,15),n(35,30,15)
      real*8 si(35,30,15),jtot(35,30,15)
      real*8   rg(35,30,15),xnx,yny,znz,xpx,ypy,zpz,xjnx,yjny,zjnz,xjn,xjp
      real*8   rii,alphan,alphap,xjpx,yjpy,zjpz,ee
      real*8  ex,ey,ez
C
      do 10 iz=2,nz-1
         do 20 iy=2,ny-1
              do 30 ix=2,nx-1
                 taun=(4e-4)/(1.0+abs(doping(ix,iy,iz))/7.1e15)
                 taup=(4e-5)/(1.0+abs(doping(ix,iy,iz))/7.1e15)
```

```
                    rsrh=(n(ix,iy,iz)*p(ix,iy,iz)-210e18)/(taup*(n(ix,iy,iz)+
     +                   1.45e15)+(taun*(p(ix,iy,iz)+1.45e15)))
                    rau=(2.8e-31*n(ix,iy,iz)+9.9e-32*p(ix,iy,iz))*(p(ix,iy,iz)
     +                   *n(ix,iy,iz)-210e18)
                    ex=-(si(ix+1,iy,iz)-si(ix,iy,iz))/(xdist(ix+1)-xdist(ix))
                    ey=-(si(ix,iy+1,iz)-si(ix,iy,iz))/(ydist(iy+1)-ydist(iy))
                    ez=-(si(ix,iy,iz+1)-si(ix,iy,iz))/(zdist(iz+1)-zdist(iz))
                    ee=(ex**2+ey**2+ez**2)**0.5
c
                    if(ee.gt.0) then
                         alphan=(1.0e6)*exp(0.0-(1.67e6/ee))
                         alphap=(2.0e6)*exp(0.0-(2.0e6/ee))
                    else
                         alphan=0.0
                         alphap=0.0
                    end if
c
                    xnx=(n(ix+1,iy,iz)-n(ix,iy,iz))/(xdist(ix+1)-xdist(ix))
                    yny=(n(ix,iy+1,iz)-n(ix,iy,iz))/(ydist(iy+1)-ydist(iy))
                    znz=(n(ix,iy,iz+1)-n(ix,iy,iz))/(zdist(iz+1)-zdist(iz))
                    xpx=(p(ix+1,iy,iz)-p(ix,iy,iz))/(xdist(ix+1)-xdist(ix))
                    ypy=(p(ix,iy+1,iz)-p(ix,iy,iz))/(ydist(iy+1)-ydist(iy))
                    zpz=(p(ix,iy,iz+1)-p(ix,iy,iz))/(zdist(iz+1)-zdist(iz))
                    xjnx=n(ix,iy,iz)*mun(ix,iy,iz)*ex+((0.026)*temp/300.0)*
     +                   mun(ix,iy,iz)*xnx
                    yjny=n(ix,iy,iz)*mun(ix,iy,iz)*ey+((0.026)*temp/300.0)*
     +                   mun(ix,iy,iz)*yny
                    zjnz=n(ix,iy,iz)*mun(ix,iy,iz)*ez+((0.026)*temp/300.0)*
     +                   mun(ix,iy,iz)*znz
c
                    xjpx=p(ix,iy,iz)*mup(ix,iy,iz)*ex-(0.026*temp/300.0)*
     +                   mup(ix,iy,iz)*xpx
                    yjpy=p(ix,iy,iz)*mup(ix,iy,iz)*ey-(0.026*temp/300.0)*
     +                   mup(ix,iy,iz)*ypy
                    zjpz=p(ix,iy,iz)*mup(ix,iy,iz)*ez-(0.026*temp/300.0)*
     +                   mup(ix,iy,iz)*zpz
                    xjn=(xjnx**2+yjny**2+zjnz**2)**0.5
                    xjp=(xjpx**2+yjpy**2+zjpz**2)**0.5
                    xjn=xjnx
                    xjp=xjpx
                    rii=0.0-(alphan*xjn+alphap*xjp)
                    jtot(ix,iy,iz)=(xjn+xjp)*(1e-19)
                    rg(ix,iy,iz)=(rsrh+rau+rii)*1e-7*0.0
c
30               continue
20            continue
10       continue
         return
         end
c
c-----------------------------------------------------------------------
c  subroutine boundcond calculates the boundary conditions.
c-----------------------------------------------------------------------
c
         subroutine boundcond(si,n,p,nx,ny,nz,xdist,ydist,zdist,temp,qox,l,
     +             z,dsl,vg,vd,tox,rg,ssd,mun,mup,doping,jtot)
         real*8  si(35,30,15),n(35,30,15),p(35,30,15),mun(35,30,15)
         real*8  jtot(35,30,15)
         real*8  xdist(60),ydist(60),zdist(60),mup(35,30,15),doping(35,30,15)
         real*8   rg(35,30,15),l
c
         iy=ny
         do 10 iz=1,nz
            do 20 ix=1,nx
                 mun(ix,iy,iz)=mun(ix,iy-1,iz)
                 mup(ix,iy,iz)=mup(ix,iy-1,iz)
```

```
                rg(ix,iy,iz)=rg(ix,iy-1,iz)
                jtot(ix,iy,iz)=jtot(ix,iy-1,iz)
C
41              if(xdist(ix).le.dsl) then
                   n(ix,iy,iz)=0.5*(sqrt(doping(ix,iy,iz)**2+4.0*210e18)
     +                 +doping(ix,iy,iz))
                   p(ix,iy,iz)=0.5*(sqrt(doping(ix,iy,iz)**2+4.0*210e18)
     +                 -doping(ix,iy,iz))+1.0e2
                   si(ix,iy,iz)=(1.38e-23*temp/1.6e-19)*log(doping(ix,iy,iz)/
     +                 1.45e15)
                elseif(xdist(ix).gt.dsl.and.xdist(ix).le.dsl+l) then
                   si(ix,iy,iz)=qox*1.6e-19+vg*(3.9*8.85e-14)/tox+(11.9*8.85e
     +                 -14)*si(ix,iy-1,iz)/(ydist(iy)-ydist(iy-1))
                   si(ix,iy,iz)=si(ix,iy,iz)/(((11.9*8.85e-14)/(ydist(iy)-
     +                 ydist(iy-1))+(3.9*8.85e-14)/tox))
                   vdx=vd*(1-(l+dsl-xdist(ix))/l)**2.0
                   n(ix,iy,iz)=(210e18/ssd)*exp((si(ix,iy,iz)-vdx)/.026)+1e2
                   p(ix,iy,iz)=ssd*exp(-si(ix,iy,iz)/.026)+1e2
                else
                   n(ix,iy,iz)=0.5*(sqrt(doping(ix,iy,iz)**2+4.0*210e18)+
     +                 doping(ix,iy,iz))
                   p(ix,iy,iz)=0.5*(sqrt(doping(ix,iy,iz)**2+4.0*210e18)-
     +                 doping(ix,iy,iz))+1.0e2
                   si(ix,iy,iz)=vd+(1.38e-23*temp/1.6e-19)*log(doping(ix,iy,iz)
     +                       /1.45e15)
                end if
C
20          continue
10       continue
C
         iy=1
         do 11 iz=1,nz
            do 21 ix=1,nx
                si(ix,iy,iz)=si(ix,iy+1,iz)
                n(ix,iy,iz)=n(ix,iy+1,iz)
                p(ix,iy,iz)=p(ix,iy+1,iz)
                jtot(ix,iy,iz)=jtot(ix,iy+1,iz)
                mun(ix,iy,iz)=mun(ix,iy+1,iz)
                mup(ix,iy,iz)=mup(ix,iy+1,iz)
                rg(ix,iy,iz)=rg(ix,iy+1,iz)
21          continue
11       continue
C
         ix=1
         do 12 iz=1,nz
            do 22 iy=1,ny
                si(ix,iy,iz)=si(ix+1,iy,iz)
                n(ix,iy,iz)=n(ix+1,iy,iz)
                p(ix,iy,iz)=p(ix+1,iy,iz)
                mun(ix,iy,iz)=mun(ix+1,iy,iz)
                jtot(ix,iy,iz)=jtot(ix+1,iy,iz)
                mup(ix,iy,iz)=mup(ix+1,iy,iz)
                rg(ix,iy,iz)=rg(ix+1,iy,iz)
22          continue
12       continue
C
         ix=nx
         do 13 iz=1,nz
            do 23 iy=1,ny
                si(ix,iy,iz)=si(ix-1,iy,iz)
                n(ix,iy,iz)=n(ix-1,iy,iz)
                jtot(ix,iy,iz)=jtot(ix-1,iy,iz)
                p(ix,iy,iz)=p(ix-1,iy,iz)
                mun(ix,iy,iz)=mun(ix-1,iy,iz)
                mup(ix,iy,iz)=mup(ix-1,iy,iz)
                rg(ix,iy,iz)=rg(ix-1,iy,iz)
```

```
23          continue
13       continue
C
         iz=1
         do 14 iy=1,ny
            do 24 ix=1,nx
                 si(ix,iy,iz)=si(ix,iy,iz+1)
                 n(ix,iy,iz)=n(ix,iy,iz+1)
                 p(ix,iy,iz)=p(ix,iy,iz+1)
                 jtot(ix,iy,iz)=jtot(ix,iy,iz+1)
                 mun(ix,iy,iz)=mun(ix,iy,iz+1)
                 mup(ix,iy,iz)=mup(ix,iy,iz+1)
                 rg(ix,iy,iz)=rg(ix,iy,iz+1)
24          continue
14       continue
C
         iz=nz
         do 15 iy=1,ny
            do 26 ix=1,nx
                 si(ix,iy,iz)=si(ix,iy,iz-1)
                 n(ix,iy,iz)=n(ix,iy,iz-1)
                 p(ix,iy,iz)=p(ix,iy,iz-1)
                 jtot(ix,iy,iz)=jtot(ix,iy,iz-1)
                 mun(ix,iy,iz)=mun(ix,iy,iz-1)
                 mup(ix,iy,iz)=mup(ix,iy,iz-1)
                 rg(ix,iy,iz)=rg(ix,iy,iz-1)
26          continue
15       continue
         return
         end
C
C-----------------------------------------------------------------------------
C  subroutine unscaleit unscales arrays.
C----------------------------------------------------------------------------
C
         subroutine unscaleit(si,n,p,doping,mun,mup,rg,xlambda2,temp,nx,ny,nz,
     +           xdist,ydist,zdist)
         real*8 mun(35,30,15),mup(35,30,15),si(35,30,15),doping(35,30,15)
         real*8 n(35,30,15),p(35,30,15),xdist(60),ydist(60),zdist(60),xlambda2
         real*8   rg(35,30,15),mvo
C
         rgo=36.0*1e20/((4.0e-3)**2)
         sio=temp*(1.38e-23)/1.6e-19
         mvo=36.0/sio
C
         do 10 iz=1,nz
            do 20 iy=1,ny
                 do 30 ix=1,nx
                    mun(ix,iy,iz)=mun(ix,iy,iz)*mvo
                    mup(ix,iy,iz)=mup(ix,iy,iz)*mvo
                    si(ix,iy,iz)=si(ix,iy,iz)*sio
                    n(ix,iy,iz)=n(ix,iy,iz)*1.0e20
                    p(ix,iy,iz)=p(ix,iy,iz)*1.0e20
                    doping(ix,iy,iz)=doping(ix,iy,iz)*1.0e20
                    rg(ix,iy,iz)=rg(ix,iy,iz)*rgo
30               continue
20          continue
10       continue
         do 11 ix=1,nx
         xdist(ix)=xdist(ix)*4.0e-3
11       continue
         do 12 iy=1,ny
         ydist(iy)=ydist(iy)*4.0e-3
12       continue
         do 13 iz=1,nz
         zdist(iz)=zdist(iz)*4.0e-3
```

```
13      continue
        return
        end
C
C------------------------------------------------------------------------
C  subroutine scaleit scales the arrays.
C------------------------------------------------------------------------
C
        subroutine scaleit(si,n,p,doping,mun,mup,rg,xlambda2,temp,nx,ny,nz,
     +          xdist,ydist,zdist)
        real*8  si(35,30,15),n(35,30,15),p(35,30,15),doping(35,30,15)
        real*8  mun(35,30,15),mup(35,30,15),mvo
        real*8  xdist(60),ydist(60),zdist(60)
        real*8   rg(35,30,15),xlambda2
C
        rgo=36.0*1.0e20/((4.0e-3)**2)
        sio=temp*(1.38e-23)/1.6e-19
        mvo=36.0/sio
C
        do 10 iz=1,nz
           do 20 iy=1,ny
                do 30 ix=1,nx
                   mun(ix,iy,iz)=mun(ix,iy,iz)/mvo
                   mup(ix,iy,iz)=mup(ix,iy,iz)/mvo
                   si(ix,iy,iz)=si(ix,iy,iz)/sio
                   n(ix,iy,iz)=n(ix,iy,iz)/1.0e20
                   p(ix,iy,iz)=p(ix,iy,iz)/1.0e20
                   doping(ix,iy,iz)=doping(ix,iy,iz)/1.0e20
                   rg(ix,iy,iz)=rg(ix,iy,iz)/rgo
30              continue
20         continue
10      continue
        xlambda2=(sio*11.9*8.85e-14)/(((4.0e-3)**2)*1.6e-19*1.0e20)
        do 11 ix=1,nx
        xdist(ix)=xdist(ix)/4.0e-3
11      continue
        do 12 iy=1,ny
        ydist(iy)=ydist(iy)/4.0e-3
12      continue
        do 13 iz=1,nz
        zdist(iz)=zdist(iz)/4.0e-3
13      continue
        return
        end
C
C--------------------------------------------------------------------------
C  function B returns the value of b.
C--------------------------------------------------------------------------
C
        real*8  function b(x)
C
        if (x.eq.0) then
                b=1
        else
                b=x/(exp(x)-1.0)
        end if
        return
        end
C
C-------------------------------------------------------------------------
C  function F1A returns the value of a of function F1
C--------------------------------------------------------------------------
C
        real*8  function f1a(xdist,ydist,zdist,xlambda2,ix,iy,iz)
        real*8   xdist(60),ydist(60),zdist(60),xlambda2
C
```

```
      fla=xlambda2*(ydist(iy+1)-ydist(iy-1))*(zdist(iz+1)-zdist(iz-1))
     +          /(4.0*(xdist(ix)-xdist(ix-1)))
      return
      end
C
C-------------------------------------------------------------------------
C  function f1b returns the value of b of function F1.
C-------------------------------------------------------------------------
C
      real*8  function f1b(xdist,ydist,zdist,xlambda2,ix,iy,iz)
      real*8   xdist(60),ydist(60),zdist(60),xlambda2
C
      f1b=xlambda2*(xdist(ix+1)-xdist(ix-1))*(zdist(iz+1)-zdist(iz-1))/
     +          (4.0*(ydist(iy)-ydist(iy-1)))
      return
      end
C
C-------------------------------------------------------------------------
C  function f1c returns the value of c of function F1.
C-------------------------------------------------------------------------
C
      real*8  function f1c(xdist,ydist,zdist,xlambda2,ix,iy,iz)
      real*8 xdist(60),ydist(60),zdist(60),xlambda2
C
      f1c=xlambda2*(xdist(ix+1)-xdist(ix-1))*(ydist(iy+1)-ydist(iy-1))/
     +          (4.0*(zdist(iz)-zdist(iz-1)))
      return
      end
C
C-------------------------------------------------------------------------
C  function f1d returns the value of d of function F1.
C-------------------------------------------------------------------------
C
      real*8  function f1d(xdist,ydist,zdist,xlambda2,ix,iy,iz)
      real*8    xdist(60),ydist(60),zdist(60),xlambda2
C
      f1d=-xlambda2*(((ydist(iy+1)-ydist(iy-1))*(zdist(iz+1)-zdist(iz-1))*
     +          (1.0/(xdist(ix+1)-xdist(ix))+1.0/(xdist(ix)-xdist(ix-1))))/4.0)
     +         +(xdist(ix+1)-xdist(ix-1))*(zdist(iz+1)-zdist(iz-1))*(1.0/
     +          (ydist(iy+1)-ydist(iy))+1.0/(ydist(iy)-ydist(iy-1)))/4.0+
     +          (xdist(ix+1)-xdist(ix-1))*(ydist(iy+1)-ydist(iy-1))*(1.0/
     +          (zdist(iz+1)-zdist(iz))+1.0/(zdist(iz)-zdist(iz-1))))/4.0)
      return
      end
C
C-------------------------------------------------------------------------
C  function f1e returns the value of e of function F1.
C-------------------------------------------------------------------------
C
      real*8  function f1e(xdist,ydist,zdist,xlambda2,ix,iy,iz)
      real*8 xdist(60),ydist(60),zdist(60),xlambda2
C
      f1e=xlambda2*(ydist(iy+1)-ydist(iy-1))*(zdist(iz+1)-zdist(iz-1))/
     +          (4.0*(xdist(ix+1)-xdist(ix)))
      return
      end
C
C-------------------------------------------------------------------------
C  function f1f returns the value of f of function F1.
C-------------------------------------------------------------------------
C
      real*8  function f1f(xdist,ydist,zdist,xlambda2,ix,iy,iz)
      real*8 xdist(60),ydist(60),zdist(60),xlambda2
C
      f1f=xlambda2*(xdist(ix+1)-xdist(ix-1))*(zdist(iz+1)-zdist(iz-1))/
     +          (4.0*(ydist(iy+1)-ydist(iy)))
```

```
      return
      end
c
c-----------------------------------------------------------------------
c  function f1g returns the value of g of function F1.
c-----------------------------------------------------------------------
c
      real*8  function f1g(xdist,ydist,zdist,xlambda2,ix,iy,iz)
      real*8 xdist(60),ydist(60),zdist(60),xlambda2
c
      f1g=xlambda2*(xdist(ix+1)-xdist(ix-1))*(ydist(iy+1)-ydist(iy-1))/
     +          (4.0*(zdist(iz+1)-zdist(iz)))
      return
      end
c
c-----------------------------------------------------------------------
c  function f2a returns the value of a of function F2.  It calls function b.
c-----------------------------------------------------------------------
c
      real*8  function f2a(xdist,ydist,zdist,si,n,p,ix,iy,iz,mun)
      real*8 xdist(60),ydist(60),zdist(60)
      real*8 si(35,30,15),n(35,30,15),p(35,30,15),mun(35,30,15),b
c
      f2a=((mun(ix-1,iy,iz)+mun(ix,iy,iz))/2.0)*
     +          b(si(ix-1,iy,iz)-si(ix,iy,iz))*(ydist(iy+1)-ydist(iy-1))*
     +          (zdist(iz+1)-zdist(iz-1))/(4.0*(xdist(ix)-xdist(ix-1)))
      return
      end
c
c-----------------------------------------------------------------------
c  function f2b returns the value of b of function F2.  It calls function b.
c-----------------------------------------------------------------------
c
      real*8  function f2b(xdist,ydist,zdist,si,n,p,ix,iy,iz,mun)
      real*8  xdist(60),ydist(60),zdist(60)
      real*8 si(35,30,15),n(35,30,15),p(35,30,15),mun(35,30,15),b
c
      f2b=((mun(ix,iy-1,iz)+mun(ix,iy,iz))/2.0)*
     +          b(si(ix,iy-1,iz)-si(ix,iy,iz))*(xdist(ix+1)-xdist(ix-1))*
     +          (zdist(iz+1)-zdist(iz-1))/(4.0*(ydist(iy)-ydist(iy-1)))
      return
      end
c
c-----------------------------------------------------------------------
c  function f2c returns the value of c of function F2.  It calls functions b.
c-----------------------------------------------------------------------
c
      real*8  function f2c(xdist,ydist,zdist,si,n,p,ix,iy,iz,mun)
      real*8 xdist(60),ydist(60),zdist(60)
      real*8 si(35,30,15),n(35,30,15),p(35,30,15),mun(35,30,15),b
c
      f2c=((mun(ix,iy,iz-1)+mun(ix,iy,iz))/2.0)*
     +          b(si(ix,iy,iz-1)-si(ix,iy,iz))*(xdist(ix+1)-xdist(ix-1))*
     +          (ydist(iy+1)-ydist(iy-1))/(4.0*(zdist(iz)-zdist(iz-1)))
      return
      end
c
c-----------------------------------------------------------------------
c  function f2d returns the value of d of function F2.  It calls function b.
c-----------------------------------------------------------------------
c
      real*8  function f2d(xdist,ydist,zdist,si,n,p,ix,iy,iz,mun,doping)
      real*8  xdist(60),ydist(60),zdist(60),b
      real*8 si(35,30,15),n(35,30,15),p(35,30,15),mun(35,30,15)
      real*8 doping(35,30,15)
c
```

```
      p1=((mun(ix+1,iy,iz)+mun(ix,iy,iz))/2.0)*
     +          b(si(ix,iy,iz)-si(ix+1,iy,iz))*(ydist(iy+1)-ydist(iy-1))*
     +          (zdist(iz+1)-zdist(iz-1))/(4.0*(xdist(ix+1)-xdist(ix)))
c
      p2=((mun(ix,iy+1,iz)+mun(ix,iy,iz))/2.0)*
     +          b(si(ix,iy,iz)-si(ix,iy+1,iz))*(xdist(ix+1)-xdist(ix-1))*
     +          (zdist(iz+1)-zdist(iz-1))/(4.0*(ydist(iy+1)-ydist(iy)))
c
      p3=((mun(ix,iy,iz+1)+mun(ix,iy,iz))/2.0)*
     +          b(si(ix,iy,iz)-si(ix,iy,iz+1))*(xdist(ix+1)-xdist(ix-1))*
     +          (ydist(iy+1)-ydist(iy-1))/(4.0*(zdist(iz+1)-zdist(iz)))
c
      p4=((mun(ix-1,iy,iz)+mun(ix,iy,iz))/2.0)*
     +          b(si(ix,iy,iz)-si(ix-1,iy,iz))*(ydist(iy+1)-ydist(iy-1))*
     +          (zdist(iz+1)-zdist(iz-1))/(4.0*(xdist(ix)-xdist(ix-1)))
c
      p5=((mun(ix,iy-1,iz)+mun(ix,iy,iz))/2.0)*
     +          b(si(ix,iy,iz)-si(ix,iy-1,iz))*(xdist(ix+1)-xdist(ix-1))*
     +          (zdist(iz+1)-zdist(iz-1))/(4.0*(ydist(iy)-ydist(iy-1)))
c
      p6=((mun(ix,iy,iz-1)+mun(ix,iy,iz))/2.0)*
     +          b(si(ix,iy,iz)-si(ix,iy,iz-1))*(xdist(ix+1)-xdist(ix-1))*
     +          (ydist(iy+1)-ydist(iy-1))/(4.0*(zdist(iz)-zdist(iz-1)))
c
      f2d=-(p1+p2+p3+p4+p5+p6)
      return
      end
c
c-------------------------------------------------------------------------------
c  function f2e returns the value of e of function F2.  It calls function b.
c-------------------------------------------------------------------------------
c
      real*8  function f2e(xdist,ydist,zdist,si,n,p,ix,iy,iz,mun)
      real*8 xdist(60),ydist(60),zdist(60),b
      real*8 si(35,30,15),n(35,30,15),p(35,30,15),mun(35,30,15)
c
      f2e=((mun(ix+1,iy,iz)+mun(ix,iy,iz))/2.0)*
     +          b(si(ix+1,iy,iz)-si(ix,iy,iz))*(ydist(iy+1)-ydist(iy-1))*
     +          (zdist(iz+1)-zdist(iz-1))/(4.0*(xdist(ix+1)-xdist(ix)))
      return
      end
c
c-------------------------------------------------------------------------------
c  function f2f returns the value of f of function F2.  It calls function b.
c-------------------------------------------------------------------------------
c
      real*8  function f2f(xdist,ydist,zdist,si,n,p,ix,iy,iz,mun)
      real*8 xdist(60),ydist(60),zdist(60),b
      real*8 si(35,30,15),n(35,30,15),p(35,30,15),mun(35,30,15)
c
      f2f=((mun(ix,iy+1,iz)+mun(ix,iy,iz))/2.0)*
     +          b(si(ix,iy+1,iz)-si(ix,iy,iz))*(xdist(ix+1)-xdist(ix-1))*
     +          (zdist(iz+1)-zdist(iz-1))/(4.0*(ydist(iy+1)-ydist(iy)))
      return
      end
c
c-------------------------------------------------------------------------------
c  function f2g returns the value of g of function F2.  It calls function b.
c-------------------------------------------------------------------------------
c
      real*8  function f2g(xdist,ydist,zdist,si,n,p,ix,iy,iz,mun)
      real*8 xdist(60),ydist(60),zdist(60),b
      real*8 si(35,30,15),n(35,30,15),p(35,30,15),mun(35,30,15)
c
      f2g=((mun(ix,iy,iz+1)+mun(ix,iy,iz))/2.0)*
     +          b(si(ix,iy,iz+1)-si(ix,iy,iz))*(xdist(ix+1)-xdist(ix-1))*
     +          (ydist(iy+1)-ydist(iy-1))/(4.0*(zdist(iz+1)-zdist(iz)))
      return
      end
C
C-----------------------------------------------------------------------------
C  function f3a returns the value of a of function F3.  It calls function b.
C-----------------------------------------------------------------------------
C
      real*8  function f3a(xdist,ydist,zdist,si,n,p,ix,iy,iz,mup)
      real*8 xdist(60),ydist(60),zdist(60),b
      real*8 si(35,30,15),n(35,30,15),p(35,30,15),mup(35,30,15)
C
      f3a=((mup(ix-1,iy,iz)+mup(ix,iy,iz))/2.0)*
     +          b(si(ix,iy,iz)-si(ix-1,iy,iz))*(ydist(iy+1)-ydist(iy-1))*
     +          (zdist(iz+1)-zdist(iz-1))/(4.0*(xdist(ix)-xdist(ix-1)))
      return
      end
C
C-----------------------------------------------------------------------------
C  function f3b returns the value of b of function F3.  It call function b.
C-----------------------------------------------------------------------------
C
      real*8  function f3b(xdist,ydist,zdist,si,n,p,ix,iy,iz,mup)
      real*8 xdist(60),ydist(60),zdist(60),b
      real*8 si(35,30,15),n(35,30,15),p(35,30,15),mup(35,30,15)
C
      f3b=((mup(ix,iy-1,iz)+mup(ix,iy,iz))/2.0)*
     +          b(si(ix,iy,iz)-si(ix,iy-1,iz))*(xdist(ix+1)-xdist(ix-1))*
     +          (zdist(iz+1)-zdist(iz-1))/(4.0*(ydist(iy)-ydist(iy-1)))
      return
      end
C
C-----------------------------------------------------------------------------
C  function f3c returns the value of c of function F3.  It calls function b.
C-----------------------------------------------------------------------------
C
      real*8  function f3c(xdist,ydist,zdist,si,n,p,ix,iy,iz,mup)
      real*8 xdist(60),ydist(60),zdist(60),b
      real*8 si(35,30,15),n(35,30,15),p(35,30,15),mup(35,30,15)
C
      f3c=((mup(ix,iy,iz-1)+mup(ix,iy,iz))/2.0)*
     +          b(si(ix,iy,iz)-si(ix,iy,iz-1))*(xdist(ix+1)-xdist(ix-1))*
     +          (ydist(iy+1)-ydist(iy-1))/(4.0*(zdist(iz)-zdist(iz-1)))
      return
      end
C
C-----------------------------------------------------------------------------
C  function f3d returns the value of d of function F3.  It calls function b.
C-----------------------------------------------------------------------------
C
      real*8  function f3d(xdist,ydist,zdist,si,n,p,ix,iy,iz,mup,doping)
      real*8 xdist(60),ydist(60),zdist(60),b
      real*8 si(35,30,15),n(35,30,15),p(35,30,15),mup(35,30,15)
      real*8 doping(35,30,15)
C
      p1=((mup(ix+1,iy,iz)+mup(ix,iy,iz))/2.0)*
     +          b(si(ix+1,iy,iz)-si(ix,iy,iz))*(ydist(iy+1)-ydist(iy-1))*
     +          (zdist(iz+1)-zdist(iz-1))/(4.0*(xdist(ix+1)-xdist(ix)))
C
      p2=((mup(ix,iy+1,iz)+mup(ix,iy,iz))/2.0)*
     +          b(si(ix,iy+1,iz)-si(ix,iy,iz))*(xdist(ix+1)-xdist(ix-1))*
     +          (zdist(iz+1)-zdist(iz-1))/(4.0*(ydist(iy+1)-ydist(iy)))
C
      p3=((mup(ix,iy,iz+1)+mup(ix,iy,iz))/2.0)*
     +          b(si(ix,iy,iz+1)-si(ix,iy,iz))*(xdist(ix+1)-xdist(ix-1))*
     +          (ydist(iy+1)-ydist(iy-1))/(4.0*(zdist(iz+1)-zdist(iz)))
```

```
C
        p4=((mup(ix-1,iy,iz)+mup(ix,iy,iz))/2.0)*
     +           b(si(ix-1,iy,iz)-si(ix,iy,iz))*(ydist(iy+1)-ydist(iy-1))*
     +           (zdist(iz+1)-zdist(iz-1))/(4.0*(xdist(ix)-xdist(ix-1)))
C
        p5=((mup(ix,iy-1,iz)+mup(ix,iy,iz))/2.0)*
     +           b(si(ix,iy-1,iz)-si(ix,iy,iz))*(xdist(ix+1)-xdist(ix-1))*
     +           (zdist(iz+1)-zdist(iz-1))/(4.0*(ydist(iy)-ydist(iy-1)))
C
        p6=((mup(ix,iy,iz-1)+mup(ix,iy,iz))/2.0)*
     +           b(si(ix,iy,iz-1)-si(ix,iy,iz))*(xdist(ix+1)-xdist(ix-1))*
     +           (ydist(iy+1)-ydist(iy-1))/(4.0*(zdist(iz)-zdist(iz-1)))
C
        f3d=-(p1+p2+p3+p4+p5+p6)
        return
        end
C
C------------------------------------------------------------------------------
C  function f3e returns the value of e of function F3.  It calls function b.
C------------------------------------------------------------------------------
C
        real*8  function f3e(xdist,ydist,zdist,si,n,p,ix,iy,iz,mup)
        real*8 xdist(60),ydist(60),zdist(60),b
        real*8 si(35,30,15),n(35,30,15),p(35,30,15),mup(35,30,15)
C
        f3e=((mup(ix+1,iy,iz)+mup(ix,iy,iz))/2.0)*
     +           b(si(ix,iy,iz)-si(ix+1,iy,iz))*(ydist(iy+1)-ydist(iy-1))*
     +           (zdist(iz+1)-zdist(iz-1))/(4.0*(xdist(ix+1)-xdist(ix)))
        return
        end
C
C------------------------------------------------------------------------------
C  function f3f returns the value of f of function F3.  It calls function b.
C------------------------------------------------------------------------------
C
        real*8  function f3f(xdist,ydist,zdist,si,n,p,ix,iy,iz,mup)
        real*8 xdist(60),ydist(60),zdist(60),b
        real*8 si(35,30,15),n(35,30,15),p(35,30,15),mup(35,30,15)
C
        f3f=((mup(ix,iy+1,iz)+mup(ix,iy,iz))/2.0)*
     +           b(si(ix,iy,iz)-si(ix,iy+1,iz))*(xdist(ix+1)-xdist(ix-1))*
     +           (zdist(iz+1)-zdist(iz-1))/(4.0*(ydist(iy+1)-ydist(iy)))
        return
        end
C
C------------------------------------------------------------------------------
C  function f3g returns the value of g of function F3.  It calls function b.
C------------------------------------------------------------------------------
C
        real*8  function f3g(xdist,ydist,zdist,si,n,p,ix,iy,iz,mup)
        real*8 xdist(60),ydist(60),zdist(60),b
        real*8 si(35,30,15),n(35,30,15),p(35,30,15),mup(35,30,15)
C
        f3g=((mup(ix,iy,iz+1)+mup(ix,iy,iz))/2.0)*
     +           b(si(ix,iy,iz)-si(ix,iy,iz+1))*(xdist(ix+1)-xdist(ix-1))*
     +           (ydist(iy+1)-ydist(iy-1))/(4.0*(zdist(iz+1)-zdist(iz)))
        return
        end
C
C
C------------------------------------------------------------------------------
C  subroutine soljacsi solves the jacobian for si.
C------------------------------------------------------------------------------
C
        subroutine soljacsi(nx,ny,nz,xdist,ydist,zdist,si,n,p,doping,xlambda2,
     +           mup,mun,rg,ersi)
```

```
      real*8 xdist(60),ydist(60),zdist(60)
      real*8 si(35,30,15),n(35,30,15),p(35,30,15),mun(35,30,15),mup(35,30,15)
      real*8 doping(35,30,15)
      real*8    rg(35,30,15),ersi(35,30,15)
      real*8  aparam,conres,conchn,resids(50),chngs(50),xlambda2
      real*8  wrkspl(35,30,15),wrksp2(35,30,15),wrksp3(35,30,15)
      real*8  wrksp4(35,30,15),a(35,30,15),b(35,30,15),c(35,30,15)
      real*8   d(35,30,15),e(35,30,15),f(35,30,15),g(35,30,15),q(35,30,15)
      real*8 f1a,f1b,f1c,f1d,f1e,f1f,f1g
      integer itused,ixn,iyn,izn,ifail
c
      n1 = nx
      n2 = ny
      n3 = nz
      n1m = nx
      n2m = ny
      itmax = 50
      itcoun = 0
      ndir = 1
      ixn=2
      iyn=2
      izn=2
      conres = 1.0e-1
      conchn =1.0e-4
      ifail = 0
      aparam=600.0
c
      open(unit=12,file='jacsi.dat',form='formatted')
c
      do 121  iz=1,nz
         do 122  iy=1,ny
              do 123  ix=1,nx
c
c
      if(iz.eq.1.or.iz.eq.nz.or.iy.eq.1.or.iy.eq.ny.or.ix.eq.1.or
     +        .ix.eq.nx)go to 124
         a(ix,iy,iz) = f1c(xdist,ydist,zdist,xlambda2,ix,iy,iz)
         b(ix,iy,iz) = f1b(xdist,ydist,zdist,xlambda2,ix,iy,iz)
         c(ix,iy,iz) = f1a(xdist,ydist,zdist,xlambda2,ix,iy,iz)
         d(ix,iy,iz) = f1d(xdist,ydist,zdist,xlambda2,ix,iy,iz)-(
     +                         n(ix,iy,iz)+p(ix,iy,iz))*1e-6
         e(ix,iy,iz) = f1e(xdist,ydist,zdist,xlambda2,ix,iy,iz)
         f(ix,iy,iz) = f1f(xdist,ydist,zdist,xlambda2,ix,iy,iz)
         g(ix,iy,iz) = f1g(xdist,ydist,zdist,xlambda2,ix,iy,iz)
         q(ix,iy,iz) = ((xdist(ix+1)-xdist(ix-1))*(ydist(iy+1)-
     +                          ydist(iy-1))*(zdist(iz+1)-zdist(iz-1))
     +                          /8.0)*(n(ix,iy,iz)-p(ix,iy,iz)-
     +                          doping(ix,iy,iz))-(n(ix,iy,iz)+p(
     +                          ix,iy,iz))*si(ix,iy,iz)*1e-6
c
c             write(12,80)a(ix,iy,iz),b(ix,iy,iz),
c     +         c(ix,iy,iz),d(ix,iy,iz),
c     +            e(ix,iy,iz),f(ix,iy,iz),
c     +            g(ix,iy,iz),q(ix,iy,iz)
c
              go to 123
c
124            a(ix,iy,iz)=0.0
               b(ix,iy,iz)=0.0
               c(ix,iy,iz)=0.0
               d(ix,iy,iz)=0.0
               e(ix,iy,iz)=0.0
               f(ix,iy,iz)=0.0
               g(ix,iy,iz)=0.0
               q(ix,iy,iz)=si(ix,iy,iz)
123             continue
```

```
122         continue
121      continue
C
         call d03ecf(n1,n2,n3,n1m,n2m,a,b,c,d,e,f,g,q,si,aparam,itmax,
     +           itcoun,itused,ndir,ixn,iyn,izn,conres,conchn,resids,chngs,
     +           wrksp1,wrksp2,wrksp3,wrksp4,ifail)
         write(6,*) itcoun,ifail
C
80       format(8e20.4)
70       format(e20.4)
         return
         end
C
C
C-----------------------------------------------------------------------
C  subroutine soljacn solves the jacobian for n.
C-----------------------------------------------------------------------
C
         subroutine soljacn(nx,ny,nz,xdist,ydist,zdist,si,n,p,doping,xlambda2,
     +           mup,mun,rg,ern,dopds)
         real*8 xdist(60),ydist(60),zdist(60)
         real*8 si(35,30,15),n(35,30,15),p(35,30,15),mun(35,30,15),mup(35,30,15)
         real*8 doping(35,30,15)
         real*8    rg(35,30,15),ern(35,30,15)
         real*8  aparam,conres,conchn,resids(50),chngs(50),xlambda2
         real*8  wrksp1(35,30,15),wrksp2(35,30,15),wrksp3(35,30,15)
         real*8  wrksp4(35,30,15),a(35,30,15),b(35,30,15),c(35,30,15)
         real*8   d(35,30,15),e(35,30,15),f(35,30,15),g(35,30,15),q(35,30,15)
         real*8 f2a,f2b,f2c,f2d,f2e,f2f,f2g
         integer itused,ixn,iyn,izn,ifail
C
         n1 = nx
         n2 = ny
         n3 = nz
         n1m = nx
         n2m = ny
         itmax = 50
         itcoun = 0
         ndir = 1
         ixn=22
         iyn=13
         izn=4
         conres = 1.0e-2
         conchn =1.0e0
         ifail = 0
         aparam=600.0
C
         open(unit=14,file='jacn.dat',form='formatted')
C
         do 121  iz=1,nz
            do 122  iy=1,ny
                 do 123  ix=1,nx
C
         if(iz.eq.1.or.iz.eq.nz.or.iy.eq.1.or.iy.eq.ny.or.ix.eq.1.
     +           or.ix.eq.nx)go to 124
C
            a(ix,iy,iz) =f2c(xdist,ydist,zdist,si,n,p,ix,iy,iz,mun)
            b(ix,iy,iz) =f2b(xdist,ydist,zdist,si,n,p,ix,iy,iz,mun)
            c(ix,iy,iz) =f2a(xdist,ydist,zdist,si,n,p,ix,iy,iz,mun)
            d(ix,iy,iz) = f2d(xdist,ydist,zdist,si,n,p,ix,iy,iz,mun,
     +                    doping)-rg(ix,iy,iz)*1.0
            e(ix,iy,iz) = f2e(xdist,ydist,zdist,si,n,p,ix,iy,iz,mun)
            f(ix,iy,iz) = f2f(xdist,ydist,zdist,si,n,p,ix,iy,iz,mun)
            g(ix,iy,iz) = f2g(xdist,ydist,zdist,si,n,p,ix,iy,iz,mun)
            q(ix,iy,iz) = -n(ix,iy,iz)*rg(ix,iy,iz)*1.0+rg(ix,iy,iz)
C        write(14,80) a(ix,iy,iz),b(ix,iy,iz),c(ix,iy,iz),
C     +          d(ix,iy,iz),e(ix,iy,iz),f(ix,iy,iz),
C     +          g(ix,iy,iz),q(ix,iy,iz)
C
C
        go to 123
124                a(ix,iy,iz)=0.0
                   g(ix,iy,iz)=0.0
                   b(ix,iy,iz)=0.0
                   f(ix,iy,iz)=0.0
                   c(ix,iy,iz)=0.0
                   e(ix,iy,iz)=0.0
                   d(ix,iy,iz)=0.0
                   q(ix,iy,iz)=n(ix,iy,iz)
123             continue
122        continue
121     continue
C
C
        call d03ecf(n1,n2,n3,n1m,n2m,a,b,c,d,e,f,g,q,n,aparam,itmax,
     +          itcoun,itused,ndir,ixn,iyn,izn,conres,conchn,resids,chngs,
     +          wrksp1,wrksp2,wrksp3,wrksp4,ifail)
C
        write(6,*) itcoun,ifail
        do 125 iz=2,nz-1
                do 126 iy=2,ny-1
                        do 127 ix=2,nx-1
        if(n(ix,iy,iz).le.1e-16)n(ix,iy,iz)=1e-16
        if(n(ix,iy,iz).gt.dopds/1e18)n(ix,iy,iz)=dopds/1e18
127             continue
126        continue
125     continue
80      format(8e20.4)
70      format(e20.4)
        return
        end
C
C
C-----------------------------------------------------------------------------
C  subroutine soljacp solves the jacobian for p.
C-----------------------------------------------------------------------------
C
        subroutine soljacp(nx,ny,nz,xdist,ydist,zdist,si,n,p,doping,xlambda2,
     +          mup,mun,rg,erp,ssd)
        real*8 xdist(60),ydist(60),zdist(60)
        real*8 si(35,30,15),n(35,30,15),p(35,30,15),mun(35,30,15),mup(35,30,15)
        real*8 doping(35,30,15)
        real*8    rg(35,30,15),erp(35,30,15)
        real*8  aparam,conres,conchn,resids(50),chngs(50)
        real*8  wrksp1(35,30,15),wrksp2(35,30,15),wrksp3(35,30,15),xlambda2
        real*8  wrksp4(35,30,15),a(35,30,15),b(35,30,15),c(35,30,15)
        real*8   d(35,30,15),e(35,30,15),f(35,30,15),g(35,30,15),q(35,30,15)
        real*8 f3a,f3b,f3c,f3d,f3e,f3f,f3g
        integer itused,ixn,iyn,izn,ifail
C
        open(unit=15,file='jacp.dat',form='formatted')
        n1 = nx
        n2 = ny
        n3 = nz
        n1m = nx
        n2m = ny
        itmax = 50
        itcoun = 0
        ndir = 1
        ixn=6
        iyn=5
        izn=3
```

```
      conres = 10.0e-2
      conchn =1.0e0
      ifail = 0
      aparam=600.0
c
c
      do 121  iz=1,nz
         do 122  iy=1,ny
             do 123  ix=1,nx
c
      if(iz.eq.1.or.iz.eq.nz.or.iy.eq.1.or.iy.eq.ny.or.ix.eq.1.
     +          or.ix.eq.nx)go to 124
c
         a(ix,iy,iz) = f3c(xdist,ydist,zdist,si,n,p,ix,iy,iz,mup)
         b(ix,iy,iz) = f3b(xdist,ydist,zdist,si,n,p,ix,iy,iz,mup)
         c(ix,iy,iz) = f3a(xdist,ydist,zdist,si,n,p,ix,iy,iz,mup)
         d(ix,iy,iz) = f3d(xdist,ydist,zdist,si,n,p,ix,iy,iz,mup,
     +                          doping)-rg(ix,iy,iz)*1.0
         e(ix,iy,iz) = f3e(xdist,ydist,zdist,si,n,p,ix,iy,iz,mup)
         f(ix,iy,iz) = f3f(xdist,ydist,zdist,si,n,p,ix,iy,iz,mup)
         g(ix,iy,iz) = f3g(xdist,ydist,zdist,si,n,p,ix,iy,iz,mup)
         q(ix,iy,iz) = -p(ix,iy,iz)*rg(ix,iy,iz)*1.0+rg(ix,iy,iz)
c
c        write(15,80)a(ix,iy,iz),b(ix,iy,iz),c(ix,iy,iz),d(ix,iy,iz)
c     +           ,e(ix,iy,iz),f(ix,iy,iz),g(ix,iy,iz),q(ix,iy,iz)
c
      go to 123
124               a(ix,iy,iz)=0
                  g(ix,iy,iz)=0
                  b(ix,iy,iz)=0
                  f(ix,iy,iz)=0
                  c(ix,iy,iz)=0
                  e(ix,iy,iz)=0
                  d(ix,iy,iz)=0
                  q(ix,iy,iz)=p(ix,iy,iz)
123          continue
122      continue
121   continue
c
      call d03ecf(n1,n2,n3,n1m,n2m,a,b,c,d,e,f,g,q,p,aparam,itmax,
     +          itcoun,itused,ndir,ixn,iyn,izn,conres,conchn,resids,chngs,
     +          wrksp1,wrksp2,wrksp3,wrksp4,ifail)
c
      write(6,*) itcoun,ifail
      do 125 iz=2,nz-1
         do 126 iy=2,ny-1
             do 127  ix=2,nx-1
      if(p(ix,iy,iz).le.0.0)p(ix,iy,iz)=1e-16
      if(p(ix,iy,iz).gt.ssd/1e18)p(ix,iy,iz)=ssd/1e18
127          continue
126      continue
125   continue
80    format(8e20.4)
90    format(e20.4)
      return
      end
```