A PN SEMICONDUCTOR DEVICE SIMULATOR USING MATLAB

by

Shawn Randall Pace

B.S.E.E., University of North Florida, 2007

A thesis submitted to the

University of Colorado Denver

in partial fulfillment

for the degree of

Master of Science

Electrical Engineering

2011
This thesis for the Master of Science

Electrical Engineering

degree by

Shawn Randall Pace

has been approved

by

Hamid Fardi

Gita Alaghband

Robert Grabbe

11/17/11

Date
Pace, Shawn Randall (M.S., Electrical Engineering)

A PN Semiconductor Device Simulator Using MATLAB

Thesis directed by Professor Hamid Fardi

ABSTRACT

The purpose of this project is to develop a functional PN semiconductor device simulator that is modular in nature in order to allow for flexibility during programming and to allow for future modification with relative ease. The device modeling program is developed using MATLAB™. In addition, the program’s main goal is to provide a simulator tool that can supplement device modeling, semiconductor device physics, and numerical analysis to construct basic semiconductor devices which can be studied using standard numerical analysis techniques. MATLAB’s capability and inherent nature of handling matrices and matrix operations makes this approach an excellent technique to develop numerical analysis algorithms. The program solution will be used to examine device parameters such as carrier statistics, device potential, and internal electric fields. The device solution is compared to analytical approximations in order to further strengthen the understanding between theory and exact numerical solutions and how those solutions are obtained.

This abstract accurately represents the content of the candidate’s thesis. I recommend its publication.

Signed

Hamid Fardi
ACKNOWLEDGEMENT

The main credit and acknowledgement for the development of this project goes to Dr. Hamid Fardi, without whose knowledge and support would have made the creation of this project nearly impossible. His knowledge of numerical analysis related to semiconductor physics and device physics operation as a whole was immensely appreciated and invaluable. Furthermore, appreciation should be given to the University of Colorado Denver, whose Electrical Engineering department and faculty provided the necessary training and knowledge over the course of my work there to make this project possible. It is my sincere hope that this tool can be used in a valuable training manner and supplemental learning aid.
TABLE OF CONTENTS

Figures..............................................................................................................vii

Tables................................................................................................................ix

Chapter

1. Introduction.................................................................................................1

2. Semiconductor Device Physics.................................................................4

2.1 Semiconductor Material Parameters.......................................................4

2.2 Device Equations.....................................................................................5

2.3 Carrier Densities.....................................................................................12

2.4 Mobility Models.....................................................................................13

2.5 Recombination and Generation Models..................................................15

3. Numerical Analysis....................................................................................18

3.1 Taylor Series Expansion.........................................................................18

3.2 Numerical Differentiation.......................................................................21

3.3 Numerical Integration.............................................................................23

3.4 Tri-diagonal Matrices.............................................................................25

3.5 PN Junction DC Analysis.......................................................................28

3.6 Numerical Solution of PN Junction.........................................................51

4. SDS Results Investigated.........................................................................56

4.1 SDS Output Compared to Analytical Approximations............................56
5. Semiconductor Device Simulator........................................... 67
6. Summary............................................................................... 73

Appendix

A. Integral Form of the Current Density Equation............................... 74
B. Derivation Techniques of the Matrix-vector Coefficients.................. 81
C. Semiconductor Device Simulator Implemented................................. 85

References.................................................................................. 114
LIST OF FIGURES

Figure

2.1 INFINITESIMAL SLICE FOR DETERMINATION OF INCREASE IN CARRIER DENSITY IN X.................................6
2.2 ELECTRON CONCENTRATION DIFFUSION AFFECT.................................9
3.1 INTEGRATION USING THE TRAPEZOIDAL RULE.................................23
3.2 TRI-DIAGONAL MATRIX........................................25
3.3 BASIC P-N JUNCTION DIODE........................................29
3.4 DIVISION POINTS FOR DC ANALYSIS.........................................32
3.5 MATRIX FORM OF THE MATRIX-VECTOR EQUATION.............................40
3.6 COEFFICIENT A OF MATRIX-VECTOR EQUATION.................................41
3.7 COEFFICIENT B OF MATRIX-VECTOR EQUATION.................................41
3.8 COEFFICIENT C OF MATRIX-VECTOR EQUATION.................................42
3.9 COEFFICIENT F OF MATRIX-VECTOR EQUATION.................................42
3.10 UPDATED BIAS POTENTIAL BASED ON PREVIOUS POTENTIAL..................54
4.1 ABRUPT STEP JUNCTION PROPERTIES.......................................59
4.2 EQUILIBRIUM DEVICE POTENTIAL.......................................60
4.3 EQUILIBRIUM ELECTRIC FIELD.......................................61
4.4 DEVICE POTENTIAL WITH UNEQUAL DOPING DENSITIES.................62
4.5 SIMWINDOWS COMPARISON TO FIGURE 4.4...........................62
4.6 SDS OUTPUT SHIFTED TO ZERO...........................................63
4.7 SIMWINDOWS ELECTRIC FIELD.............................................63
4.8 SDS ELECTRIC FIELD......................................................64
4.9 SDS DEVICE POTENTIAL WITH EXTERNAL APPLIED BIAS........65
4.10 SIMWINDOWS POTENTIAL WITH EXTERNAL APPLIED BIAS..........66
5.1 SDS BASIC FLOWCHART....................................................72
A.1 INTEGRAL FORM OF CURRENT DENSITY EQUATIONS.................79
LIST OF TABLES

Table

2.1  Material parameters for silicon.........................................................4
2.2  Mobility parameters........................................................................14
2.3  Ionization generation parameters.....................................................17
3.1  Partial derivatives required for matrix coefficients........................43
3.2  Numerical constants...........................................................................54
3.3  Physical constants..............................................................................55
1. **Introduction**

Although there are several powerful industry and student level device physics simulators already available, the hope is that with this new tool and supplemental report, the theory of numerical analysis applied to device physics can be studied at a more fundamental level. Traditional simulators allow for device simulation but do not offer any insight into how that solution was obtained. The addition of the report that supplements the Semiconductor Device Simulator will develop the basic tools necessary to understand the operation of the program and allow future modifications as necessary.

This project was developed in the programming environment provided by MATLAB™ which is developed and supported by Mathworks. Almost all academic institutions use MATLAB heavily in the learning process and it is readily available to most students, hence the choice of this programming environment. Additionally, MATLAB is an extremely powerful numerical analysis tool that makes the solution of a program like this one rather straightforward. MATLAB’s capability and inherent nature of handling matrices and matrix operations makes this an excellent tool to develop numerical analysis algorithms.

The current semiconductor simulation tools available do not lend to updates or modifications in a simple or straightforward manner. In fact most tools are not open to be changed in any manner unless the source code is available, which is rare. With the open nature of MATLAB the solution process can be studied throughout and all variables and parameters are available for examination. The program could have the capability to be standalone which allows for the use of the tool without the requirement of the MATLAB environment.*

The symbol "*" used throughout this manual is to indicate a future development not yet built into the current version of the program. Almost all sections in this manual include a future development section to indicate where the code can be easily updated to include new functions or models as necessary.
The remaining discussion in this section gives a brief overview of the contents included in the following sections.

Chapter 2 gives a brief introduction to the devices physics required to build a numerical analysis program to simulate semiconductor devices in general. Device physics equations (continuity, Poisson’s, current, etc.) are discussed as are the models, such as mobility, SRH (Shockley-Read-Hall) recombination, and generation. Boundary conditions are also discussed. Lastly, silicon material parameters are given and briefly discussed. Currently no other semiconductor material is used in this program, but the addition of other materials is rather straightforward*.

Chapter 3 is broken into several sections. The first section will give a brief introduction to numerical analysis techniques such as differentiation using the difference method, integration using the trapezoidal technique, Taylor series expansion, and basic tri-diagonal matrix solutions.

The next section will discuss the numerical parameters required to support the program, such as mesh spacing, iteration limits, convergence tolerances, etc. The final section will give an in-depth view of setting up and solving the numerical equations for a basic PN junction device. The section will cover converting the basic device physics equations in Chapter 2 into numerical equations, setting up the tri-diagonal block matrix, performing the recursive solution algorithm and determining convergence criteria.

Chapter 4 will provide simulation results and compare them to basic theory (depletion width, built-in potential, etc.) as well as with other Simulation tools, such as SimWindows.

Chapter 5 will discuss the program as written and the various functions built to support it. Each function will be discussed and explained. It should be noted however, that the MATLAB code itself is heavily commented as a supplement to this section.

Chapter 6 gives a brief summary of the report.
A final note on the Introduction is in order. Currently this program supports basic 1-D PN junction devices with uniform mesh spacing. Although the program is hardcoded for a Step Junction, the junction profile is easily changed just as the numerical parameters (convergence tolerance, relaxation factor, device length and doping) are without any further modification to the program. Modifications for 2-D solutions, refined mesh spacing, Shottky diodes, and BJT’s could be further implemented*. The same can be said for different numerical solution techniques if desired.
2. Semiconductor Device Physics

In this chapter, a brief discussion of simple device physics will be covered along with the differential equations governing them as well as any necessary models required to establish a numerical solution of the differential equations. Semiconductor physical parameters will be given in Section 2.1, device equations discussed in Section 2.2, carrier densities in Section 2.3, mobility models in Section 2.4 and recombination-generation models in Section 2.5.

2.1. Semiconductor Material Parameters

A tabulated list of the semiconductor physical parameters necessary to reach a numerical solution and some extra parameters used for future expansions are given below. Currently only parameters for Silicon material are used. The corresponding program variable is also shown.

Table 2.1 – Material parameters for silicon

<table>
<thead>
<tr>
<th>Nomenclature</th>
<th>Program Variable</th>
<th>Default Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative Permittivity</td>
<td>er</td>
<td>11.7</td>
<td>-</td>
</tr>
<tr>
<td>Silicon Permittivity</td>
<td>es</td>
<td>1.036 x 10-12</td>
<td>F/cm</td>
</tr>
<tr>
<td>Energy Gap (300K)</td>
<td>EG300</td>
<td>1.08</td>
<td>eV</td>
</tr>
<tr>
<td>Alpha</td>
<td>EG_alpha</td>
<td>4.73 x 10-4</td>
<td>-</td>
</tr>
<tr>
<td>Beta</td>
<td>EG_beta</td>
<td>636</td>
<td>-</td>
</tr>
<tr>
<td>Conduction Band Density (300K)</td>
<td>NC300</td>
<td>2.8 x 1019</td>
<td>cm-3</td>
</tr>
<tr>
<td>Valence Band Density (300K)</td>
<td>NV300</td>
<td>1.04 x 1019</td>
<td>cm-3</td>
</tr>
<tr>
<td>Electron Mobility</td>
<td>Function*</td>
<td>1350</td>
<td>cm2/V-sec</td>
</tr>
<tr>
<td>Hole Mobility</td>
<td>Function*</td>
<td>495</td>
<td>cm2/V-sec</td>
</tr>
<tr>
<td>Intrinsic Carrier Concentration</td>
<td>ni (Function)*</td>
<td>1.45 x 1010</td>
<td>cm-3</td>
</tr>
<tr>
<td>SRH Electron Lifetime</td>
<td>tau_n0</td>
<td>0.2 x 10-7</td>
<td>sec</td>
</tr>
<tr>
<td>SRH Hole Lifetime</td>
<td>tau_p0</td>
<td>0.2 x 10-7</td>
<td>sec</td>
</tr>
<tr>
<td>SRH Electron Concentration</td>
<td>Nsrh_n</td>
<td>5 x 1015</td>
<td>cm-3</td>
</tr>
</tbody>
</table>
Table 2.1 (Con’t.)

<table>
<thead>
<tr>
<th>Nomenclature</th>
<th>Program Variable</th>
<th>Default Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRH Hole Concentration</td>
<td>Nsrh_p</td>
<td>5 x 10^15</td>
<td>cm⁻³</td>
</tr>
<tr>
<td>Electron Affinity**</td>
<td>-</td>
<td>4.17</td>
<td>eV</td>
</tr>
<tr>
<td>Donor Energy Level**</td>
<td>-</td>
<td>0.044</td>
<td>eV</td>
</tr>
<tr>
<td>Acceptor Energy Level**</td>
<td>-</td>
<td>0.045</td>
<td>eV</td>
</tr>
<tr>
<td>Saturation Velocity**</td>
<td>Function**</td>
<td>-</td>
<td>cm/sec</td>
</tr>
<tr>
<td>Electron Auger Coef.**</td>
<td>-</td>
<td>2.8 x 10⁻³¹</td>
<td>cm⁶/sec</td>
</tr>
<tr>
<td>Hole Auger Coef.**</td>
<td>-</td>
<td>9.9 x 10⁻³²</td>
<td>cm⁶/sec</td>
</tr>
</tbody>
</table>

* Designates parameters whose values are outputs of a function but are currently set to a constant. This is also in keeping with the theme of future expansion mentioned in the introduction chapter. These terms can be further modeled with little effort.

** Parameters are not currently used.

2.2. Device Equations

Semiconductor device phenomenon is described and governed by Poisson’s equation (2.1).

\[
\frac{\partial^2 \psi}{\partial x^2} = -\frac{q}{\varepsilon} (\Gamma + p - n)
\]  

(2.1)

where, \( \Gamma = N(x) = N_d - N_a \)

and the continuity equations for electrons and holes (2.2a and 2.2b).

\[
\frac{\partial n}{\partial t} = \frac{1}{q} \frac{\partial J_n}{\partial x} + G - U
\]  

(2.2a)

\[
\frac{\partial p}{\partial t} = -\frac{1}{q} \frac{\partial J_p}{\partial x} + G - U
\]  

(2.2b)
Γ is the effective doping concentration defined for the semiconductor, whether it be a PN Junction, Bipolar Junction Transistor, MOSFET, and so on. Equation (2.1) is the differential equation for the potential distribution in an arbitrarily doped semiconductor. This equation, depending upon the particulars, is difficult to solve analytically and in some cases can be impossible. Approximations must be made in most analytical solutions, alternatively numerical methods can be used which is the focus of this report. [1] Because much of semiconductor analysis is concerned with the spatial variation of carriers and potentials from one region of a device to another, Poisson’s’ equation is almost always encountered in the solution procedure. It is one of the most useful principles in semiconductor device analysis.

Equations (2.2a) and (2.2b) are known as the continuity equations for electrons and holes respectively. These equations are used to describe the free carrier densities and their properties within the device by examining their interactions within a small infinitesimal volume that we can extend across the device as a whole. To derive a one-dimensional continuity equation for electrons, we consider an infinitesimal slice of thickness \( dx \) located at \( x \) as shown in Figure 2.1.

![Figure 2.1 - INFINITESIMAL SLICE FOR DETERMINATION OF INCREASE IN CARRIER DENSITY IN X](image)

Figure 2.1 – INFINITESIMAL SLICE FOR DETERMINATION OF INCREASE IN CARRIER DENSITY IN X
The number of electrons in the slice can increase because of the net flow into the volume and from net carrier generation in the device. The overall rate of electron increase equals the algebraic sum of

1. The number of electrons flowing into the slice, minus
2. The number flowing out, plus
3. The rate at which electrons are generated, minus
4. The rate at which they recombine. [1]

The first two components (1) and (2) are found by dividing the currents at each side of the slice by the charge on an electron, and for now, the last two components (3) and (4) are simply denoted by G and R, respectively. The rate of change in the number of electron in the slice is then given by equation (2.3).

$$\frac{\partial n}{\partial x} A dx = \left( \frac{J_n(x)}{-q} - \frac{J_n(x + dx)}{-q} \right) A + (G - R) A dx$$  \hspace{1cm} (2.3)

A is the cross-section area of the slice, q is the electron charge, and G and R is the generation and recombination densities respectively. If we use Taylor series expansion on the second term then,

$$J_n(x + dx) = J_n(x) + \frac{\partial J_n}{\partial x} dx + .... \hspace{1cm} \text{and substituting into equation (2.3) obtains the continuity equation for electrons as shown in equation (2.2b). A similar procedure will result in equation (2.2a) for the continuity equation for holes.} \hspace{1cm} \text{To yield equations that can be used analytically or numerically we need to introduce the relationship between current density } J \text{ and the carrier densities } n, \text{ and } p.

The current transport equations or current densities are defined by a combination of the drift and diffusion affects on the free carriers within the device. The drift portion of the current expression is simply due to the presence of an electric field, if there is one. This electric field creates an overall drift velocity \( v_d \), and is proportional to the electric field and the mobility of the carrier as shown in equation (2.4).

$$v_d = -\mu_n E$$  \hspace{1cm} (2.4)
The mobility of the free carrier is a parameter that describes the ease of motion of an electron in response to an electric field and is affected by several things. The mobility model and how it is used in the application of this program is discussed shortly.

The current density flowing in the direction of the electric field can be found by summing the product of the charge on each electron times its velocity over all electrons n per unit volume, or that is

\[ J_{n,\text{drift}} = \sum_{i=1}^{n} -q\nu_i = -nq\nu_d = nq\mu_n E \]

The same procedure can be extended to the hole free carriers and then combined to give the total current density due to the applied electric field in equation (2.5).

\[ J_{\text{drift}} = J_{n,\text{drift}} + J_{p,\text{drift}} = \left(q\mu_n n + q\mu_p p\right)E \quad (2.5) \]

The other affect on the current density through a device is due to the diffusion of free carriers in the device if a special variation of carrier energies or densities exist. To investigate this affect we start by examining a portion of a semiconductor device that is of n-type and with an electron density that only varies in one dimension. The temperature across the region is average so that only the density varies with x and we then consider the number of electrons passing the plane at x=0 per unit time per unit area (Figure 2.2).
Because the electrons are at finite temperature, they only have random thermal motion along one line even with no electric field applied. On average the number of electrons crossing the plane at $x=0$ from the left side start at approximately $x=-\lambda$ after a collision. The lambda term is defined as the mean-free path for an electron, or that is the average distance an electron will travel before encountering another collision and is given by $\lambda = v_{th} \tau$. Therefore the average rate per unit area of electrons crossing the plane at $x=0$ depends upon the density of electrons that started at $x=-\lambda$ and is $\frac{1}{2}N(-\lambda)v_{th}$. The factor of $\frac{1}{2}$ is due to the fact that half of the electrons move further to the left of $x=-\lambda$ after a collision and half to the right. Similarly, the rate at which electrons pass the plane $x=0$ from the right is $\frac{1}{2}N(\lambda)v_{th}$ which gives a net rate of particle flow from the left is.
\[ F = \frac{1}{2} v_{ih} \left[ n(-\lambda) - n(\lambda) \right] \] which can be approximated at \( x = \pm \lambda \) by the Taylor series expansion to give

\[ F = \frac{1}{2} v_{ih} \left\{ \left[ n(0) - \frac{dn}{dx} \lambda \right] - \left[ n(0) + \frac{dn}{dx} \lambda \right] \right\} = -v_{ih} \lambda \frac{dn}{dx} \] and finally because each electron carries charge \( q \), the particle flow corresponds to a current per unit area in equation (2.6).

\[ J_{n,\text{diff}} = -qF = q\lambda v_{ih} \frac{dn}{dx} \quad (2.6) \]

A more useful expression of equation (2.6) can be developed by using the theorem for equipartition of energy in this one-dimensional case. [1] This allows us to write

\[ \frac{1}{2} m_n^* v_{th}^2 = \frac{1}{2} kT \quad \text{and} \quad \lambda = v_{th} \tau_{en}, \] and rearrange equation (2.6) into equation (2.7).

\[ J_{n,\text{diff}} = qD_n \frac{dn}{dx} \quad \text{where,} \]

\[ D_n = \frac{kT}{q} \mu_n \quad (2.7) \]

The quantity \( D_n \) is known as the diffusion constant or Einstein relationship. It relates the two important coefficients that characterize free carrier transport by drift and by diffusion in a solid. Now with the drift and diffusion terms derived, the total electron current density can be expressed by equation (2.8)

\[ J_n = J_{n,\text{drift}} + J_{n,\text{diff}} = q\mu_n nE + qD_n \frac{dn}{dx} \quad (2.8) \]

Following the same procedure for holes leads to equation (2.9) and combing the expresses the current density for both carriers as in equation (2.10)
Before we continue to the next section, the equations discussed so far are summarized in equations (2.11a-e) below which are often referred to as the equations of state [2], and a brief discussion on non-thermal equilibrium conditions is given. The current SDS program does not employ these techniques as solution mechanisms.

Equations of State:

\[ J_p = J_{p,\text{drift}} - J_{p,\text{diff}} = q\mu_p p E - qD_p \frac{\partial p}{\partial x} \]  
(2.9)

\[ J = J_n + J_p \]  
(2.10)

Not all devices operate under thermal equilibrium conditions and so techniques are developed which allow the analysis of semiconductors that are not at thermal equilibrium to occur. We call this quasi-Fermi analysis. Through this analysis the use of quasi-Fermi potential levels are established which relate the current to the gradient of the carrier at that given quasi-Fermi level. The expressions for the quasi-Fermi carrier densities and current densities are shown below in equations (2.12-2.15).
2.3. Carrier Densities

The carrier models used in this program follow the Boltzmann statistics approximation of the Fermi-Dirac distribution model. The carrier densities are approximated in equations (2.16) and (2.17).

\[ n = n_i \exp \left( \frac{E_{fn} - E_i}{kT} \right) = n_i \exp \left( \frac{q(\psi_{fn} - \psi_i)}{kT} \right) \]  
(2.12)

\[ p = n_i \exp \left( \frac{E_i - E_{fp}}{kT} \right) = n_i \exp \left( \frac{q(\psi_{fp} - \psi_i)}{kT} \right) \]  
(2.13)

\[ J_n = \mu_n n \frac{dE_{fn}}{dx} = -q\mu_n n \frac{d\psi_{fn}}{dx} \]  
(2.14)

\[ J_p = \mu_p p \frac{dE_{fp}}{dx} = -q\mu_p p \frac{d\psi_{fp}}{dx} \]  
(2.15)

\( n_i \) is the effective intrinsic carrier concentration and has dependencies on temperature and total doping concentration. This project neglects band gap narrowing due to heavy doping levels but could be implemented as discussed in [3]*. The intrinsic carrier concentration then is only modeled as a function of temperature as shown below in equation (2.18).

\[ n = N_C \exp \left( \frac{E_{fn} - E_C}{kT} \right) = n_ie \exp \left( \frac{q(\psi - \phi_n)}{kT} \right) \]  
(2.16)

\[ p = N_V \exp \left( \frac{E_V - E_{fp}}{kT} \right) = n_ie \exp \left( \frac{q(\phi_p - \psi)}{kT} \right) \]  
(2.17)
\[n_e(T) = \sqrt{N_c(T)N_v(T)} \exp\left(\frac{E_g(T)}{2kT}\right)\] (2.18)

\[N_c \text{ and } N_v \text{ are known as the density of states and along with the band-gap energy, } E_g \text{ also have dependencies on temperature as shown in equations (2.19-2.21).}\]

\[E_g(T) = E_g(300) + \alpha \left[\frac{300^2}{300 + \beta} - \frac{T^2}{T + \beta}\right]\] (2.19)

\[N_c(T) = \left(\frac{T}{300}\right)^{\frac{3}{2}} \cdot N_c(300)\] (2.20)

\[N_v(T) = \left(\frac{T}{300}\right)^{\frac{3}{2}} \cdot N_v(300)\] (2.21)

\[N_c(300) \text{ and } N_v(300) \text{ along with } \alpha \text{ and } \beta, \text{ are predefined quantities given in section 2.1.}\]

2.4. Mobility Models

The mobility of an electron or hole to move throughout the silicon crystal lattice is a representation of the responsiveness of a free carrier to move through the conduction and valence bands in the presence of an electric field. This phenomenon is governed by several complex interactions at the quantum mechanical level. [4]

Temperature plays a major role because at higher temperatures the lattice will vibrate more causing the overall free carrier mobility to lower. Dopant levels also have a significant role since the dopant sites create local distortions in the lattice allowing the free carriers to scatter more.
The models built into SDS allow for effective doping dependency and electric field dependency as shown in equation (2.22). There is also a factor included to take into account for high-injection levels in high power devices. These models can be turned on or off to account for any dependency that is desired. The constants and fitting parameters are tabulated below.

\[
\mu = \left[ \frac{\mu_{\text{max}} - \mu_{\text{min}}}{1 + \left( \frac{N(x)}{N_{\text{ref}}} \right)^a + f} + \mu_{\text{min}} \right] \cdot \frac{1}{\left[ 1 + \left( \frac{E}{E_c} \right)^{\beta} \right]^{\frac{1}{\beta}}}
\]

where,

\[
f = \left( \frac{np \frac{1}{2}}{2.04N_{\text{ref}}} \right)^a
\]

Table 2.2 – Mobility parameters

<table>
<thead>
<tr>
<th>Nomenclature</th>
<th>Program Variable</th>
<th>Default Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electron Mobility Concentration Reference*</td>
<td>Nrefn</td>
<td>8.5 x 1016</td>
<td>cm(^{-3})</td>
</tr>
<tr>
<td>Electron Mobility Alpha*</td>
<td>alphan</td>
<td>0.72</td>
<td>-</td>
</tr>
<tr>
<td>Electron Mobility Beta*</td>
<td>betan</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>Electron Critical Electric Field*</td>
<td>Ec</td>
<td>8.0 x 103</td>
<td>V/cm</td>
</tr>
<tr>
<td>Electron Maximum Mobility*</td>
<td>mu_maxn</td>
<td>1330</td>
<td>cm(^2)/V-sec</td>
</tr>
<tr>
<td>Electron Minimum Mobility*</td>
<td>mu_minn</td>
<td>65</td>
<td>cm(^2)/V-sec</td>
</tr>
<tr>
<td>Hole Mobility Concentration Reference*</td>
<td>Nrefp</td>
<td>6.3 x 1016</td>
<td>cm(^{-3})</td>
</tr>
<tr>
<td>Hole Mobility Alpha*</td>
<td>alphap</td>
<td>0.72</td>
<td>-</td>
</tr>
<tr>
<td>Hole Mobility Beta*</td>
<td>betap</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>Hole Critical Electric Field*</td>
<td>Ec</td>
<td>1.95 x 104</td>
<td>V/cm</td>
</tr>
</tbody>
</table>
Table 2.2 (Con’t.)

<table>
<thead>
<tr>
<th>Nomenclature</th>
<th>Program Variable</th>
<th>Default Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hole Maximum Mobility*</td>
<td>mu_maxp</td>
<td>495</td>
<td>cm²/V-sec</td>
</tr>
<tr>
<td>Hole Minimum Mobility*</td>
<td>mu_minp</td>
<td>47.7</td>
<td>cm²/V-sec</td>
</tr>
</tbody>
</table>

* Constants to calculate field dependant mobility terms, when used.

2.5. Recombination and Generation Models

The SDS program models both generation and recombination for use in determining current flow in the specified device when solving both the continuity equations and Poisson’s equation. The most fundamental type of recombination defined as Shockley-Read Hall recombination. The recombination rates are defined in equation (2.23).

\[
U = U_n = U_p = \frac{np - n_i^2}{\tau_p(n + n_0) + \tau_n(p + p_0)}
\]  

(2.23)

where,

\[
n_0 = n_i \exp \left( \frac{E_i - E_f}{kT} \right) \quad \text{and} \quad p_0 = n_i \exp \left( \frac{E_f - E_i}{kT} \right)
\]

The electron and hole lifetime are defined by \( \tau_n \) and \( \tau_p \), respectively and are defined below in equation (2.24) where \( \sigma \) is the capture cross-section and \( N_t \), the density of trapping centers.

\[
\tau_n = \frac{1}{\sigma_n v_{th} N_t} \quad \text{and} \quad \tau_p = \frac{1}{\sigma_p v_{th} N_t}
\]

(2.24)
Another method to calculate the lifetime as used in this program is from [3] and is shown in equation (2.25) below.

\[
\tau_n = \frac{\tau_{n0}}{1 + \frac{N_{\text{total}}}{N_{\text{SRH}},n}} \quad \text{and} \quad \tau_p = \frac{\tau_{p0}}{1 + \frac{N_{\text{total}}}{N_{\text{SRH}},p}}
\]

(2.25)

The constants are tabulated below. Another type of recombination is Auger recombination, characterized by electron-hole pairs disappearing without the aid of trapping centers as in SRH recombination, and is mainly used when there are large high-injection rates. Auger recombination is defined by equation (2.27). This model and the SRH model are added together when used in conjunction to provide the overall recombination rate as shown in equation (2.26). To use the Auger model, the partial derivative terms used in Chapter 3 - Numerical Analysis will need significant modification since Auger recombination is a function of both carrier densities.

\[
U = U_{\text{SRH}} + U_{\text{AUG}}
\]

(2.26)

where,

\[
U_{\text{AUG}} = r \left( n^2 p - p^2 n \right)
\]

(2.27)

The generation model used in this program models the basic generation affect due to impact ionization as defined in equation (2.28). It is useful in characterizing reverse applied bias breakdown. In low electric field applied bias cases the generation rates are typically negligible. [4] The constants and fitting parameters are tabulated below.

\[
G = G_n = G_p = \frac{1}{q} \left( \alpha_n |J_n| + \alpha_p |J_p| \right)
\]

(2.28)

where,

\[
\alpha_n = \alpha_{n0} \exp\left( -\frac{E_{n0}}{|E|} \right) \quad \text{and} \quad \alpha_p = \alpha_{p0} \exp\left( -\frac{E_{p0}}{|E|} \right)
\]
Table 2.3 – Ionization generation parameters

<table>
<thead>
<tr>
<th>Nomenclature</th>
<th>Program Variable</th>
<th>Default Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electron Alpha</td>
<td>alpha_n0</td>
<td>2.25 x 10^7</td>
<td></td>
</tr>
<tr>
<td>Hole Alpha</td>
<td>alpha_p0</td>
<td>3.08 x 10^6</td>
<td></td>
</tr>
<tr>
<td>Electron Energy</td>
<td>E_n0</td>
<td>3.26 x 10^6</td>
<td>V/cm</td>
</tr>
<tr>
<td>Hole Energy</td>
<td>E_p0</td>
<td>1.75 x 10^6</td>
<td>V/cm</td>
</tr>
<tr>
<td>Fitting Parameter</td>
<td>m</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>
3. Numerical Analysis

In this chapter, we will proceed in setting up the numerical equations in order to solve the physical device equations established in the previous chapter. A procedure for setting up the equations in numerical form, linearizing those equations, and solving the subsequent block tri-diagonal matrix equation will be given in a step by step manner.

However, first a brief introduction into some of the basic numerical analysis techniques required to solve the device equations will be given below. All of these techniques are used in transforming the differential device equations in to linearized numerical equations suitable for a numerical analysis solution.

3.1. Taylor Series Expansion

The Taylor series is named after the English mathematician Brook Taylor (1685-1731) and the Maclaurin series is named in honor of the Scottish mathematician Colin Maclaurin (1698-1746) despite the fact that the Maclaurin series is just a special type of Taylor series. But the idea of representing particular functions as sums of power series goes back to Newton, and the general Taylor series was known to Scottish mathematician James Gregory in 1668 and to the Swiss mathematician John Bernoulli in the 1660s. Taylor was apparently unaware of the work of Gregory and Bernoulli when he published his discoveries on series in 1715 in his book *Methodus incrementorum directa et inversa*. Maclaurin series are named after Colin Maclaurin because he popularized them in his calculus textbook *Treatise of Fluxions* published in 1742. [5]
As we have seen in Chapter 2 – Semiconductor Device Physics, the basic equations describing the physics of semiconductors are differential in nature and nonlinear. In order to perform numerical analysis on these equations, they must be linearized and in order to do this, the Taylor series expansion is performed about a point of interest. Additionally, for our equations of interest, the linearization must be done in each variable of interest (n, p, and ψ) and this can be seen later in section 3.5.

The general form of the Taylor series is shown in equation (3.1) below and is referred to as the Taylor series expansion of the function f at a (or about a, or centered at a). [5]

\[ f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + \frac{f'(a)}{1!}(x-a) \]
\[ + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + ... \]

For the special case of a = 0, the Taylor series becomes equation (3.2) and is given the special name Maclaurin series. In our analysis, we will be using the standard Taylor series expansion as the expansion point is not always zero.

\[ f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!}(x)^n = f(0) + \frac{f'(0)}{1!}(x) + \frac{f''(0)}{2!}(x)^2 + \frac{f'''(0)}{3!}(x)^3 + ... \] (3.2)

Since we are dealing with a system of nonlinear equations in which we are interested in multiple variables (n, p, and ψ), each equation that is linearized must be done so around each variable and so therefore the partial derivatives of each variable is introduced. Extending the above discussion on series expansion to a multiple variables is rather straightforward. We will also introduce the concept of perturbation state (or delta state).

Let us re-examine equation (3.1) and transform it into the form shown in equation (3.3) below. We assume that ‘a’ is a point such that f(a) = 0 and this point is defined as the equilibrium point of the system f'(x) = f(x) since f'(x) = 0, when x = a. Since f(a) = 0, the nonlinear differential equation can be approximated near the equilibrium point by equation (3.4).
\[ f(x) = f(a) + \frac{df}{dx}_{x=a} (x-a) + \text{higher order terms} \]  \hspace{1cm} (3.3)

\[ f'(x) = \frac{df}{dx}_{x=a} (x-a) \]  \hspace{1cm} (3.4)

When the expansion point, in our case ‘a’, is sufficiently close to \( x \), the higher order terms in equation (3.3) can be neglected as they are very close to zero. To complete the linearization, we define the perturbation state, or delta state \( \delta x = x-a \) and using the fact that \( \delta f'(x) = f'(x) \) we obtain the linearized model show in equation (3.5).

\[ \delta f'(x) = \frac{df}{dx}_{x=a} \delta x \]  \hspace{1cm} (3.5)

This can be easily applied to multi-variable systems and is show in equation (3.6). This is the form employed in section 3.5 to linearize the system.

\[ \delta f_i'(x, y) = \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}_{x_j=a_j} \delta x_j + \sum_{j=1}^{m} \frac{\partial f_i}{\partial y_j}_{y_j=a_j} \delta y_j \]  \hspace{1cm} (3.6)

In summary, this technique allows us to linearize a system of nonlinear equations about a point of expansion. In this analysis, the concept is a more generic approach. As will be seen later, we linearize the current equations and generation and recombination equations as they involve nonlinear terms of an exponential nature. Since the three equations of interest are the two current continuity equations and Poisson’s equation, and since the current continuity equations are differential expressions of the current equations, these must also be linearized. This allows the linearized current and recombination/generation equations to be substituted in directly. This will be more closely examined in section 3.5.
3.2. Numerical Differentiation

It is often necessary, especially in the procedures following, to estimate the derivative of a function and is one of the fundamental requirements in performing finite difference method techniques. The procedure used to perform this is called the difference method and is applicable to first or second order derivatives. The procedure can be used to extend to higher order derivatives but is not discussed here. Additionally, more elegant difference techniques can be used in order to lower the inherent numerical error associated with calculating the derivative of a function but is also not discussed here.

Numerical differentiation is performed using a series expansion and in this instance a Taylor’s series expansion is also used. If we assume that \( f(x) \) has as many continuous derivatives as necessary, then from Taylor’s formula, equation (3.7) is found, which is the same form as equation (3.1) but with the expansion about ‘\( x \)’, instead of ‘\( a \)’, which was arbitrary for the above discussion. [6]

\[
f(x + h) = f(x) + \frac{h}{1!} f'(x) + \frac{h^2}{2!} f''(x) + \frac{h^3}{3!} f'''(x) + \ldots
\] (3.7)

One way to think of equation (3.7) is that we are expanding the function \( f(x) \) about \( x \), within small increments of \( h \). So if we replace the arbitrary ‘\( a \)’ in equation (3.1) with ‘\( x+h \)’, equation (3.7) is a straightforward application. If we solve equation (3.7) for \( f'(x) \), the equation can be written as

\[
f'(x) = \frac{f(x + h) - f(x)}{h} - \frac{h}{2!} f''(x) - \frac{h^2}{3!} f'''(x) + \ldots
\]

and neglecting higher order terms gives equation (3.8),

\[
f'(x) = \frac{f(x + h) - f(x)}{h}, \quad \text{for small value of } h.
\] (3.8)
Equation (3.8) is called the forward divided difference approximation and graphically is the representation of the slope of the line passing through points \((x, f(x))\) and \((x+h, f(x+h))\). There are two other main approximations for a first order derivative and they are directly obtained from Taylor's series expansion as well.

Assume we replace 'h' with '-h' in equation (3.7) and repeat the same process we obtain equation (3.9) below. Equation (3.9) is referred to as the backward divided difference approximation.

\[
f'(x) = \frac{f(x) - f(x-h)}{h}, \text{ for small value of } h. \tag{3.9}
\]

Finally, if we replace 'h' with '-h' in equation (3.7) and then subtract the resulting equation from the old one, we obtain

\[
f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{3!} f^{(2)}(x) - \frac{h^4}{5!} f^{(4)}(x) - ... 
\]

and neglecting higher order terms gives,

\[
f'(x) = \frac{f(x+h) - f(x-h)}{2h}, \text{ for small value of } h. \tag{3.10}
\]

Equation (3.10) is known as the central difference approximation. Each numerical technique to establish the derivative of a function lends to round-off or truncation errors as we are approximating the value. Keeping higher order terms of the expansion will lower this error but is not necessary for this application.
3.3. Numerical Integration

Just as with derivatives, calculating the integrals of functions numerically is also of interest for this application. The main use for this technique, recalling the device continuity equations from Chapter 2, is to calculate the current density across the device using the recombination and generation terms that have previously been calculated. Since mesh spacing is also defined, use of a simple trapezoidal integration rule is rather straightforward.

Let us suppose we would like to evaluate the definite integral of the form shown in equation (3.11) and which is graphically represented in Figure 3.1.

\[ I = \int_a^b f(x)dx \]  

(3.11)

Figure 3.1 – INTEGRATION USING THE TRAPEZOIDAL RULE
To calculate the integral (area under the curve), we begin by splitting up the interval \([a, b]\) into \(n\) subintervals, each with uniform mesh spacing as shown in Figure 3.1. This gives equation (3.12).

\[
h = \frac{b-a}{n} \quad \text{and} \quad x_i = a + ih \quad \text{for} \quad i = 0,1,2,...,n
\]  

(3.12)

Now by examining each subinterval, the area under the curve can be approximated by the sum of the areas of the trapezoids. The area of the \(i\)th trapezoid is therefore

\[
A_i = \frac{h}{2} [f(x_{i-1}) + f(x_i)]
\]

with the sum or total area \(T_n\) being obtained by extending over the entire area.

\[
T_n = A_1 + A_2 + ... + A_{n-1} + A_n
\]

\[
= \frac{h}{2} [f(x_0) + f(x_1)] + \frac{h}{2} [f(x_1) + f(x_2)] + ... \\
+ \frac{h}{2} [f(x_{n-2}) + f(x_{n-1})] + \frac{h}{2} [f(x_{n-1}) + f(x_n)]
\]

Finally, by collecting like terms we arrive at equation (3.13).

\[
T_n = \frac{h}{2} [f(x_0) + f(x_n)] + h \sum_{i=1}^{n-1} f(x_i)
\]  

(3.13)

Equation (3.13) is referred to as the composite trapezoid rule as the integral is estimated by a composite of the areas of \(n\) trapezoids. [6]
3.4. Tri-diagonal Matrices

This section will briefly discuss matrices with respect to diagonal dominate matrices and especially tri-diagonal matrices. A tri-diagonal matrix, as shown in Figure 3.2, is simply a matrix which has non-zero entities on the main diagonal, super-diagonal (upper), and sub-diagonal (lower) and frequently arises in the solution of numerical differential equations, such as in this project.

\[
A = \begin{bmatrix}
  a_{11} & a_{12} & 0 & 0 & 0 \\
  a_{21} & a_{22} & a_{23} & 0 & 0 \\
  0 & a_{32} & a_{33} & a_{34} & 0 \\
  0 & 0 & a_{43} & a_{44} & a_{45} \\
  0 & 0 & 0 & a_{54} & a_{55}
\end{bmatrix}
\]

Figure 3.2 – TRI-DIAGONAL MATRIX

The main content of this section will discuss a recursive method for solving a block tri-diagonal matrix equation as this is what is employed in the solution of this project. The algorithm is borrowed from [4] directly but is a standard algorithm used frequently. There are many methods to solve these types of matrices, from the very basic naïve Gaussian elimination, to more complex solutions, like LU decomposition, Jacobi method, etc.

As we will see in Section 3.5 below, for a 1-D solution, a block tri-diagonal matrix solution must be obtained. The recursive algorithm for this procedure is as follows.

Assume the matrix equation that needs to be solved is of the form in equation (3.14).

\[
A(N)\delta y(N-1) + B(N)\delta y(N) + C(N)\delta y(N+1) = F(N), \quad 2 \leq N \leq L-1
\]

for \( \delta y(N) = [\delta p(N), \delta t(N), \delta \nu(N)]^T \), \( 2 \leq N \leq L-1 \)
where matrices A, B, and C are 3x3 dimension and vector F is of 3x1 dimension. Since A, B, and C are 3x3 matrices we have a block tri-diagonal matrix as opposed to a simple tri-diagonal matrix. A direct solution would be to construct this matrix per equation (3.14) and use any of the aforementioned solution algorithms but this is computationally expensive. It would be much more efficient to only have to calculate the matrix blocks and solve for the delta state (or error value) and not have to manipulate every element in the matrix, including the majority of empty or zero elements. The use of the recursive algorithm allows for this computation savings. To continue, equation (3.14) is transformed into the following equation (3.15) involving unknown vectors at two points, N and N+1, instead of three.

\[
B(N)\delta y(N) + C(N)\delta y(N+1) = F'(N)
\]  

(3.15)

Equation (3.15) can be solved for \(\delta y(N)\) explicitly using basic matrix math, which yields equation (3.16) below.

\[
\delta y(N) = B(N)F'(N) - B'(N)C(N)\delta y(N+1)
\]  

(3.16)

If we then take equation (3.16) and reduce the division point N by one, we can substitute this result back into equation (3.14), rearranging the terms to arrive at equation (3.17). The following steps illustrate this procedure.

\[
\delta y(N-1) = B'(N-1)F'(N-1) - B'(N-1)C'(N-1)\delta y(N)
\]  

(division point N reduced by one)

Substituting back into equation (3.14) yields,

\[
A(N)B'(N-1)F'(N-1) - A(N)B'(N-1)C'(N-1)\delta y(N) + B(N)\delta y(N) + C(N)\delta y(N+1) = F(N)
\]

Rearranging terms yields,

\[
\left[ B(N) - A(N)B'(N-1)C'(N-1) \right] \delta y(N) + C(N)\delta y(N+1) = F(N) - A(N)B'(N-1)F'(N-1)
\]  

(3.17)
If we now compare equation (3.17) with equation (3.15) we obtain the recursive relations given below in equation (3.18), but for one less division point calculation.

\[
B' (N) = B(N) - A(N)B' (N - 1)C' (N - 1) \quad C' (N) = C(N) \quad \text{and}
\]
\[
F' (N) = F(N) - A(N)B' (N - 1)F' (N - 1), \quad 3 \leq N \leq L - 1
\]

To finalize the algorithm since we would like to solve for \( \delta y(N) \), the error values, we compare equation (3.18) with equation (3.14) for \( N=2 \) specifically and obtain equation (3.19).

\[
B' (2) = B(2), \quad C' (2) = C(2), \quad F' (2) = F(2)
\]

We know that \( \delta y(N-1) \) for \( N=2 \) is \( \delta y(1) \), which is equal to zero. \( \delta y(1) \) corresponds to the boundary condition error, but the boundary condition is fixed and so there is no error associated with that term and therefore reduces to zero. Finally, starting with equation (3.19) at \( N=2 \), then using equation (3.18) for the rest of the division points (\( N=3, 4, ..., L-1 \)) we can use equation (3.16) to calculate the error values, \( \delta y(N) \), across the whole device (\( N=L-1, L-2, ..., 2 \)) starting with \( \delta y(L)=0 \), since this is also a fixed boundary condition.

At this point, matrices A, B, C, and F are arbitrary with the dimensions defined above. As we will see in the next section, these matrices will be defined and so calculation of the above algorithm becomes very straightforward. Once A, B, C, and F are calculated across the device using the recursive relations given, the error potential is solved for directly. The definition of these matrices comes from performing the finite element methods (difference method) on the device equations and then linearizing those equations.
3.5. P-N Junction DC Analysis

In this section, a procedure for performing DC analysis of a 1-D, P-N junction device will be developed. The basic P-N junction diode is examined as further development techniques for other devices follow the same basic procedures and solutions used in this section. For a 1-D Bipolar Junction Transistor, for example, only slight modification of this procedure is required and the resulting simulator program can be modified to use either solution depending upon the device specified. Another example of modification is time domain analysis, which requires extra division spacing but ultimately requires little modification to the base program with the conceptual idea that these further implementations can be readily incorporated at a later date.*

Equations (3.20a-e) and Figure 3.1 show the basic setup for a 1-D P-N junction diode. The overall doping profile as of yet is undefined but could be a step junction, linear junction, and so on.

\[
\begin{align*}
J_p &= -qD_p \frac{\partial p}{\partial x} - q\mu_p p \frac{\partial \psi}{\partial x}, \quad (3.20a) \\
J_n &= qD_n \frac{\partial n}{\partial x} - q\mu_n n \frac{\partial \psi}{\partial x}, \quad (3.20b) \\
\frac{\partial p}{\partial t} &= -\frac{1}{q} \frac{\partial J_p}{\partial x} + G_p - U_p, \quad (3.20c) \\
\frac{\partial n}{\partial t} &= -\frac{1}{q} \frac{\partial J_n}{\partial x} + G_n - U_n, \quad (3.20d) \\
\frac{\partial^2 \psi}{\partial x^2} &= -\frac{q}{\varepsilon} (\Gamma + p-n), \quad \text{where} \quad \Gamma = N_D - N_A \quad (3.20e)
\end{align*}
\]

For DC analysis the time derivatives in equations (3.20c and 3.20d) will become zero as the device is considered to be in steady state.
Before we can proceed with numerical analysis of the device shown in Figure 2, a few points of interest should be noted. Many of the models used in the solution procedure are of the basic type, however can be expanded easily to incorporate future models. For example, the intrinsic carrier concentration model is currently only temperature dependant and the mobility model is fixed across the device. Both could be modified to incorporate band gap narrowing for heavily doped devices or electric field dependency respectively.* More discussion on each model used in this section along with its capabilities is further discussed in Chapter 5 – Semiconductor Device Simulator and also in Chapter 2.

The recombination model employed in this solution is of the basic Shockley-Read Hall type. It is the most fundamental type of recombination method and so therefore is most applicable to basic PN junction devices. Equations 18C and d imply that the recombination and generation terms are the net terms and can constitute different models, such as SRH and Auger recombination. In this case, the net recombination rate from both models is used. Furthermore, the recombination centers only occupy a single energy level directly in the middle of the forbidden energy band. This implies that the equilibrium hole concentration is equal to the equilibrium electron concentration which is in turn equal to the intrinsic carrier concentration, n_i = p_i = n_i. For higher power devices, Auger recombination, as discussed in Chapter 2, can also be considered but is not explicitly used in this solution procedure. This allows us to use the basic SRH recombination equation from Chapter 2, repeated as shown here in equation (21).
If we only want to evaluate the device in low electric field applications technically the generation terms are not necessary and are not supported in some programs, such as in [3]. However, this program includes the generation terms and as with recombination we assume that the net hole generation terms are equal to the net electron generation terms. These terms are useful in supporting devices with extremely high or reverse applied biases. We can examine the process of avalanche breakdown by incorporating these terms, however when low applied biases are used, these terms are essentially zero. See more discussion in Chapter 5 on the generation function. Repeating the equations from Chapter 2 for the one dimensional case gives us equation (3.22) below.

\[ G = G_p = G_n = \frac{1}{q} (\alpha_p J_p + \alpha_n J_n), \text{ where} \]

\[ \alpha_p = \alpha_{p0} \exp \left( \frac{E_{p0}}{|E|} \right), \quad \alpha_n = \alpha_{n0} \exp \left( \frac{E_{n0}}{|E|} \right), \text{ and} \]

\[ E = -\frac{\partial \psi}{\partial x} \]
The boundary conditions need to satisfy a zero space charge region at each contact. Usually the semiconductor forming an ohmic contact with the metal has sufficiently high doping concentration. If there were a nonzero space charge region in such a highly doped semiconductor then there would be a correspondingly high electric field which would lead to breakdown conditions at the metal-semiconductor contact, so therefore the zero space charge condition must be satisfied. Future work can include the Shottky boundary conditions necessary to satisfy Shottky devices.*

\[
\begin{align*}
\Gamma(0) + p(0) - n(0) &= 0, \quad \Gamma(w) + p(w) - n(w) = 0, \\
p(0)n(0) &= n_i^2, \quad p(w)n(w) = n_i^2, \\
\psi(0) &= V - \frac{1}{\theta} \ln \left[ \frac{p(0)}{n_i} \right], \quad \psi(w) = V - \frac{1}{\theta} \ln \left[ \frac{n(w)}{n_i} \right], \quad \text{where,} \quad \theta = \frac{q}{kT}
\end{align*}
\] (3.23)

Equation (3.23a) satisfies the zero space charge condition specified above, while equation (3.23b) satisfies the thermal equilibrium condition necessary at each contact. Equations (3.23a and 3.23b) are used to establish the two-point boundary conditions that are self-consistent among the two equations and the result is shown in equation (3.24).

\[
\begin{align*}
p(0) &= -\frac{\Gamma(0)}{2} \left[ 1 + \left[ 1 + \left( \frac{2n_i}{\Gamma(0)} \right)^2 \right]^{1/2} \right], \quad n(0) = \frac{n_i^2}{p(0)} \\
n(w) &= -\frac{\Gamma(w)}{2} \left[ 1 + \left[ 1 + \left( \frac{2n_i}{\Gamma(w)} \right)^2 \right]^{1/2} \right], \quad p(w) = \frac{n_i^2}{n(w)}
\end{align*}
\] (3.24)

Conditions for the potential boundary conditions given in equation (3.23c) are deduced by the use of the quasi-Fermi potentials briefly discussed in Chapter 2. These are re-written here in equation (3.25) for reference.

\[
\begin{align*}
\psi_p &= \phi_p - \frac{1}{\theta} \ln \left[ \frac{p}{n_i} \right], \quad \psi_n = \phi_n - \frac{1}{\theta} \ln \left[ \frac{n}{n_i} \right], \quad \text{where,} \quad \theta = \frac{q}{kT}
\end{align*}
\] (3.25)
If the definition of the zero potential point on the n-region and applied bias on the p-region is extended to equation (3.25) it follows that $\varphi_p(w) = \varphi_n(w) = 0$ and $\varphi_p(0) = \varphi_n(0) = V$. Substituting these into equation (3.25) yields equation (3.23c) directly. With equation (3.23c) and equation (3.24) we now have a two-point boundary condition with respect to the three unknown variables, n, p, and $\psi$ for use in the solution of the device.

We can now proceed with the numerical analysis of the device equations but in order to do that, we need to transform the differential equations into difference equations with the variables defined at a finite number of division points (mesh points) as shown in Figure 3. Main division points are defined by $N$, with $N=1$ and $N=w$ corresponding to the device endpoints $x=0$ and $x=w$ respectively. Auxiliary points are defined such that they are exactly half the way between main division points.

This solution uses uniform mesh spacing and is sufficient for most classes of devices, however when the extent of the depletion region becomes much smaller than the neutral bulk regions it is sometimes more beneficial to have coarse mesh spacing in the neutral regions and finer mesh spacing in the depletion region. The SDS program can be easily modified to incorporate non-uniform mesh spacing.

![Figure 3.4 - Division Points for DC Analysis](image)

In the following analysis, variables n, p, and $\psi$ are estimated on the main points, while the derivatives of these quantities are estimated on auxiliary points. The device equations in (3.20a-e), which need to be transformed into difference equations, are repeated here in equation (3.26a-e) neglecting the time derivative. [4]

$$J_p = -qD_p \frac{\partial p}{\partial x} - q\mu_p p \frac{\partial \psi}{\partial x},$$  
(3.26a)
The equations are now transformed into the difference equations. A brief discussion on equations (3.27a and 3.27b) will follow shortly (including the coefficients for the lambda terms and how they are obtained, but equations (3.27c-e) are obtained directly by using the difference methods employing the techniques discussed in Section 3.2 – Numerical Differentiation.

\[
J_p(M) = \frac{q}{h(M)} \left[ \lambda_{p1}(M)p(N) + \lambda_{p2}(M)p(N + 1) \right] \tag{3.27a}
\]

\[
J_n(M) = \frac{q}{h(M)} \left[ \lambda_{n1}(M)n(N) + \lambda_{n2}(M)n(N + 1) \right] \tag{3.27b}
\]

where,

\[
\lambda_{p1}(M) = \mu_p(M) \frac{\psi(N) - \psi(N + 1)}{1 - e^{-\beta(M)}}, \quad \lambda_{n1}(M) = \mu_n(M) \frac{\psi(N) - \psi(N + 1)}{1 - e^{-\beta(M)}}
\]

\[
\lambda_{p2}(M) = \mu_p(M) \frac{\psi(N) - \psi(N + 1)}{1 - e^{-\beta(M)}}, \quad \lambda_{n2}(M) = \mu_n(M) \frac{\psi(N) - \psi(N + 1)}{1 - e^{-\beta(M)}}
\]

\[
\beta(M) = \theta E(M) h(M) = \theta \left[ \psi(N) - \psi(N + 1) \right]
\]

\[
1 \cdot \frac{J_p(M) - J_p(M - 1)}{q \cdot h'(N)} - G(N) + U(N) = 0 \tag{3.27c}
\]

\[
1 \cdot \frac{J_n(M) - J_n(M - 1)}{q \cdot h'(N)} + G(N) - U(N) = 0 \tag{3.27d}
\]
\[ \gamma_1 \psi(N-1) + \gamma_2 \psi(N) + \gamma_3 \psi(N+1) = -\frac{q}{\epsilon} \left[ \Gamma(N) + p(N) - n(N) \right] \] (3.27e)

As you can see in the continuity equations (3.26c and 3.26d), the derivative of the current density with respect to the spatial variable, \( x \), has been transformed into the difference approximation in equations (3.27c and 3.27d). Also Poisson’s equation in (3.26e) was transformed directly to equation (3.27e) also by using the difference method. This time, the second order central difference method was used since Poisson’s equation is a second order differential equation. The transformation of equation (3.26e) into difference form is shown below in equation (3.28).

\[ \frac{1}{h(N)} \left[ \frac{\psi(N+1) - \psi(N)}{h(M)} - \frac{\psi(N) - \psi(N-1)}{h(M-1)} \right] = -\frac{q}{\epsilon} \left[ \Gamma(N) + p(N) - n(N) \right] \] (3.28)

Throughout the solution process, the mesh spacing is continually defined at each division point, whether it is a main division point or auxiliary point. Since we are using uniform mesh spacing and the auxiliary spacing is equal to the main spacing, equation (3.28) could be further transformed into the standard looking central difference equation for a second order derivative. Mesh spacing is left as a function of the division point so that future expansion of non-uniform mesh spacing is rather straightforward*. Lastly, if we examine equation (3.28) with equation (3.27e) the gamma coefficients are obtained directly.

\[ \gamma_1(N) = \frac{1}{h(M-1)h(N)}, \quad \gamma_2(N) = \frac{1}{h(N)} \left[ \frac{1}{h(M-1)} + \frac{1}{h(M)} \right], \quad \gamma_3(N) = \frac{1}{h(M)h(N)} \]

At this point in the solution process we have all of the equations transformed into numerical difference equations. The current density equations can be plugged into the current continuity equations and combined with Poisson’s equation; the three equations can be then solved simultaneously. However, the current density equations, as well as the generation and recombination terms, are not linear and so the application of Newton’s iteration principle cannot yet be used. The next step in the solution process is to linearize these equations about the fundamental variables of interest, \( n, p, \) and \( \psi \).
However, before we proceed to linearize the equations let us return to equations (3.27a and 3.27b), where if we compare these to equations (3.26a and 3.26b) it is obvious they were not directly obtained from the difference technique. Before continuing, please proceed to Appendix A, where a complete derivation and explanation of equations (3.27a and 3.27b) is given.

Because the continuity equations contain expressions for the current densities which themselves contain derivatives of the fundamental variables, $n$, $p$, and $\psi$, the derivatives for the current densities are defined at main points as well. Therefore, it is necessary to define the spacing between the auxiliary points shown in equation (29).

\[
h'(N) = \frac{1}{2} \left[ h(M - 1) + h(M) \right]
\]  

(3.29)

Equation (3.30) shows the hole current density equation linearized through the use of Taylor’s expansion theorem since it involves nonlinear functions. Since each equation is a function of one or more fundamental variables, the use of partial derivatives is applicable.

\[
J_p(M) = J_p^0(M) + \delta J_p(M) \\
= J_p^0(M) + \frac{\partial J_p^0(M)}{\partial p(N)} \delta p(N) + \frac{\partial J_p^0(M)}{\partial p(N + 1)} \delta p(N + 1) \]

\[
+ \frac{\partial J_p^0(M)}{\partial \psi(N)} \delta \psi(N) + \frac{\partial J_p^0(M)}{\partial \psi(N + 1)} \delta \psi(N + 1)
\]  

(3.30)

Solving for $\delta J_p(M)$ gives

\[
\delta J_p(M) = \frac{\partial J_p^0(M)}{\partial p(N)} \delta p(N) + \frac{\partial J_p^0(M)}{\partial p(N + 1)} \delta p(N + 1) \]

\[
+ \frac{\partial J_p^0(M)}{\partial \psi(N)} \delta \psi(N) + \frac{\partial J_p^0(M)}{\partial \psi(N + 1)} \delta \psi(N + 1)
\]
The hole current density is used as an example and the rest of the linearization process for the electron current density, recombination and generation equations is the exact same. Once the continuity equations are linearized, since they contain derivatives of the current densities, the delta state in equation (3.29) can be substituted in along with the rest shown below.

\[
\delta J_p(M) = \frac{\partial J_p^0(M)}{\partial p(N)} \delta p(N) + \frac{\partial J_p^0(M)}{\partial p(N+1)} \delta p(N+1) \\
+ \frac{\partial J_p^0(M)}{\partial \psi(N)} \delta \psi(N) + \frac{\partial J_p^0(M)}{\partial \psi(N+1)} \delta \psi(N+1)
\]

\[
\delta J_p(M-1) = \frac{\partial J_p^0(M-1)}{\partial p(N-1)} \delta p(N-1) + \frac{\partial J_p^0(M-1)}{\partial p(N)} \delta p(N) \\
+ \frac{\partial J_p^0(M-1)}{\partial \psi(N-1)} \delta \psi(N-1) + \frac{\partial J_p^0(M-1)}{\partial \psi(N)} \delta \psi(N)
\]

\[
\delta J_n(M) = \frac{\partial J_n^0(M)}{\partial n(N)} \delta n(N) + \frac{\partial J_n^0(M)}{\partial n(N+1)} \delta n(N+1) \\
+ \frac{\partial J_n^0(M)}{\partial \psi(N)} \delta \psi(N) + \frac{\partial J_n^0(M)}{\partial \psi(N+1)} \delta \psi(N+1)
\]

\[
\delta J_n(M-1) = \frac{\partial J_n^0(M-1)}{\partial n(N-1)} \delta n(N-1) + \frac{\partial J_n^0(M-1)}{\partial n(N)} \delta n(N) \\
+ \frac{\partial J_n^0(M-1)}{\partial \psi(N-1)} \delta \psi(N-1) + \frac{\partial J_n^0(M-1)}{\partial \psi(N)} \delta \psi(N)
\]

\[
\delta G(N) = \frac{\partial G^0(N)}{\partial p(N-1)} \delta p(N-1) + \frac{\partial G^0(N)}{\partial n(N-1)} \delta n(N-1) + \frac{\partial G^0(N)}{\partial \psi(N-1)} \delta \psi(N-1) \\
+ \frac{\partial G^0(N)}{\partial p(N)} \delta p(N) + \frac{\partial G^0(N)}{\partial n(N)} \delta n(N) + \frac{\partial G^0(N)}{\partial \psi(N)} \delta \psi(N) \\
+ \frac{\partial G^0(N)}{\partial p(N+1)} \delta p(N+1) + \frac{\partial G^0(N)}{\partial n(N+1)} \delta n(N+1) + \frac{\partial G^0(N)}{\partial \psi(N+1)} \delta \psi(N+1)
\]

\[
\delta U(N) = \frac{\partial U^0(N)}{\partial p(N)} \delta p(N) + \frac{\partial U^0(N)}{\partial n(N)} \delta n(N)
\]
Replacement of $J^p_p(M)$ by $J^0_p(M) + \partial J^p_p(M)$ and $J^p_p(M-1)$ by $J^0_p(M-1) + \partial J^p_p(M-1)$ and so on in equation (25c and d) and rearranging the terms yields equation (3.31).

$$\frac{1}{q h(N)} \delta J^p_p(M) - \delta J^p_p(M-1) = -\frac{1}{q h(N)} \delta G^0(N) + \delta U^0(N)$$

(3.31)

We now have the continuity equations (3.31) linearized in terms of incremental variables. The incremental variables (delta state) are given above and these can now be substituted into equation (3.31) which yields the two continuity equations involving only fundamental variable increments as unknowns. Due to the amount of physical space required to show the substitution and rearrangement process, the two equations are given directly in equation (3.32) and (3.33). It is a simple algebraic process but care must be taken that all signs are kept in their proper order and place.

Continuity equation for holes...

$$-\left[ \frac{1}{q h(N)} \frac{\partial J^0_p(M-1)}{\partial p(N-1)} + \frac{\partial G^0(N)}{\partial p(N-1)} \right] \cdot \delta p(N-1)...
$$

$$+ \left[ \frac{1}{q h(N)} \left( \frac{\partial J^0_p(M)}{\partial p(N)} - \frac{\partial J^0_p(M-1)}{\partial p(N)} \right) \right] \cdot \frac{\partial G^0(N)}{\partial p(N)} + \frac{\partial U^0(N)}{\partial p(N)} \cdot \delta p(N)...
$$

$$+ \left[ \frac{1}{q h(N)} \frac{\partial J^0_p(M)}{\partial p(N+1)} - \frac{\partial G^0(N)}{\partial p(N+1)} \right] \cdot \delta p(N+1) - \frac{\partial G^0(N)}{\partial n(N-1)} \cdot \delta n(N-1)...
$$

$$- \left[ \frac{\partial G^0(N)}{\partial n(N)} - \frac{\partial U^0(N)}{\partial n(N)} \right] \cdot \delta n(N)...
$$

37
\[ - \frac{\partial G^0(N)}{\partial n(N+1)} \cdot \delta n(N+1) - \left[ \frac{1}{qh(N)} \cdot \frac{\partial J^0_p(M-1)}{\partial \psi(N-1)} + \frac{\partial G^0(N)}{\partial \psi(N-1)} \right] \cdot \delta \psi(N-1) \]

\[ + \left\{ \frac{1}{qh(N)} \left[ \frac{\partial J^0_p(M)}{\partial \psi(N)} - \frac{\partial J^0_p(M-1)}{\partial \psi(N)} \right] - \frac{\partial G^0(N)}{\partial \psi(N)} \right\} \cdot \delta \psi(N) \]

\[ + \left[ \frac{1}{qh(N)} \cdot \frac{\partial J^0_p(M)}{\partial \psi(N+1)} - \frac{\partial G^0(N)}{\partial \psi(N+1)} \right] \cdot \delta \psi(N+1) \]  

\[ = - \frac{1}{qh(N)} \left[ J^0_p(M) - J^0_p(M-1) \right] + G^0(N) - U^0(N) \] (3.32)

Continuity equation for electrons...

\[ \frac{\partial G^0(N)}{\partial p(N-1)} \cdot \delta p(N-1) + \left[ \frac{\partial G^0(N)}{\partial p(N)} - \frac{\partial U^0(N)}{\partial p(N)} \right] \cdot \delta p(N) \]

\[ + \frac{\partial G^0(N)}{\partial p(N+1)} \cdot \delta p(N+1) \]

\[ + \left[ \frac{1}{qh(N)} \cdot \frac{\partial J^0_n(M-1)}{\partial n(N-1)} + \frac{\partial G^0(N)}{\partial n(N-1)} \right] \cdot \delta n(N-1) \]

\[ + \left\{ \frac{1}{qh(N)} \left[ \frac{\partial J^0_n(M)}{\partial n(N)} - \frac{\partial J^0_n(M-1)}{\partial n(N)} \right] + \frac{\partial G^0(N)}{\partial n(N)} - \frac{\partial U^0(N)}{\partial n(N)} \right\} \cdot \delta n(N) \]

\[ + \left[ \frac{1}{qh(N)} \cdot \frac{\partial J^0_n(M)}{\partial n(N+1)} + \frac{\partial G^0(N)}{\partial n(N+1)} \right] \cdot \delta n(N+1) \]

\[ + \left[ \frac{1}{qh(N)} \cdot \frac{\partial J^0_n(M-1)}{\partial \psi(N-1)} + \frac{\partial G^0(N)}{\partial \psi(N-1)} \right] \cdot \delta \psi(N-1) \]

\[ + \left\{ \frac{1}{qh(N)} \left[ \frac{\partial J^0_n(M)}{\partial \psi(N)} - \frac{\partial J^0_n(M-1)}{\partial \psi(N)} \right] + \frac{\partial G^0(N)}{\partial \psi(N)} \right\} \cdot \delta \psi(N) \]

\[ + \left[ \frac{1}{qh(N)} \cdot \frac{\partial J^0_n(M)}{\partial \psi(N+1)} + \frac{\partial G^0(N)}{\partial \psi(N+1)} \right] \cdot \delta \psi(N+1) \]  

\[ = - \frac{1}{qh(N)} \left[ J^0_n(M) - J^0_n(M-1) \right] - G^0(N) + U^0(N) \] (3.33)
Poisson’s equation does not involve nonlinear functions and does not technically need to be linearized in incremental variables but for ease of calculation and to match the continuity equations we have done so in equation (3.34). This allows us to setup up a standard block tri-diagonal matrix. The linearization process is not repeated but is the same for the continuity equations. For example, $p(N)$ is replaced by $p^0(N) + \partial p(N)$ and so on and then the terms rearranged and grouped.

\[
\frac{q}{\varepsilon} \delta p(N) - \frac{q}{\varepsilon} \delta n(N) + \gamma_1(N) \delta \psi(N - 1) + \gamma_2(N) \delta \psi(N) + \gamma_3(N) \delta \psi(N + 1) = -\frac{q}{\varepsilon} \left[ \Gamma(N) + p^0(N) - n^0(N) \right] \ldots \tag{3.34}
\]

These three equations (3.32-3.34) are meant to be solved simultaneously and in order to do so we can transform the three equations into a matrix-vector representation that allows us to employ the recursive algorithm discussed earlier. It will also allow us to develop the coefficients of each matrix. Equation (3.35) is the matrix-vector representation of the two continuity equations and Poisson’s equation combined.

\[
A(N) \delta y(N - 1) + B(N) \delta y(N) + C(N) \delta y(N + 1) = F(N),
\]

for $2 \leq N \leq L - 1$

To evaluate the matrix coefficients $A$, $B$, $C$, and $F$ let us examine equation (3.35) in matrix form for three division points, $N-1$, $N$, and $N+1$ as shown in Figure 3.5 and then expand each equation in the matrix to form equations (36a-c).
Now if we compare equations (3.36a-c) with equations (3.32-3.34) respectively, the matrix coefficients for A, B, C, and F are obtained directly. The matrix coefficients are tabulated below for easy reference in Figures 3.6-3.9.
\[ a_{11} = -\frac{1}{qh(N)} \left( \frac{\partial J_0^0(M)}{\partial p(N)} - \frac{\partial J_0^0(M-1)}{\partial p(N-1)} \right) \frac{\partial G^0(N)}{\partial p(N)} + \frac{\partial U^0(N)}{\partial p(N)} \]

\[ a_{12} = \frac{\partial G^0(N)}{\partial n(N)} \]

\[ a_{13} = -\frac{1}{qh(N)} \left( \frac{\partial J_0^0(M)}{\partial \psi(N)} - \frac{\partial J_0^0(M-1)}{\partial \psi(N-1)} \right) \frac{\partial G^0(N)}{\partial \psi(N)} + \frac{\partial U^0(N)}{\partial \psi(N)} \]

\[ a_{21} = \frac{\partial G^0(N)}{\partial p(N)} - \frac{\partial U^0(N)}{\partial p(N)} \]

\[ a_{22} = \frac{1}{qh(N)} \left( \frac{\partial J_0^0(M)}{\partial n(N)} - \frac{\partial J_0^0(M-1)}{\partial n(N-1)} \right) + \frac{\partial G^0(N)}{\partial n(N)} - \frac{\partial U^0(N)}{\partial n(N)} \]

\[ a_{23} = \frac{1}{qh(N)} \left( \frac{\partial J_0^0(M)}{\partial \psi(N)} - \frac{\partial J_0^0(M-1)}{\partial \psi(N-1)} \right) + \frac{\partial G^0(N)}{\partial \psi(N)} \]

\[ a_{31} = 0, \quad a_{32} = 0, \quad a_{33} = \gamma_1(N) = \frac{1}{h(M-1)h(N)} \]

Figure 3.6 – COEFFICIENT A OF MATRIX-VECTOR EQUATION

\[ b_{11} = \frac{1}{qh(N)} \left( \frac{\partial J_0^0(M)}{\partial p(N)} - \frac{\partial J_0^0(M-1)}{\partial p(N-1)} \right) \frac{\partial G^0(N)}{\partial p(N)} + \frac{\partial U^0(N)}{\partial p(N)} \]

\[ b_{12} = -\frac{\partial G^0(N)}{\partial n(N)} \]

\[ b_{13} = \frac{1}{qh(N)} \left( \frac{\partial J_0^0(M)}{\partial \psi(N)} - \frac{\partial J_0^0(M-1)}{\partial \psi(N-1)} \right) \frac{\partial G^0(N)}{\partial \psi(N)} \]

\[ b_{21} = \frac{\partial G^0(N)}{\partial n(N)} - \frac{\partial U^0(N)}{\partial n(N)} \]

\[ b_{22} = \frac{1}{qh(N)} \left( \frac{\partial J_0^0(M)}{\partial n(N)} - \frac{\partial J_0^0(M-1)}{\partial n(N-1)} \right) + \frac{\partial G^0(N)}{\partial n(N)} - \frac{\partial U^0(N)}{\partial n(N)} \]

\[ b_{23} = \frac{1}{qh(N)} \left( \frac{\partial J_0^0(M)}{\partial \psi(N)} - \frac{\partial J_0^0(M-1)}{\partial \psi(N-1)} \right) + \frac{\partial G^0(N)}{\partial \psi(N)} \]

\[ b_{31} = \frac{q}{\varepsilon}, \quad b_{32} = -\frac{q}{\varepsilon}, \quad b_{33} = \gamma_2(N) \left( \frac{1}{h(M-1)} + \frac{1}{h(N)} \right) \]

Figure 3.7 – COEFFICIENT B OF MATRIX-VECTOR EQUATION
\[ c_{11} = \frac{1}{qN(N)} \frac{\partial J^0_p(M)}{\partial p(N+1)} - \frac{\partial G^0(N)}{\partial p(N+1)}, \quad c_{12} = -\frac{\partial G^0(N)}{\partial p(N+1)} \]

\[ c_{13} = \frac{1}{qN(N)} \frac{\partial J^0_p(M)}{\partial \psi(N+1)} - \frac{\partial G^0(N)}{\partial \psi(N+1)}, \quad c_{21} = \frac{\partial G^0(N)}{\partial p(N+1)} \]

\[ c_{22} = \frac{1}{qN(N)} \frac{\partial J^0_p(M)}{\partial n(N+1)} + \frac{\partial G^0(N)}{\partial n(N+1)} \]

\[ c_{23} = \frac{1}{qN(N)} \frac{\partial J^0_p(M)}{\partial \psi(N+1)} + \frac{\partial G^0(N)}{\partial \psi(N+1)} \]

\[ c_{31} = 0, \quad c_{32} = 0, \quad c_{33} = \frac{1}{h(M)h(N)} \]

Figure 3.8 – COEFFICIENT C OF MATRIX-VECTOR EQUATION

\[ f_{11} = -\frac{1}{qN(N)} \left[ J^0_p(M) - J^0_p(M-1) \right] + G^0(N) - U^0(N) \]

\[ f_{12} = -\frac{1}{qN(N)} \left[ J^0_p(M) - J^0_p(M-1) \right] - G^0(N) + U^0(N) \]

\[ f_{13} = -\frac{q}{\varepsilon} \left[ \Gamma(N) + p^0(N) - n^0(N) \right] \]

\[ - \gamma_1(N)\psi^0(N-1) - \gamma_2(N)\psi^0(N) - \gamma_3(N)\psi^0(N+1) \]

Figure 3.9 – COEFFICIENT F OF MATRIX-VECTOR EQUATION

Now that we have the coefficients for the matrix-vector equation determined they can be calculated across the whole device in order to solve equation (3.35). However, as we have seen the recursive algorithm is more efficient and requires less computations than solving equation (3.35) directly. Refer to section 3.4 for the recursive algorithm and appropriate relationships.

Before the coefficients for equation (3.35) can be evaluated the partial derivatives embedded in the coefficients must be determined first. The partial derivatives are determined directly from the current density, recombination, and generation equations.
(3.27), (3.21), and (3.22) respectively. The derivation techniques for several of the partial derivatives are given in Appendix B, you may refer to that now for more information. The required partial derivatives are given in Table 3.1 below for reference. Their calculations are given below in summarized form. No equations numbers are associated with the calculations; they must be referenced to the table numbers below. It may seem excessive but the number of partial derivatives is not as extensive as the list appears. Most of the derivatives can be coded into one function as we will see in Chapter 5. The first table below simply lists all the partial derivatives required to support the matrix coefficients $A$, $B$, $C$, and $F$.

### Table 3.1 – Partial derivatives required for matrix coefficients

<table>
<thead>
<tr>
<th>Partial Derivative</th>
<th>(1a)</th>
<th>(1b)</th>
<th>(1c)</th>
<th>(1d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{\partial J^0_p(M-1)}{\partial p(N)}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\frac{\partial J^0_p(M-1)}{\partial p(N-1)}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\frac{\partial J^0_p(M-1)}{\partial \psi(N)}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\frac{\partial J^0_p(M-1)}{\partial \psi(N-1)}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\frac{\partial J^0_n(M-1)}{\partial n(N-1)}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\frac{\partial J^0_n(M-1)}{\partial n(N)}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\frac{\partial J^0_n(M-1)}{\partial \psi(N-1)}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\frac{\partial J^0_n(M-1)}{\partial \psi(N)}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Referring to the current density equation from Appendix A at division point $M$-1 and $M$ repeated below in equation (3.37), the partial derivatives for (1a-d) are obtained directly.

43
\[ J_p(M - 1) = \frac{q}{h(M - 1)} \left[ \lambda_{p1}(M - 1)p(N - 1) + \lambda_{p2}(M - 1)p(N) \right] \]

\[ J_p(M) = \frac{q}{h(M)} \left[ \lambda_{p1}(M)p(N) + \lambda_{p2}(M)p(N + 1) \right] \]

\[
\begin{align*}
\frac{\partial J_P^0(M - 1)}{\partial p(N - 1)} &= \frac{q}{h(M - 1)} \lambda_{p1}(M - 1), \quad (la) \\
\frac{\partial J_P^0(M - 1)}{\partial p(N)} &= \frac{q}{h(M - 1)} \lambda_{p2}(M - 1), \quad (lb) \\
\frac{\partial J_P^0(M)}{\partial p(N)} &= \frac{q}{h(M)} \lambda_{p1}(M), \quad (lc) \\
\frac{\partial J_P^0(M)}{\partial p(N + 1)} &= \frac{q}{h(M)} \lambda_{p2}(M), \quad (ld)
\end{align*}
\]

The partial derivative of the current density with respect to the potential, \(\psi\) is more complicated and requires use of the product rule and chain rule for derivatives since the lambda terms are functions of the potential themselves. The results are tabulated below. The lambda terms calculated as a part of the current density equations are repeated here for reference in equation (3.38). The partial derivatives for (2a-b) are tabulated below.

\[
\begin{align*}
\lambda_{p1}(M) &= \mu_p(M) \frac{\psi(N) - \psi(N + 1)}{1 - e^{-\beta(M)}}, \\
\lambda_{p2}(M) &= \mu_p(M) \frac{\psi(N) - \psi(N + 1)}{1 - e^{-\beta(M)}}, \\
\lambda_{p1}(M - 1) &= \mu_p(M - 1) \frac{\psi(N - 1) - \psi(N)}{1 - e^{-\beta(M)}}, \\
\lambda_{p2}(M - 1) &= \mu_p(M - 1) \frac{\psi(N - 1) - \psi(N)}{1 - e^{-\beta(M)}}, \quad \text{where} \\
\beta(M) &= \theta[\psi(N) - \psi(N + 1)] \\
\beta(M - 1) &= \theta[\psi(N - 1) - \psi(N)]
\end{align*}
\]
\[
\frac{\partial \lambda_{p1}(M-1)}{\partial \psi(N-1)} = \frac{q}{h(M-1)} \left\{ p(N-1) \frac{\partial \lambda_{p1}(M-1)}{\partial \psi(N-1)} + p(N) \frac{\partial \lambda_{p2}(M-1)}{\partial \psi(N-1)} \right\} \quad (2a)
\]

where,
\[
\frac{\partial \lambda_{p1}(M-1)}{\partial \psi(N-1)} = \frac{\mu_p(M-1)}{(1-e^{-\beta(M-1)})} \left[ 1 - \frac{\beta(M-1)e^{-\beta(M-1)}}{1-e^{-\beta(M-1)}} \right]
\]
\[
\frac{\partial \lambda_{p2}(M-1)}{\partial \psi(N-1)} = \frac{\mu_p(M-1)}{(1-e^{\beta(M-1)})} \left[ 1 + \frac{\beta(M-1)e^{\beta(M-1)}}{1-e^{\beta(M-1)}} \right]
\]

\[
\frac{\partial \lambda_0(M-1)}{\partial \psi(N)} = \frac{\partial \lambda_0(M-1)}{\partial \psi(N-1)} \quad (2b)
\]

where,
\[
\frac{\partial \lambda_{p1}(M-1)}{\partial \psi(N)} = -\frac{\partial \lambda_{p1}(M-1)}{\partial \psi(N-1)}
\]
\[
\frac{\partial \lambda_{p2}(M-1)}{\partial \psi(N)} = -\frac{\partial \lambda_{p2}(M-1)}{\partial \psi(N-1)}
\]

\[
\frac{\partial \lambda_0(M)}{\partial \psi(N)} = \frac{q}{h(M)} \left\{ p(N) \frac{\partial \lambda_{p1}(M)}{\partial \psi(N)} + p(N+1) \frac{\partial \lambda_{p2}(M)}{\partial \psi(N)} \right\} \quad (2c)
\]

where,
\[
\frac{\partial \lambda_{p1}(M)}{\partial \psi(N)} = \frac{\mu_p(M)}{(1-e^{-\beta(M)})} \left[ 1 - \frac{\beta(M)e^{-\beta(M)}}{1-e^{-\beta(M)}} \right]
\]
\[
\frac{\partial \lambda_{p2}(M)}{\partial \psi(N)} = \frac{\mu_p(M)}{(1-e^{\beta(M)})} \left[ 1 + \frac{\beta(M)e^{\beta(M)}}{1-e^{\beta(M)}} \right]
\]

45
\[
\frac{\partial J^0_p (M)}{\partial \psi(N+1)} = - \frac{\partial J^0_p (M)}{\partial \psi(N)} \quad (2d)
\]

where,
\[
\frac{\partial \lambda_{p_1} (M)}{\partial \psi(N+1)} = - \frac{\partial \lambda_{p_1} (M)}{\partial \psi(N)}
\]
\[
\frac{\partial \lambda_{p_2} (M)}{\partial \psi(N+1)} = - \frac{\partial J^0_p (M)}{\partial \psi(N)}
\]

The procedure for evaluating the electron related partials derivatives in 3a-d and 4a-d are exactly the same. The partial derivatives for these terms are tabulated below.

\[
\frac{\partial J^0_n (M - 1)}{\partial n(N - 1)} = \frac{q}{h(M - 1)} \lambda_{n_1} (M - 1), \quad (3a)
\]
\[
\frac{\partial J^0_n (M - 1)}{\partial n(N)} = \frac{q}{h(M - 1)} \lambda_{n_2} (M - 1), \quad (3b)
\]
\[
\frac{\partial J^0_n (M)}{\partial n(N)} = \frac{q}{h(M)} \lambda_{n_1} (M), \quad (3c)
\]
\[
\frac{\partial J^0_n (M)}{\partial n(N + 1)} = \frac{q}{h(M)} \lambda_{n_2} (M), \quad (3d)
\]
\[
\frac{\partial J^0_n (M - 1)}{\partial \psi(N - 1)} = \frac{q}{h(M - 1)} \left\{ n(N - 1) \frac{\partial \lambda_{n_1} (M - 1)}{\partial \psi(N - 1)} + n(N) \frac{\partial \lambda_{n_2} (M - 1)}{\partial \psi(N - 1)} \right\} \quad (4a)
\]

where,
\[
\frac{\partial \lambda_{n_1} (M - 1)}{\partial \psi(N - 1)} = \frac{\mu_n (M - 1)}{\mu_p (M - 1)} \frac{\partial \lambda_{p_2} (M - 1)}{\partial \psi(N - 1)}
\]
\[
\frac{\partial \lambda_{n_2} (M - 1)}{\partial \psi(N - 1)} = \frac{\mu_n (M - 1)}{\mu_p (M - 1)} \frac{\partial \lambda_{p_1} (M - 1)}{\partial \psi(N - 1)}
\]
\[
\frac{\partial J^0_n (M - 1)}{\partial \psi(N)} = \frac{q}{h(M - 1)} \left\{ n(N - 1) \frac{\partial \lambda_{n_1} (M - 1)}{\partial \psi(N)} + n(N) \frac{\partial \lambda_{n_2} (M - 1)}{\partial \psi(N)} \right\} \quad (4b)
\]
where,
\[
\frac{\partial \lambda_{n1}(M-1)}{\partial \psi(N)} = -\frac{\partial \lambda_{n1}(M-1)}{\partial \psi(N-1)} = -\frac{\mu_n(M-1)}{\mu_p(M-1)} \frac{\partial \lambda_{p2}(M-1)}{\partial \psi(N-1)}
\]
\[
\frac{\partial \lambda_{n2}(M-1)}{\partial \psi(N)} = -\frac{\partial \lambda_{n2}(M-1)}{\partial \psi(N-1)} = -\frac{\mu_n(M-1)}{\mu_p(M-1)} \frac{\partial \lambda_{p1}(M-1)}{\partial \psi(N-1)}
\]

\[
\frac{\partial J_n^0(M)}{\partial \psi(N)} = \frac{q}{h(M)} \left\{ n(N) \frac{\partial \lambda_{n1}(M)}{\partial \psi(N)} + n(N+1) \frac{\partial \lambda_{n2}(M)}{\partial \psi(N)} \right\} \quad (4c)
\]

where,
\[
\frac{\partial \lambda_{n1}(M)}{\partial \psi(N)} = \frac{\mu_n(M)}{\mu_p(M)} \frac{\partial \lambda_{p2}(M)}{\partial \psi(N)}
\]
\[
\frac{\partial \lambda_{n2}(M)}{\partial \psi(N)} = \frac{\mu_n(M)}{\mu_p(M)} \frac{\partial \lambda_{p1}(M)}{\partial \psi(N)}
\]

\[
\frac{\partial J_n^0(M)}{\partial \psi(N+1)} = \frac{q}{h(M)} \left\{ n(N) \frac{\partial \lambda_{n1}(M)}{\partial \psi(N+1)} + n(N+1) \frac{\partial \lambda_{n2}(M)}{\partial \psi(N+1)} \right\} \quad (4d)
\]

where,
\[
\frac{\partial \lambda_{n1}(M)}{\partial \psi(N+1)} = -\frac{\partial \lambda_{n1}(M)}{\partial \psi(N)} = \frac{\mu_n(M)}{\mu_p(M)} \frac{\partial \lambda_{p2}(M)}{\partial \psi(N)}
\]
\[
\frac{\partial \lambda_{n2}(M)}{\partial \psi(N+1)} = -\frac{\partial \lambda_{n2}(M)}{\partial \psi(N)} = \frac{\mu_n(M)}{\mu_p(M)} \frac{\partial \lambda_{p1}(M)}{\partial \psi(N)}
\]

The amount of partial derivatives for the current density equations may seem excessive, but as will be explained in Chapter 5, only one function for the lambda coefficients and partial derivative lambda expressions are required to calculate all of the above which allows the calculation for each of the matrix coefficients to be more efficient. The partial derivatives for the recombination and generation terms are also given below. Their derivations are similar and briefly discussed in Appendix B as well.
Recombination terms:

\[
\frac{\partial U(N)}{\partial p(N)} = \frac{n(N) - \tau_n U(N)}{\tau_n (p(N) + n_i) + \tau_p (n(N) + n_i)} \tag{5a}
\]

\[
\frac{\partial U(N)}{\partial n(N)} = \frac{p(N) + \tau_p U(N)}{\tau_n (p(N) + n_i) + \tau_p (n(N) + n_i)} \tag{5b}
\]

Generation terms:

\[
\frac{\partial G(N)}{\partial p(N - 1)} = \pm \frac{m_a}{h(M - 1)} \cdot \lambda_p (M - 1) \cdot \alpha_p (N) \tag{6a}
\]

\[
\frac{\partial G(N)}{\partial p(N)} = \pm \left[ \frac{m_a}{h(M - 1)} \cdot \lambda_p (M - 1) + \frac{m_b}{h(M)} \cdot \lambda_p (M) \right] \cdot \alpha_p (N) \tag{6b}
\]

\[
\frac{\partial G(N)}{\partial p(N + 1)} = \pm \frac{m_b}{h(M)} \cdot \lambda_p (M) \cdot \alpha_p (N) \tag{6c}
\]

where,

\[
m_a = \frac{h(M)}{2 h(N)}, \quad m_b = \frac{h(M - 1)}{2 h(N)}
\]

\[
\alpha_p (N) = \alpha_{p0} \exp \left[ - \frac{E_{p0}}{|E(N)|} \right] \quad \text{and} \quad E(N) = m_a E(M - 1) + m_b E(M)
\]

The choice of the sign for the partial derivative terms depends upon the sign of the current density at the given mesh point, that is \( J_p (N) > 0 \) or \( J_p (N) < 0 \) determines + or - respectively when,

\[
J_p (N) = m_a J_p (M - 1) + m_b J_p (M)
\]
\[
\frac{\partial G(N)}{\partial n(N-1)} = \pm \frac{m_a}{h(M-1)} \cdot \lambda_{n_1}(M-1) \cdot \alpha_n(N) \quad (7a)
\]

\[
\frac{\partial G(N)}{\partial n(N)} = \pm \left[ \frac{m_a}{h(M-1)} \cdot \lambda_{n_2}(M-1) + \frac{m_b}{h(M)} \cdot \lambda_{n_1}(M) \right] \cdot \alpha_n(N) \quad (7b)
\]

\[
\frac{\partial G(N)}{\partial n(N+1)} = \pm \frac{m_b}{h(M)} \cdot \lambda_{n_2}(M) \cdot \alpha_n(N) \quad (7c)
\]

\[
m_a = \frac{h(M)}{2h(N)}, \quad m_b = \frac{h(M-1)}{2h(N)}
\]

\[
\alpha_n(N) = \alpha_{n_0} \exp \left[ -\frac{E_{n_0}}{E(N)} \right] \quad \text{and} \quad E(N) = m_a E(M-1) + m_b E(M)
\]

The choice of the sign for the partial derivative terms again depends upon the sign of the current density at the given mesh point, that is \( J_n(N) > 0 \) or \( J_n(N) < 0 \) determines + or − respectively when,

\[
J_n(N) = m_a J_n(M-1) + m_b J_n(M)
\]

The choice of the sign \( \pm \) or \( \mp \) for the remaining partial derivatives below depends upon whether \( J_n(N), J_p(N), E(N) > 0 \) or \( J_n(N), J_p(N), E(N) < 0 \) respectively. For example, \( G_{\psi_1} \) is positive is \( E(N) > 0 \) but negative if \( E(N) < 0 \), however \( G_{\psi_3} \) is negative is \( E(N) > 0 \) but positive if \( E(N) < 0 \). The term that governs the sign follows each \( G_{\psi ij} \) below.
\[ \frac{\partial G(N)}{\partial \psi(N-1)} = G_{\psi 11} + G_{\psi 12} + G_{\psi 13} + G_{\psi 14} \quad (8a) \]

\[ \frac{\partial G(N)}{\partial \psi(N)} = G_{\psi 21} + G_{\psi 22} + G_{\psi 23} + G_{\psi 24} \quad (8b) \]

\[ \frac{\partial G(N)}{\partial \psi(N+1)} = G_{\psi 31} + G_{\psi 32} + G_{\psi 33} + G_{\psi 34} \quad (8c) \]

where,

\[ G_{\psi 11} = \pm \frac{m_a}{h(M - 1)} \alpha_p(N) \frac{E_{p0}}{E(N)^2}, \quad \frac{|J_p(N)|}{q}, \quad E(N) \]

\[ G_{\psi 12} = \pm \frac{m_a \alpha_p(N)}{q} \frac{\partial J_p(M - 1)}{\partial \psi(N - 1)}, \quad J_p(N) \]

\[ G_{\psi 13} = \pm \frac{m_a}{h(M - 1)} \alpha_n(N) \frac{E_{n0}}{E(N)^2}, \quad \frac{|J_n(N)|}{q}, \quad E(N) \]

\[ G_{\psi 14} = \pm \frac{m_a \alpha_n(N)}{q} \frac{\partial J_n(M - 1)}{\partial \psi(N - 1)}, \quad J_n(N) \]

\[ G_{\psi 31} = \mp \frac{m_b}{h(M)} \alpha_p(N) \frac{E_{p0}}{E(N)^2}, \quad \frac{|J_p(N)|}{q}, \quad E(N) \]

\[ G_{\psi 32} = \mp \frac{m_b \alpha_p(N)}{q} \frac{\partial J_p(M)}{\partial \psi(N)}, \quad J_p(N) \]

\[ G_{\psi 33} = \mp \frac{m_b}{h(M)} \alpha_n(N) \frac{E_{n0}}{E(N)^2}, \quad \frac{|J_n(N)|}{q}, \quad E(N) \]

\[ G_{\psi 34} = \mp \frac{m_b \alpha_n(N)}{q} \frac{\partial J_n(M)}{\partial \psi(N)}, \quad J_n(N) \]
With all of the coefficients for the matrix-vector form of the continuity equations and Poisson’s equation tabulated and the partial derivatives for those coefficients calculated, we can use them to develop an algorithm which calculates the thermal equilibrium or applied DC bias potential and carrier densities throughout the device.

### 3.6. Numerical Solution

This section will discuss the numerical solution process used to solve the matrix-vector equation (3.35), including development of the trial values, or initial guess, and finally Newton’s iteration principle to update the trial values and check for convergence of the solution. A basic solution process is also discussed which was used to develop the program given in Appendix C.

This solution algorithm involves solving the two continuity equations and Poisson’s equation as previously discussed. The process of transforming these equations into finite difference equations allows the use of the implicit method as a solution procedure which is simply a multidimensional Newton iteration problem. Equations (3.32-3.34) have been developed such that full coupling exists between the three fundamental variables, \( n \), \( p \), and \( \psi \) or that is each fundamental variable is allowed to change during each iteration. In this manner the equations can be solved simultaneously in a straightforward fashion. Other methods exist to solve the numerical equation developed including decoupled methods (Gummel method [3]) but are not discussed here. They have their various differences and benefits for specialized situations but for a generic solution algorithm for most possible device conditions (low injection vs. high injection for example) the couple Newton iteration
procedure is preferred. It will increase the overall number of computations per
iteration but decrease the overall computation time for convergence.

The solution to equation (3.35), repeated below for reference and show in matrix
form, can involve a rather significant amount of matrix elements. An efficient
solution procedure is required or the computation time, even for simple devices, can
become excessive and limits the use of such simulator tools to any practical sense.
Use of the coupled Newton iteration principle with the recursive solution procedure
will suit the needs of this program adequately.

\[ A(N) \delta y(N-1) + B(N) \delta y(N) + C(N) \delta y(N+1) = F(N), \quad \text{for} \quad 2 \leq N \leq L-1 \]

where,

\[ y(N) = \begin{bmatrix} p(N) \\ n(N) \\ y(N) \end{bmatrix}, \quad \delta y(N) = \begin{bmatrix} \delta p(N) \\ \delta n(N) \\ \delta y(N) \end{bmatrix} \]

\[
\begin{bmatrix}
B(2) & C(2) & 0 & \cdots & \cdots & 0 \\
A(3) & B(3) & C(3) & & & 0 \\
0 & A(4) & B(4) & C(4) & & \vdots \\
& \vdots & & & & \vdots \\
0 & \cdots & 0 & A(L-2) & B(L-2) & C(L-2) \\
0 & \cdots & \cdots & 0 & A(L-1) & B(L-1)
\end{bmatrix} \begin{bmatrix}
\delta y(2) \\
\delta y(3) \\
\vdots \\
\delta y(L-2) \\
\delta y(L-1)
\end{bmatrix} = \begin{bmatrix}
F(2) \\
F(3) \\
\vdots \\
F(L-2) \\
F(L-1)
\end{bmatrix}
\]

Newton’s iteration principle requires that we have an initial guess or starting value to
calculate the error state for the first iteration. The calculated error is then added to the
initial value and the process repeated until convergence is reached. All devices,
whether simple 1-D P-N junctions or elaborate 2-D bipolar devices, should have the
thermal equilibrium condition solved for first, or that is no external applied bias. Not
only does it allow for faster computation speeds on the applied bias conditions but
allows the user to obtain a good feel for whether the device’s equilibrium conditions
are being met as intended.
For the thermal equilibrium condition, trial potentials are simply provided based on the charge neutral assumption that the boundary conditions were established under.

Carrier densities are given by $p_n = -\Gamma$, $n_n = -\frac{n_i^2}{\Gamma}$, where $\Gamma$ is the effective doping concentration throughout the device. The zero-bias trial values for the potential are given by equation (3.39) below.

$$
\psi_T = \begin{cases} 
\frac{1}{\theta} \ln \left[ -\frac{n_i}{\Gamma(x)} \right], & p - \text{region} \\
\frac{1}{\theta} \ln \left[ \frac{\Gamma(x)}{n_i} \right], & n - \text{region}
\end{cases}
$$

(3.39)

Now that the trial potentials are defined we can conceptualize a solution procedure, which is given below. A flow chart representation is provided in [4] as well as the author’s version in Chapter 5, which represents this program’s solution.

1. Define physical and numerical parameters
2. Define iteration limit and set current iteration number to zero
3. Use the matrix-vector coefficients, A, B, C, and F to calculate the error for each iteration using the recursive algorithm discussed in Section 3.4. This is equivalent to solving equation (33) using direct Gaussian elimination for example.
4. Improve the initial guess or trial potentials by adding the calculated error.
5. Check for convergence against defined convergence tolerance
6. End program if solution has converged otherwise update current iteration limit by 1 and return to step (3) repeating the process until solution has converged or iteration limit is reached.

Assuming that a converged solution for the equilibrium condition has been reached, extending the solution to an external biased condition only requires updating the equilibrium condition potential with a simple proportional relationship and then repeating the process defined above. Technically updating the potential is not necessary but will lower the total iteration number for the DC condition. This relationship is given in equation (3.40) and illustrated in Figure 3.10 below.
\[ \psi_r(N) = \left[ 1 - \frac{V_2 - V_1}{\psi_1(L) - \psi_i(L)} \right] \cdot \psi_i(N) + \frac{V_2 - V_1}{\psi_1(L) - \psi_i(L)} \cdot \psi_i(L) \]  

Figure 3.10 – UPDATED POTENTIAL BASED ON PREVIOUS POTENTIAL [4]

A tabulated list of physical constants and user defined constants as well as a tabulated list of numerical constants required to develop the program in Appendix C is given below.

Table 3.2 – Numerical constants

<table>
<thead>
<tr>
<th>Nomenclature</th>
<th>Program Variable</th>
<th>Default Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Length</td>
<td>length</td>
<td>10</td>
<td>um</td>
</tr>
<tr>
<td>Convergence Tolerance</td>
<td>etol</td>
<td>1 x 10^-6</td>
<td></td>
</tr>
<tr>
<td># Grid Points</td>
<td>L</td>
<td>1001</td>
<td></td>
</tr>
<tr>
<td>Iteration Number</td>
<td>it</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Iteration Limit</td>
<td>itl</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>
Table 3.3 – Physical constants

<table>
<thead>
<tr>
<th>Nomenclature</th>
<th>Program Variable</th>
<th>Default Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dielectric Permittivity Constant</td>
<td>eo</td>
<td>$8.854 \times 10^{-14}$</td>
<td>F/cm</td>
</tr>
<tr>
<td>Electronic Charge</td>
<td>q</td>
<td>$1.602 \times 10^{-19}$</td>
<td>C (A-sec)</td>
</tr>
<tr>
<td>Boltzmann’s Constant</td>
<td>k</td>
<td>$8.62 \times 10^{-5}$</td>
<td>eV/K</td>
</tr>
<tr>
<td>Thermal Voltage</td>
<td>Vt</td>
<td>0.02586</td>
<td>V</td>
</tr>
<tr>
<td>Boltzmann Factor (Thermal Voltage)</td>
<td>l/Vt</td>
<td>38.6</td>
<td>l/V</td>
</tr>
<tr>
<td>Temperature</td>
<td>T</td>
<td>300</td>
<td>K</td>
</tr>
<tr>
<td>Donor Concentration</td>
<td>Nd</td>
<td>$1 \times 10^{15}$</td>
<td>cm$^{-3}$</td>
</tr>
<tr>
<td>Acceptor Concentration</td>
<td>Na</td>
<td>$1 \times 10^{15}$</td>
<td>cm$^{-3}$</td>
</tr>
<tr>
<td>Left Contact</td>
<td>Vl</td>
<td>0</td>
<td>V</td>
</tr>
<tr>
<td>Right Contact</td>
<td>Vr</td>
<td>0</td>
<td>V</td>
</tr>
</tbody>
</table>
4. SDS Results Investigated

In this chapter, we will review the output of the SDS program and compare it to analytical results as well as to other simulators, notable SimWindows. [7] SimWindows is also a 1-D semiconductor device simulator capable of simulating the same devices as the SDS program. The first section will compare the SDS results with respect to standard analytical models, for example the built-in potential with respect to using the depletion approximation as opposed to solving the numerical solution which makes no such assumptions. The purpose of this is to verify the working operation of the program versus general semiconductor operation that is learned at the classroom level.

4.1. SDS Output Compared to Analytical Assumptions

Most basic PN Junction analysis is performed using the analytical device expressions as defined in Chapter 2. To make these differential equations mathematically tractable, assumptions must be made in order to arrive at a solution. One of the main assumptions used in this approximation is the Depletion Approximation. [1]

The Depletion Approximation is a way to simplify Poisson’s equation such that the potential throughout the device can be calculated without the need of the carrier concentrations, n and p. The approximation is performed by assuming that the transition region between the n and p bulk regions, or that is depletion region, is totally depleted of mobile carriers and that the mobile majority carrier densities abruptly become equal to corresponding dopant concentrations at the edges of the depletion region. The charge density is therefore zero everywhere except in the depletion region, where it equals the ionized dopant concentration. Additionally, we generally assume that the dopant atoms are complete (100%) ionized and free to contribute to current conduction. This reduces Poisson’s equation to what is shown in equation (4.1) below.
If we further only evaluate Poisson’s equation with respect to the n-region of the PN junction, equation (4.1) reduces to equation (4.2) below. This reflects no acceptor atoms in the n-region and can easily be integrated from an arbitrary point in the n-region depletion region to the edge of the region resulting in equation (4.3). Similar results are obtained in the p-region in equation (4.4).

\[
\frac{\partial^2 \phi}{\partial x^2} = -\frac{\partial E}{\partial x} = -\frac{qN_d}{\varepsilon_s}
\]  
(4.2)

\[
E(x) = -\frac{qN_d}{\varepsilon_s} (x_n - x) \quad 0 < x < x_n
\]  
(4.3)

\[
E(x) = -\frac{qN_a}{\varepsilon_s} (x + x_p) \quad -x_p < x < 0
\]  
(4.4)

If we further integrate equations (4.3) and (4.4) one more time, we arrive at expressions for the potential in each respective n and p-region shown in equations (4.5) and (4.6).

\[
\phi(x) = \phi_n - \frac{qN_d}{2\varepsilon_s} (x_n - x)^2 \quad 0 < x < x_n
\]  
(4.5)

\[
\phi(x) = \phi_p + \frac{qN_a}{\varepsilon_s} (x + x_p)^2 \quad -x_p < x < 0
\]  
(4.6)

Where \( \phi_n \) and \( \phi_p \) is the potential at the edges of the depletion regions and is equal to

\[
\phi_n = \frac{kT}{q} \ln \frac{N_d}{n_i} \quad \text{and} \quad \phi_p = -\frac{kT}{q} \ln \frac{N_a}{n_i}
\]  
(4.7)
The total potential change from one region to the other is referred to as the built-in potential and is the potential whereby at equilibrium it balances the drift and diffusion components across the junction. The expression for the built-in potential is shown below in equation (4.8).

\[
\phi_i = \frac{kT}{q} \ln \frac{N_d N_a}{n_i^2}
\]  

(4.8)

With these expressions developed from an analytical approach using the Depletion Approximation we can evaluate the SDS program solution with respect to the analytical solution. The equations derived above generally result in the graphs shown in Figure 4.1. These graphs apply to an abrupt step junction which is the default device loaded into SDS.

Plot (a) shows the step junction profile, with one dopant type per region. This creates a step junction trial potential for use in the numerical solution. Plot (b) is simply showing the depletion approximation, or that is, the depletion region carrier density is negligible. Plot (c) is the charge density associated with the dopant atoms. Plot (d) is the electric field in the device, while plot (d) is the potential in the device.
From plot (d) in Figure 4.1 the potential is only zero at the transition region when the doping profiles on each side are equal. When they are not, the depletion region will push further into the region that has less doping.
Referencing plot (d) and equations (7) and (8), we can calculate the built in potential and potential in the n and p-regions only to see if they compare to our numerical solution. Let us assume that the dopant profile in each region is equal which gives us

\[
\phi_n = \frac{kT}{q} \ln \frac{N_d}{n_i} = \frac{1.38 \times 10^{-23} \cdot 300}{1.602 \times 10^{-19}} \cdot \ln \frac{1 \times 10^{15}}{(1.45 \times 10^{10})^2} = 0.2878 \text{V} \quad \text{and}
\]

\[
\phi_p = -\frac{kT}{q} \ln \frac{N_a}{n_i} = -\frac{1.38 \times 10^{-23} \cdot 300}{1.602 \times 10^{-19}} \cdot \ln \frac{1 \times 10^{15}}{(1.45 \times 10^{10})^2} = -0.2878 \text{V}
\]

\[
\phi_t = \phi_n - \phi_p = \frac{kT}{q} \ln \frac{N_d N_a}{n_i^2} = \frac{1.38 \times 10^{-23} \cdot 300}{1.602 \times 10^{-19}} \cdot \ln \frac{1 \times 10^{15} \cdot 1 \times 10^{15}}{(1.45 \times 10^{10})^2} = 0.5756 \text{V}
\]

Figure 4.2 shows the output for the potential of the SDS program when run with an abrupt step junction of equal doping in both regions. The cursor in the graph is displaying the pertinent information.

Figure 4.2 – EQUILIBRIUM DEVICE POTENTIAL
We can also observe in Figure 4.2 that the potential is zero at the junction transition as we expect since the doping densities are equal in the bulk n and p-regions. Figure 4.3 shows the equilibrium electric field throughout the device. It is also symmetrical about the junction as we expect.

![Equilibrium Electric Field](image)

**Figure 4.3 – EQUILIBRIUM ELECTRIC FIELD**

Figures 4.4 and 4.5 show the potential for a PN junction where the acceptor dopant density is greater than the donor density. This shows the depletion region extending further into the n-region as it should, as well as a nonzero value at the transition region. Figure 4.5 is taken from SimWindows as verification the correct solution was obtained. The only difference in SimWindows is that all plots are scaled to zero, meaning there are no negative potential values. SDS has an option to do this as well and the result is shown in Figure 4.6. The electric field is always max at the junction transition; however when the dopant densities are unequal across the junction, the field will be push further into the region with less doping atoms just as the depletion region does.
Figure 4.4 – DEVICE POTENTIAL WITH UNEQUAL DOPING DENSITIES

Figure 4.5 – SIMWINDOWS COMPARISON TO FIGURE 4.4
Figure 4.6 – SDS OUTPUT SHIFTED TO ZERO

Figure 4.7 below is the SimWindows equilibrium electric field with unequal doping densities. This compares to the SDS output in Figure 4.8.

Figure 4.7 – SIMWINDOWS ELECTRIC FIELD
Finally the abrupt step junction is evaluated with an external applied bias of 0.5 volts at the left contact. This forward biases the PN junction as we would expect. A 0.5 volt signal is also applied at the right contact to reverse bias the junction. Alternatively, a negative voltage could be applied to the left contact to achieve the same affect. The doping densities are kept unequal as in the previous examples.

Figure 4.9 shows the device potential when the device is forwarded biased. From classic analytical theory we expect the built-in potential to be lowered as is shown in the graph. Figure 4.10 is a SimWindows comparison of the same output.

While the device is forwarded biased we can also calculate the current density through the device and compare it to SimWindows output as well. One final experiment not shown here is to reverse bias the device and slowly increase the bias.
until a breakdown condition is achieved. This better shows the effect of the generation dependent models as a high bias tends to achieve impact ionization more readily than lower level electric fields.

Figure 4.9 – SDS DEVICE POTENTIAL WITH EXTERNAL APPLIED BIAS
Figure 4.10 – SIMWINDOWS POTENTIAL WITH EXTERNAL APPLIED BIAS
5. Semiconductor Device Simulator

In this chapter, a brief overview of the SDS program is given with explanations for key aspects of the program that are not explicitly covered in any other section. We will begin by looking at the "main" program and then investigate the functions that are implemented in support of the main program.

As we have seen throughout the project, the main purpose is to write a program that solves the matrix-vector equation which was established from the classic differential equations in Chapter 2. In most aspects the program can be developed however the developer likes, but in certain instances poor coding techniques will lead to longer computation times and ultimately slower convergence speeds for a given device setup. The main important goal is to manipulate the matrix coefficients efficiently and to solve the matrix-vector equation with as little iteration as possible. On key element to this is establishing the proper boundary conditions and trial values.

The first few blocks simply establish any constants, or numerical parameters required to solve the device. The second block establishes any user defined inputs which also include device information for the device to be simulated. A note should be made that the code provided in Appendix III, and referenced in sections here, is heavily commented and the reader should reference the comments for any confusion that may arise.

clear all;  %Clears all previously stored variable information
close all;  %Closes all previously open plots
clc;  %Clears the command window to blank
TSTART = tic;  %Starts a timer to record solution time
format long eng;  %Formats the data to engineering mode

This block is a simple setup that is used before any new simulation is run. All stored variable information in the Matlab workspace is cleared, all open plots are closed, and the command window is cleared. This section also starts a timer, which is used to investigate the speed of the computation algorithm for various device setups.
Every calculated value that is stored in a variable is accessible after the program is run in the main Matlab workspace. For example, the last iterated value for the error values is stored in an error vector. The program can be easily modified to run through one iteration at a time allowing the user to investigate the change in error over each iteration. Every opportunity for the user to access the calculated data has been provided so that any insight into those variables can be gained if necessary.

One of the main reasons the author chose to use Matlab was due to the environment's natural capability to work with matrices. Everything in Matlab, even a one-dimensional variable or constant is treated as a matrix. One of the key features in SDS is the use of Cell arrays in Matlab. This allows the user to store various types of data in each cell. In fact, the first cell could have a standard 3x3 matrix, while the second cell has a string text in it for example.

The cell arrays are treated like matrices as well and indexed similarly. All standard matrix operations work, provided that matrix dimensions agree, etc. This works elegantly with the tri-diagonal block matrices. Each matrix coefficient is a 3x3 matrix but the calculated values are stored in a cell array where each element is a 3x3 matrix. The cell array is used in conjunction with the recursive algorithm for the solution process. The delta_y array, yo, and y arrays are cell arrays as well and the instantiation process is shown below.

```matlab
delta_y = cell(1,L);
delta_y(:) = {[0;0;0]};
matrix
yo = cell(1,L);
yo(:) = {[0;0;0]};
matrix
y = cell(1,L);
y(:) = {[0;0;0]};
matrix

Apr = cell(1,L);
Bpr = cell(1,L);
Cpr = cell(1,L);
Fpr = cell(1,L);

delta_y{N} = inverse_func(Bpr{N})*Fpr{N}...
- inverse_func(Bpr{N})*Cpr{N}*delta_y{N+1};
```
This calculates the error during each iteration. Following the matrix dimensions gives

\[ \delta_y[N] = 3 \times 3 \times 3 \times 1 - 3 \times 3 \times 3 \times 3 \times 1 = 3 \times 1, \]

which agrees.

This method of using cell arrays is efficient and compact and is highly recommended. Even the update process is compact as shown below. In one line, the original trial potential, \( y_0[N] \) is updated with the calculated error in \( \delta_y[N] \) at each division point \( N \). Remember the boundary conditions have no error associated with them as they are fixed.

\[ y[N] = y_0[N] + \delta_y[N]; \]

After each iteration, the solution needs to be checked against the error tolerance. There is effectively two main ways to accomplish this. The first method requires that the absolute value of each calculated error value for the three fundamental variables, \( n, p, \) and \( s_i \), are checked against the convergence tolerance at each division point. If every single point has converged then the solution has been reached. This is somewhat inefficient, however is accurate for the whole device. A second method checks the normalized value of the entire error vector (normalized value of the error at each division point) against the convergence tolerance. If the normalized value is small enough the solution has converged. The process for one error vector is shown below.

\[ t_{\text{empl}} = \text{abs}(\text{norm}(p_{\text{error}})/\text{norm}(p_0)); \]

\[ \text{if} \ (t_{\text{empl}} > \text{etol}) \]
\[ \quad \text{flag1} = 1; \]
\[ \text{else} \]
\[ \quad \text{flag1} = 0; \]
\[ \text{end} \]

Once all three flags have been cleared to zero (they are initially set to one), then the device has converged. The solution algorithm loops until this occurs or the iteration limit set by the user has been reached. The default is 15 iterations.

The final part of the “main” code or function simply plots the results, whether the device solution converged or not. The electric field, carrier densities, and device potential are all plotted against the spatial variable, \( x \).
The rest of the SDS program is a series of functions that are called by the "main" function or sub-functions. This is where the matrix coefficients are calculated and the partial derivatives as well. The code for one element of the B_func is shown below. It references several functions itself.

\[
\begin{align*}
b(1,1) &= \left( \frac{1}{hpr(N)} \right) \times \left( \frac{\mu_{\text{func}}(M, h, po, no, Ntot, saio, 1)}{h(M)} \right) \times \lambda_{\text{func}}(M, saio, 1) - \left( \frac{\mu_{\text{func}}(M-1, h, po, no, Ntot, saio, 1)}{h(M-1)} \right) \times \lambda_{\text{func}}(M-1, saio, -1) + \text{diffsrh}_{\text{func}}(N, po, no, Ntot, 1); \\
\end{align*}
\]

Referencing back to Chapter 3 Section 3.5, you can see that there are several functions to support the calculation of the matrix coefficient. Mobility is calculated through a function so that models of temperature and effective doping can be used, and the lambda terms and recombination terms are supported by function calls as well. This allows for efficient storage of the calculated data. We don't need the lambda value stored across the whole device because unless you are troubleshooting, there is no interest in these values.

The lambda function code is shown below. Recalling the various lambda parameters from Chapter 3, the beta term is also referenced as a function. This way beta(M) and beta(M-1) can be calculated in one function by passing in the current division point. The other important note for the lambda function is the use of the variable, \( I \). This variable is passed in so that the proper sign is used for the exponential terms. In this manner, one function can support all of the lambda terms from Chapter 3 in a compact fashion.

\[
\begin{align*}
V_t &= 0.02586; \\
\theta &= 1/V_t; \\
temp &= \beta_{\text{func}}(M, saio); \\
\end{align*}
\]

\[
\begin{align*}
\text{if}(I == 1) \\
\quad \text{if}(\text{abs}(\text{temp}) < 1E-5) \\
\quad\quad \lambda_{\text{out}} = (1/\theta) \times (1/(1-(1/2) \times \text{temp}+(1/6) \times \text{temp}^2)); \\
\quad\text{else} \\
\quad\quad \lambda_{\text{out}} = (1/\theta) \times (\text{temp} / (1 - \exp(-\text{temp}))); \\
\quad\text{end} \\
\text{elseif}(I == -1) \\
\quad \text{if}(\text{abs}(\text{temp}) < 1E-5) \\
\quad\quad \lambda_{\text{out}} = -(1/\theta) \times (1/(1+(1/2) \times \text{temp}+(1/6) \times \text{temp}^2)); \\
\quad\text{else} \\
\quad\quad \lambda_{\text{out}} = -(1/\theta) \times (\text{temp} / (1 - \exp(-\text{temp}))); \\
\quad\text{end}
\end{align*}
\]
\[
\text{lambda\_out} = (1/\theta) \ast (\text{temp} / (1 - \exp(\text{temp}))) ;
\]

One final note on this function, which also applies to the difflambda function as well, is that for special cases, the actual value of lambda needs to be approximated or some of the resulting matrix coefficient calculations will become undefined due to a division by zero. For the default device in this program, a step junction profile is used. This results in equal values of the device potential in each bulk region which will cause the output of the lambda function to go to zero or undefined. Therefore, depending upon the value of beta (which is the difference of the device potential between two division points), instead of using the direction equation and approximated equation is used.

In essence a series expansion of the lambda terms are performed so that the exponentials are approximated. The technique used here was known as Bernoulli’s function. It is also applied to the difflambda function which is used to calculate the partial derivative terms of the lambda coefficient.

Several other functions exist but won’t be discussed here. As mentioned, each block of code is commented in Appendix C. The remaining functions are similar in nature in that they support the overall solution process, many being to support the calculation of the matrix-vector coefficients, like generation and recombination. The current function will calculate the current density equations and so on.

A basic flowchart of the program is provided in Figure 5.1. This should be used in conjunction with Appendix C to better understand the flow of the program and where various calculations are made in reference to the discussions throughout the report.
Set Constants/
Numerical Parameters
(T,VT,eo,q,it,itl,L,etol)

Set Device Parameters
(Na,Nd,er,es,length,
Vl,Vr)

Define memory space
(for faster computation)

Set boundary conditions
and trial potentials
(n=n0,p=p0,sai=sai0)

Solve matrix-vector
equation using the
recursive algorithm

Update trial potentials
with calculated errors

Figure 5.1 – SDS BASIC FLOWCHART
6. Summary

Throughout this report an attempt has been made to thoroughly document the procedure used to develop a numerical solution to the simple PN junction diode. The basic device equations were transformed into numerical equations that could then be solved in any desired manner. This program implemented the implicit Newton method as the chosen solution procedure to the system of numerical equations. A recursive algorithm was used to solve the system of numerical equations which had been placed in matrix form resulting in a tri-diagonal block matrix.

The program is intended to allow for different PN junction diode characteristics to be studied by giving the user the capability to alter the default device already built into the program. Parameters like doping profile, device length, and device materials can all be changed to suit the simulation desired. Numerical parameters like convergence tolerance, mesh points, and mesh spacing can also be modified to suit the user's need or even to help solidify the understanding of the numerical parameter's role in defining the obtained solution. The solution to the default PN junction was used to compare with the theoretical solutions obtained from making certain assumptions, like the depletion approximation, when solving the standard differential equations.

The work done to develop this program lays the groundwork necessary to develop a more in-depth simulator capable of solving multiple semiconductor devices over a wide range of applications. The base program can easily be modified and updated to include further enhancements and capabilities without drastically changing the nature of the program. The program was written modularly and only portions need be modified at a time.

Applications like zener breakdown, reverser recovery time, and more can be studied with the use of this simulator, with some suitable setup to the program.
APPENDIX A

Integral Form of the Current Density Equation

As was seen in Chapter 3, Section 3.5, the current density equations were not transformed directly from the difference method as was the continuity and Poisson equations. The main reason behind this is to generate stability within the tri-diagonal block matrix to ensure proper convergence when using the implicit method. It is possible however to employ the standard difference technique on the current density equations and substitute them directly into the continuity equations, but this can lead to numerical instability if the potential changes drastically between mesh points. To examine why this occurs let us re-define the current density equation for holes, repeated from Chapters 2 and 3 in equation (A.1) and then transform the equation into standard difference form as shown in equation (A.2a and A.2b). [4]

\[ J_p = -qD_p \frac{\partial p}{\partial x} - q\mu_p p \frac{\partial \psi}{\partial x} \]  
\[ J_p(M) = -\frac{q\mu_p(M)}{\theta h(M)} \cdot [p(N+1) - p(N)] \]
\[ -\left( \frac{q\mu_p(M)}{h(M)} \cdot [\psi(N+1) - \psi(N)] \right) \cdot \left( \frac{p(N) + p(N+1)}{2} \right) \]  
\[ J_p(M-1) = -\frac{q\mu_p(M-1)}{\theta h(M-1)} \cdot [p(N) - p(N-1)] \]
\[ -\left( \frac{q\mu_p(M-1)}{h(M-1)} \cdot [\psi(N) - \psi(N-1)] \right) \cdot \left( \frac{p(N-1) + p(N)}{2} \right) \]  

The continuity equation for holes is again given in equation (A.3) in difference form. Equations (A.2a and A.2b) are then substituted into equation (A.3) and rearranged with like terms and simplified to obtain equation (A.4).
\[
\frac{1}{q} \frac{J_p(M) - J_p(M-1)}{h(N)} - G(N) + U(N) = 0 \quad (A.3)
\]

Expanding \(J_p(M)\) and \(J_p(M-1)\)

\[
J_p(M) = - \frac{q \mu_p(M)}{\theta h(M)} p(N+1) + \frac{q \mu_p(M)}{\theta h(M)} p(N) - \frac{q \mu_p(M)}{2h(M)} \psi(N+1) p(N)
\]
\[
- \frac{q \mu_p(M)}{2h(M)} \psi(N+1) p(N+1) + \frac{q \mu_p(M)}{2h(M)} \psi(N) p(N)
\]
\[
+ \frac{q \mu_p(M)}{2h(M)} \psi(N) p(N+1)
\]

and...

\[
J_p(M-1) = - \frac{q \mu_p(M-1)}{\theta h(M-1)} p(N) + \frac{q \mu_p(M-1)}{\theta h(M-1)} p(N-1) - \frac{q \mu_p(M-1)}{2h(M-1)} \psi(N) p(N-1)
\]
\[
- \frac{q \mu_p(M-1)}{2h(M-1)} \psi(N) p(N) + \frac{q \mu_p(M-1)}{2h(M-1)} \psi(N-1) p(N-1)
\]
\[
+ \frac{q \mu_p(M-1)}{2h(M-1)} \psi(N-1) p(N)
\]

Subtracting both current densities...

\[
\frac{1}{q} \cdot J_p(M) - \frac{1}{q} \cdot J_p(M-1) = - \frac{\mu_p(M)}{\theta h(M)} p(N+1) + \frac{\mu_p(M)}{\theta h(M)} p(N)
\]
\[
- \frac{\mu_p(M)}{2h(M)} \psi(N+1) p(N) - \frac{\mu_p(M)}{2h(M)} \psi(N+1) p(N+1) + \frac{\mu_p(M)}{2h(M)} \psi(N) p(N)
\]
\[
+ \frac{\mu_p(M-1)}{2h(M-1)} \psi(N) p(N+1) + \frac{\mu_p(M-1)}{\theta h(M-1)} p(N) - \frac{\mu_p(M-1)}{\theta h(M-1)} p(N-1)
\]
\[
+ \frac{\mu_p(M-1)}{2h(M-1)} \psi(N) p(N-1) + \frac{\mu_p(M-1)}{2h(M-1)} \psi(N) p(N)
\]
\[
- \frac{\mu_p(M-1)}{2h(M-1)} \psi(N-1) p(N-1) - \frac{\mu_p(M-1)}{2h(M-1)} \psi(N-1) p(N)
\]
Combining like terms...

\[
\left[ \frac{\mu_p(M-1)}{\theta h(M-1)} - \frac{\mu_p(M-1)}{2h(M-1)} \psi(N-1) + \frac{\mu_p(M-1)}{2h(M-1)} \psi(N) \right] \cdot p(N-1)
\]
\[
+ \left[ \frac{\mu_p(M-1)}{\theta h(M-1)} + \frac{\mu_p(M-1)}{2h(M-1)} \psi(N) - \frac{\mu_p(M-1)}{2h(M-1)} \psi(N-1) + \frac{\mu_p(M)}{\theta h(M)} \psi(N) \right] \cdot p(N)
\]
\[
+ \left[ -\frac{\mu_p(M)}{2h(M)} \psi(N+1) + \frac{\mu_p(M)}{2h(M)} \psi(N) \right] \cdot p(N+1)
\]

Finally, simplifying...

\[
\frac{\mu_p(M-1)}{2h(M-1)} \left[ \frac{2}{\theta} + \psi(N-1) - \psi(N) \right] \cdot p(N-1)
\]
\[
+ \frac{\mu_p(M-1)}{2h(M-1)} \left[ \frac{2}{\theta} - \psi(N-1) + \psi(N) \right] \cdot p(N)
\]
\[
+ \frac{\mu_p(M)}{2h(M)} \left[ \frac{2}{\theta} + \psi(N) - \psi(N+1) \right] \cdot p(N+1)
\]
\[
= h'(N) [G(N) - U(N)]
\]

If we assume that the mobility and mesh spacing is constant and that the potential variation between the three points, N-1, N, N+1 is linear, that is

\[
\psi(N-1) + \psi(N) = \psi(N)-\psi(N+1) = V_0 \]

then we obtain equation (A.5)
Examining equation (A.5) shows that unless the generation or recombination terms grow to proportions that deform the matrix coefficients on the left, the diagonal dominance of the equation is only lost if \( |V_D| \geq \frac{2}{\theta} \) which would allow the sub- and super-diagonal elements to dominate the equation. The middle element \( p(N) \) is not affected by the change in potential from mesh points and so the only way to combat this issue when using the standard difference form is to decrease the mesh spacing to sufficient size to lower the potential change between mesh points. However, this can lead to an extensive amount of device grid points leading to excessive computation times. [1]

This issue can be resolved by transforming the current density equation into integral form and then applying the difference technique. This allows for deviation from diagonal dominance to be negligible to the overall solution algorithm, according to Sharfetter and Gummel. [8]

To do this, we assume that in the range of \( x \), between two division points \( N \) and \( N+1 \), the electric field, mobility, and current density are constant. Then equation (A.1) can be rearranged to integrate the hole density with respect to \( x \) to obtain equation (A.6).

\[
\int_0^x \frac{\partial p}{\partial x} = \int_0^x \theta E \, p \, dx - \frac{1}{qD_p} J_p \int_0^x \partial \psi \, dx, \quad \text{where}
\]

\[
D_p = \frac{kT}{q \mu_p}, \quad E = -\frac{\partial \psi}{\partial x}, \quad \theta = \frac{q}{kT}
\]

\[
J_p(M) = -\frac{q \mu_p(M) E(M) \cdot e^{\theta E M h(M)}}{1 - e^{\theta E M h(M)}} p(N) + \frac{q \mu_p(M) E(M)}{1 - e^{\theta E M h(M)}} p(N + 1) \quad (A.6)
\]
To arrive at a more simplified solution, as shown in Chapter 3, we must do some more algebraic manipulation of equation (A.6). First, multiply the first term in equation (A.6) by

\[ \frac{-e^{-\theta E(M)h(M)}}{-e^{-\theta E(M)h(M)}} \]

which gives,

\[ J_p(M) = \frac{q\mu_p(M)E(M) \cdot e^{\theta E(M)h(M)} \cdot e^{-\theta E(M)h(M)}}{(1-e^{\theta E(M)h(M)}) \cdot (e^{-\theta E(M)h(M)})} p(N) + \frac{q\mu_p(M)E(M)}{1-e^{\theta E(M)h(M)}} p(N+1) \]

Combing the exponential terms gives,

\[ J_p(M) = \frac{q\mu_p(M)E(M)}{(1-e^{-\theta E(M)h(M)})} p(N) + \frac{q\mu_p(M)E(M)}{1-e^{\theta E(M)h(M)}} p(N+1) \text{ then,} \]

\[ J_p(M) = \frac{q}{h(M)} \left[ \hat{\lambda}_{p1}(M) p(N) + \hat{\lambda}_{p2}(M) p(N+1) \right] \]

where,

\[ \hat{\lambda}_{p1}(M) = \mu_p(M) \cdot \frac{\psi(N) - \psi(N+1)}{1-e^{-\beta(M)}} \]

\[ \hat{\lambda}_{p2}(M) = \mu_p(M) \cdot \frac{\psi(N) - \psi(N+1)}{1-e^{\beta(M)}} \]

\[ \beta(M) = \theta [\psi(N) - \psi(N+1)] \]

Equation (A.7) is exactly as shown in Chapter 3 and the same process can be used to arrive at a stable equation for the electron current density. This process is omitted here but is also the same and for both \( J_p(M-1) \) and \( J_n(M-1) \) as well. In Chapter 5, we will see that only one lambda equation function must be coded, but depending upon the variables passed into the function we can calculate the desired lambda terms. The only difference between \( \hat{\lambda}_{p1}(M) \) and \( \hat{\lambda}_{p2}(M) \) is the sign of the beta term. \( \hat{\lambda}_{n1}(M) \) and \( \hat{\lambda}_{n2}(M) \) beta terms are simply swapped. So with one equation we can evaluate any of the four lambda terms. Also when calculations for M-1 division point are required, we simply calculate the function at the current division point less 1.
\[ J_p(M) = \frac{q}{h(M)} [ \lambda_{p1}(M)p(N) + \lambda_{p2}(M)p(N+1) ] \quad \text{and} \]

\[ J_p(M-1) = \frac{q}{h(M-1)} [ \lambda_{p1}(M-1)p(N-1) + \lambda_{p2}(M-1)p(N) ] \]

where,

\[ \lambda_{p1}(M) = \mu_p(M) \frac{\psi(N) - \psi(N+1)}{1 - e^{-\beta(M)}} \],

\[ \lambda_{p2}(M) = \mu_p(M) \frac{\psi(N) - \psi(N+1)}{1 - e^{-\beta(M)}} \], \quad \text{and} \]

\[ \lambda_{p1}(M-1) = \mu_p(M-1) \frac{\psi(N-1) - \psi(N)}{1 - e^{-\beta(M-1)}} , \]

\[ \lambda_{p2}(M-1) = \mu_p(M-1) \frac{\psi(N-1) - \psi(N)}{1 - e^{-\beta(M-1)}} \], \quad \text{and} \]

\[ \beta(M) = \theta[\psi(N) - \psi(N + 1)] \]

\[ \beta(M - 1) = \theta[\psi(N + 1) - \psi(N)] \]

\[ J_n(M) = \frac{q}{h(M)} [ \lambda_{n1}(M)n(N) + \lambda_{n2}(M)n(N+1) ] \quad \text{and} \]

\[ J_n(M-1) = \frac{q}{h(M-1)} [ \lambda_{n1}(M-1)n(N-1) + \lambda_{n2}(M-1)n(N) ] \]

where,

\[ \lambda_{n1}(M) = \mu_n(M) \frac{\psi(N) - \psi(N+1)}{1 - e^{-\beta(M)}} \],

\[ \lambda_{n2}(M) = \mu_n(M) \frac{\psi(N) - \psi(N+1)}{1 - e^{-\beta(M)}} \], \quad \text{and} \]

\[ \lambda_{n1}(M-1) = \mu_n(M-1) \frac{\psi(N-1) - \psi(N)}{1 - e^{-\beta(M-1)}} , \]

\[ \lambda_{n2}(M-1) = \mu_n(M-1) \frac{\psi(N-1) - \psi(N)}{1 - e^{-\beta(M-1)}} \], \quad \text{and} \]

\[ \beta(M) = \theta[\psi(N) - \psi(N + 1)] \]

\[ \beta(M - 1) = \theta[\psi(N + 1) - \psi(N)] \]

Figure A.1 – INTEGRAL FORM OF CURRENT DENSITY EQUATIONS
Finally, if we take the new integral forms of the current densities in equation (A.7) just derived and plug them back into the continuity equation (A.3) and again assume that the mobility and mesh spacing are constant and that the potential variation between the three points, N-1, N, N+1 is linear we obtain equation (A.8).

\[
- \frac{q \mu_p V_D}{h} \left[ \frac{1}{1 - e^{-\beta}} \cdot p(N - 1) + \left( \frac{1}{1 - e^{-\beta}} + \frac{1}{1 - e^{\beta}} \right) \cdot p(N) + \frac{1}{1 - e^{\beta}} \cdot p(N + 1) \right] = h [G(N) - U(N)]
\]

(A.8)

With this form, even if \( |V_D| \leq \frac{1}{\theta} |V_D| \), deviation from the diagonal dominant condition is nonessential. Using the integral form for the current densities greatly improves the matrix property, in that the nonsingularity condition is at least approximately satisfied even for high potential difference between two neighboring division points.
APPENDIX B

Derivation Techniques for the Partial Derivative Coefficients

The following appendix will discuss the techniques used to derive the partial derivatives of the matrix-vector coefficients, $A$, $B$, and $C$. The derivation requires use of the product and chain rule from basic calculus. One example derivation is given for the hole current density and one example for the recombination terms. The rest of the derivations are left to the reader, however the partial derivatives for the coefficients $A$, $B$, and $C$ are tabulated below so that they can be compared to the relationships given in Chapter 3 Section 3.5. Doing so will help develop the understanding required to program one function that supports the various differential lambda terms that greatly reduces the number of computations. Repeating the current density equation from Appendix A,

\[
J_p(M) = \frac{q}{h(M)} \left[ \lambda_{p1}(M)p(N) + \lambda_{p2}(M)p(N + 1) \right]
\]

where,

\[
\lambda_{p1}(M) = \mu_p(M) \frac{\psi(N) - \psi(N + 1)}{1 - e^{-\beta(M)}}
\]

\[
\lambda_{p2}(M) = \mu_p(M) \frac{\psi(N) - \psi(N + 1)}{1 - e^{-\beta(M)}}
\]

\[
\beta(M) = \theta[\psi(N) - \psi(N + 1)]
\]

In partial derivative form:

\[
\frac{\partial J_p^0(M)}{\partial \psi(N)} = \frac{q}{h(M)} \left[ p(N) \frac{\partial \lambda_{p1}(M)}{\partial \psi(N)} + p(N + 1) \frac{\partial \lambda_{p2}(M)}{\partial \psi(N)} \right]
\]

where,

\[
\frac{\partial \lambda_{p1}(M)}{\partial \psi(N)} = \frac{\partial}{\partial \psi(N)} \left[ \mu_p(M)(\psi(N) - \psi(N + 1))(1 - e^{-\beta(M)})^{-1} \right]
\]

and,

\[
\frac{\partial \lambda_{p2}(M)}{\partial \psi(N)} = \frac{\partial}{\partial \psi(N)} \left[ \mu_p(M)(\psi(N) - \psi(N + 1))(1 - e^{\beta(M)})^{-1} \right]
\]
To take the partial derivative of the lambda term with respect to the potential at mesh point N we can start by first applying the product rule since the lambda terms is not a straightforward function of the potential.

\[
\frac{\partial \lambda_{pl}(M)}{\partial \psi(N)} = \mu_p(M) \frac{\partial}{\partial \psi(N)} \left[ f(\psi(N)) g(\psi(N)) \right]
\]

where,

\[
f(\psi(N)) = \psi(N) - \psi(N + 1)
\]

\[
g(\psi(N)) = \left(1 - e^{-\beta(M)}\right)^{-1}
\]

Using the product rule:

\[
\frac{\partial \lambda_{pl}(M)}{\partial \psi(N)} = \mu_p(M) \left\{ f(\psi(N)) \frac{\partial}{\partial \psi(N)} \left[ g(\psi(N)) \right] + g(\psi(N)) \frac{\partial}{\partial \psi(N)} \left[ f(\psi(N)) \right] \right\}
\]

However, before going further the partial derivative of the newly defined g term requires use of the chain rule, so that is evaluated first.

to take \(\frac{\partial}{\partial \psi(N)} \left[ g(\psi(N)) \right] \), let \(g(\psi(N))\) be a function of \(u\) then,

\[
g(\psi(N)) = u^{-1} \quad \text{and} \quad \frac{\partial g(\psi(N))}{\partial u} = -u^{-2} \quad \text{where},
\]

\[
u = 1 - e^{-\beta(M)} \quad \text{and} \quad \frac{\partial u}{\partial \psi(N)} = \theta e^{-\beta(M)}
\]

Using the chain rule:

\[
\frac{\partial g(\psi(N))}{\partial \psi(N)} = \frac{\partial g(\psi(N))}{\partial u} \cdot \frac{\partial u}{\partial \psi(N)} = -\left(1 - e^{-\beta(M)}\right)^{-2} \left(\theta e^{-\beta(M)}\right)
\]

finally,

\[
\frac{\partial \lambda_{pl}(M)}{\partial \psi(N)} = \mu_p(M) \left\{ -(\psi(N) - \psi(N + 1))(1 - e^{-\beta(M)})^{-2} \left(\theta e^{-\beta(M)}\right) + \left(1 - e^{-\beta(M)}\right)^{-1} \right\}
\]

\[
\frac{\partial \lambda_{pl}(M)}{\partial \psi(N)} = \frac{\mu_p(M)}{(1 - e^{-\beta(M)})} \left[ 1 - \frac{\beta(M)e^{-\beta(M)}}{(1 - e^{-\beta(M)})} \right]
\]
The same procedure is followed for the lambda term $2$.

\[
\frac{\partial \lambda_{p2}(M)}{\partial \psi(N)} = \mu_p(M) \frac{\partial}{\partial \psi(N)} \left[ f(\psi(N))g(\psi(N)) \right]
\]

where,

\[
f(\psi(N)) = \psi(N) - \psi(N + 1)
g(\psi(N)) = \left(1 - e^{\beta(M)}\right)^{-1}
\]

Using the product rule:

\[
\frac{\partial \lambda_{p2}(M)}{\partial \psi(N)} = \mu_p(M) \left\{ f(\psi(N)) \frac{\partial}{\partial \psi(N)} [g(\psi(N))] + g(\psi(N)) \frac{\partial}{\partial \psi(N)} [f(\psi(N))] \right\}
\]

However, before going further the partial derivative of the newly defined $g$ term requires use of the chain rule, so that is evaluated first.

To take \(\frac{\partial}{\partial \psi(N)} [g(\psi(N))]\), let \(g(\psi(N))\) be a function of \(u\) then,

\[
g(\psi(N)) = u^{-1} \quad \text{and} \quad \frac{\partial g(\psi(N))}{\partial u} = -u^{-2} \quad \text{where},
\]

\[
u = 1 - e^{\beta(M)} \quad \text{and} \quad \frac{\partial u}{\partial \psi(N)} = -\theta e^{\beta(M)}
\]

Using the chain rule:

\[
\frac{\partial g(\psi(N))}{\partial \psi(N)} = \frac{\partial g(\psi(N))}{\partial u} \cdot \frac{\partial u}{\partial \psi(N)} = -\left(1 - e^{\beta(M)}\right)^{-2} \left(-\theta e^{\beta(M)}\right)
\]

Finally,

\[
\frac{\partial \lambda_{p2}(M)}{\partial \psi(N)} = \mu_p(M) \left\{ (\psi(N) - \psi(N + 1)) \left(1 - e^{\beta(M)}\right)^{-2} \left(\theta e^{\beta(M)}\right) + \left(1 - e^{\beta(M)}\right)^{-1} \right\}
\]

\[
\frac{\partial \lambda_{p2}(M)}{\partial \psi(N)} = \frac{\mu_p(M)}{\left(1 - e^{\beta(M)}\right)} \left[ 1 + \frac{\beta(M)e^{\beta(M)}}{\left(1 - e^{\beta(M)}\right)} \right]
\]
Essentially the same procedure is used for the recombination and generation partial derivatives. An example of the recombination derivation is given below.

\[
U(N) = \frac{p(N)n(N) - n_i^2}{\tau_n (p(N) + n_i) + \tau_p (n(N) + n_i)}
\]

\[
\frac{\partial U(N)}{\partial p(N)} = \frac{\partial}{\partial p(N)} \left[ \left( p(N)n(N) - n_i^2 \right) \cdot \left( \tau_n (p(N) + n_i) + \tau_p (n(N) + n_i) \right)^{-1} \right]
\]

\[
\frac{\partial U(N)}{\partial n(N)} = \frac{\partial}{\partial n(N)} \left[ \left( p(N)n(N) - n_i^2 \right) \cdot \left( \tau_n (p(N) + n_i) + \tau_p (n(N) + n_i) \right)^{-1} \right]
\]

Using the product and chain rule:

\[
\frac{\partial}{\partial p(N)} \left[ \left( p(N)n(N) - n_i^2 \right) \cdot \left( \tau_n (p(N) + n_i) + \tau_p (n(N) + n_i) \right)^{-1} \right]
\]

\[
= \left( p(N)n(N) - n_i^2 \right) \cdot \left( -\tau_n \left( \tau_n (p(N) + n_i) + \tau_p (n(N) + n_i) \right)^{-2} \right)...
\]

\[
+ n(N) \left( \tau_n (p(N) + n_i) + \tau_p (n(N) + n_i) \right)^{-1}
\]

so,

\[
\frac{\partial U(N)}{\partial p(N)} = \frac{n(N) - \tau_n U(N)}{\tau_n (p(N) + n_i) + \tau_p (n(N) + n_i)}
\]
APPENDIX C

Semiconductor Device Simulator Implemented

%Title: 1D PN Step Junction Numerical Device Simulator
%Description: Using pre-defined material parameters for Silicon, a device can be described numerically along a grid at which the physical differential equations are solved for the three basic device parameters, potential, and carrier concentrations, herein referred to as sai, n, and p.
%Written By: Shawn Pace
%Date: 2011 UCD Thesis Project
%Contributions: Many thanks to Dr. Hamid Fardi whose knowledge and previous solution program fardev, without which, would have made this nearly impossible.

clear all; %Clears all previously stored variable information close all; %Closes all previously open plots clc; %Clears the command window to blank TSTART = tic; %Starts a timer to record solution time format long eng; %Formats the data to engineering mode

%Input Parameters
Nd = 1E15; %Donor Concentration
Na = 1E15; %Acceptor Concentration
T = 300; %Temperature in Kelvin
ni = nieff_func(T); %Intrinsic Carrier Concentration
Vt = 0.02586; %Thermal Voltage
eo = 8.854E-14; %Relative Permittivity Constant
er = 11.7; %Relative Permittivity of Silicon
es = eo * er; %Permittivity of Silicon
length = 10E-6; %Device C
q = 1.602E-19; %Electron Charge
theta = 1/Vt; %Boltzmann Factor
alpha_p0 = 2.25E7; %Generation defined constants; see Chapter 5
E_p0 = 3.26E6; %Generation defined constants; see Chapter 5
alpha_n0 = 3.80E6; %Generation defined constants; see Chapter 5
E_n0 = 1.75E6; %Generation defined constants; see Chapter 5
m=1;

%Initial applied voltage:
Vl = 0; %Applied voltage to left contact
Vr = 0; %Applied voltage to right contact

%Numerical Parameters:
it = 0; %Iteration number initialized to zero
itl = 15; %Max iteration limit to find solution within %a given tolerance
L = 1001; %Numerical Device Solution Points
etol = 1E-6; %Convergence Tolerance; calculated and %normalized error must be less than etol to %converge

delta = (length/(L-1))*10^2;

for (N=2:1:L)
    x(N) = x(N-1)+delta;
end

for (M=1:1:L-1)
    h(M) = x(M+1)-x(M);
end

for (N=2:1:L-1)
    M=N;
    hpr(N) = (1/2)*(h(M-1) + h(M));
end

x1(1:L)=0; %Not currently used
gamma(1:L)=0; %Device doping profile
Ntot(1:L)=0;
saio(1:L)=0;
sai(1:L)=0;
po(1:L)=0;
no(1:L)=0;
E(1:L)=0;

%Device doping total concentration (Nd + Na)
%Device potential previous solution
%Device potential final solution
%Device hole concentration
%Device electron concentration
%Device energy field

delta_y = cell(1,L);
delta_y(:) = {[0;0;0]};

yo = cell(1,L);
yo(:) = {[0;0;0]};

y = cell(1,L);
y(:) = {[0;0;0]};

%Defines an empty cell from 1 to L
%Initializes each cell to a 3x3 empty matrix
%Initializes each cell to a 1x3 empty matrix
%Initializes each cell to a 1x3 empty matrix

%y and yo above are used to store the updated values after adding
%the error for each iteration. They store sai, n, and p.

%Boundary Conditions:
The solution of the device equations require a two-point boundary
condition and below are the boundary conditions for the device
length, doping profile, carrier concentrations, and built-in
potential
x(1) = 0;
x(L) = length*10^2;
gamma(1)= -Na;
gamma(L)= Nd;
Ntot(1)= Na;
Ntot(L)= Nd;

po(1)=( -gamma(1)/2)*(1+sqrt(1+((2*ni)/gamma(1))^2));
no(1)=ni^2/po(1);
no(L)=(gamma(L)/2)*(1+sqrt(1+((2*ni)/gamma(L))^2));
po(L)=ni^2/no(L);

phi_p = -Vt*log(po(1)/(ni));
saio(L) = Vr + Vt*log(no(L)/(ni)) - phi_p;
saio(1) = Vl;
yo{1} = [po(1);no(1);saio(1)];
yo{L} = [po(L);no(L);saio(L)];

%Numerical Solution

%Trial Values:
Before the device equations can be calculated, trial values for the
carrier concentrations and device potential must be provided across
the whole device.

for(N=2:1:L-1) %#ok<*NO4LP>
if(x(N) <= (length*10^2)/2)
gamma(N) = -Na;
Ntot(N) = Na;
else
  gamma(N) = Nd;
  Ntot(N) = Nd;
end

end
end

if (sign(gamma(N)) == -1)
    po(N) = -gamma(N);
    no(N) = -(ni^2/gamma(N));
else
    po(N) = ni^2/gamma(N);
    no(N) = gamma(N);
end

saio(N) = Vt*log(abs(gamma(N))/ni)*sign(gamma(N)) - phi_p;
if (saio(N) < 1E-9)
    saio(N) = 0;
end

yo{N} = [po(N);no(N);saio(N)];

% Figure 1 plots the trial potential calculated above. This should be
% a step profile for a step junction.
figure(1)
plot(x.*10^-2,saio)
xlim([0 length])
ylim([0 1.0])
title('saio - initial trial potential', 'FontSize',16)

% This pre-defined memory space is to store the error calculated
% after each iteration for device potential and carrier
% concentrations. This was mainly used for troubleshooting purposes
% during development of the program, however these values can be
% plotted to give an understanding of how the error changes during
% each iteration.
p_error(1:L) = 0;
n_error(1:L) = 0;
v_error(1:L) = 0;

% This pre-defined memory space is defining an empty cell from 1 to
% L; it is not necessary to intialize each cell to 3x3 empty matrices
% as above, because each cell can hold whatever the programmer
% chooses. These cells hold the 3x3 matrices used to calculate the
% error for each device point in the recursive equation that is
% employed.
Apr = cell(1,L);
Bpr = cell(1,L);
Cpr = cell(1,L);
Fpr = cell(1,L);

% Since the boundary conditions are pre-defined, there is no error
% associated with grid points 1 and L, so only 2 through L-1 must be
% solved; please refer to the appendices in the Kurata book
N=2;
Bpr{2} = b_func(N,h,hpr,po,no,Ntot,saio);
Cpr{2} = c_func(N,h,hpr,po,no,Ntot,saio);
\[
\begin{align*}
\text{Fpr}\{2\} &= f_\text{func}(N,h,hpr,po,\text{no},N\text{tot},\gamma,\text{saio}); \\
\text{for (N=3:1:L-1)} \\
\quad \text{Apr}\{N\} &= a_\text{func}(N,h,hpr,po,\text{no},N\text{tot},\text{saio}); \\
\quad \text{Bpr}\{N\} &= b_\text{func}(N,h,hpr,po,\text{no},N\text{tot},\text{saio}) - \text{Apr}\{N\}*... \\
\quad \text{Cpr}\{N\} &= c_\text{func}(N,h,hpr,po,\text{no},N\text{tot},\text{saio}); \\
\quad \text{Fpr}\{N\} &= f_\text{func}(N,h,hpr,po,\text{no},N\text{tot},\gamma,\text{saio}) - \text{Apr}\{N\}*... \\
\quad &\text{end}
\end{align*}
\]

Once \text{Apr} (A prime), \text{Bpr}, \text{Cpr}, and \text{Fpr} are calculated across the whole device, the recursive equation can be used to calculate the error at each device point. This error is then added to the trial potential and if the convergence criteria are satisfied, the solution has been found.

\[
\begin{align*}
\text{for (N=L-1:-1:2)} \\
\quad \text{delta}_\text{y}\{N\} &= \text{inverse}_\text{func} (\text{Bpr}\{N\})*\text{Fpr}\{N\}... \\
\quad &\text{end}
\end{align*}
\]

Error for each variable is extracted for review if necessary

\[
\begin{align*}
\text{for (N=1:1:L)} \\
\quad \text{p_error}\{N\} &= \text{delta}_\text{y}\{N\}(1); \\
\quad \text{n_error}\{N\} &= \text{delta}_\text{y}\{N\}(2); \\
\quad \text{v_error}\{N\} &= \text{delta}_\text{y}\{N\}(3); \\
\end{align*}
\]

This section adds the calculated error to the trial values; recall that \text{y} is a cell matrix with each cell containing a 1x3 matrix and since \text{delta}_\text{y} is of the same type they can easily be added together. Then each updated value is extracted into its own vector for convergence determination and also replaced in the trial value cell matrix \text{yo}.

\[
\begin{align*}
\text{for (N=1:1:L)} \\
\quad \text{y}\{N\} &= \text{yo}\{N\} + \text{delta}_\text{y}\{N\}; \\
\quad \text{po}(N) &= \text{y}\{N\}(1); \\
\quad \text{no}(N) &= \text{y}\{N\}(2); \\
\quad \text{saio}(N) &= \text{y}\{N\}(3); \\
\quad \text{if (saio}(N) < 0) \\
\quad \quad \text{saio}(N) &= 0; \\
\quad \text{end} \\
\quad \text{yo}\{N\} &= [\text{po}(N); \text{no}(N); \text{saio}(N)];
\end{align*}
\]
%Variable temp1, 2, and 3 contain the normalized calculation for the updated (trial + error) value which is then compared to the convergence tolerance, etol and a flag is set to signify convergence or not.

temp1 = abs(norm(p_error)/norm(po));

if (temp1 > etol)
    flag1 = 1;
else
    flag1 = 0;
end

temp2 = abs(norm(n_error)/norm(no));

if (temp2 > etol)
    flag2 = 1;
else
    flag2 = 0;
end

temp3 = abs(norm(v_error)/norm(saio));

if (temp3 > etol)
    flag3 = 1;
else
    flag3 = 0;
end

%Whether the solution was found, converged, or not, the iteration number is still increased by 1 to signify one more iteration was completed.
it = it + 1;

%This while loop repeats the solution algorithm above until the iteration limit is reached or the solution found.
while(((flag1 || flag2 || flag3 == 1) && (it < itl))

N=2;
Bpr{2} = b_func(N,h,hpr,po,no,Ntot,saio);
Cpr{2} = c_func(N,h,hpr,po,no,Ntot,saio);
Fpr{2} = f_func(N,h,hpr,po,no,Ntot,gamma,saio);

for(N=3:1:L-1)
    Bpr{N} = b_func(N,h,hpr,po,no,Ntot,saio)...
    - a_func(N,h,hpr,po,no,Ntot,saio)*...
    inverse_func(Bpr{N-1})*Cpr{N-1};
    Cpr{N} = c_func(N,h,hpr,po,no,Ntot,saio);
\[
Fpr\{N\} = f_{\text{func}}(N, h, hpr, po, no, N_{\text{tot}}, \gamma, saio) \ldots \\
- a_{\text{func}}(N, h, hpr, po, no, N_{\text{tot}}, saio) \ldots \\
\text{inverse\_func}(Bpr\{N-1\}) \ast Fpr\{N-1\};
\]

\[
\text{end}
\]

\[
\text{for} (N=L-1:-1:2) \\
\quad \delta y\{N\} = \text{inverse\_func}(Bpr\{N\}) \ast Fpr\{N\} \ldots \\
\quad - \text{inverse\_func}(Bpr\{N\}) \ast Cpr\{N\} \ast \delta y\{N+1\};
\]

\[
\text{end}
\]

\[
\text{for} (N=1:1:L) \\
\quad p\_error\{N\} = \delta y\{N\}(1); \\
\quad n\_error\{N\} = \delta y\{N\}(2); \\
\quad v\_error\{N\} = \delta y\{N\}(3);
\]

\[
\text{end}
\]

\[
\text{for} (N=1:1:L) \\
\quad y\{N\} = yo\{N\} + \delta y\{N\}; \\
\quad po\{N\} = y\{N\}(1); \\
\quad no\{N\} = y\{N\}(2); \\
\quad saio\{N\} = y\{N\}(3); \\
\quad \text{if} (saio\{N\} < 0) \\
\quad \quad saio\{N\} = 0;
\]

\[
\text{end}
\]

\[
yo\{N\} = [po\{N\}; no\{N\}; saio\{N\}];
\]

\[
\text{end}
\]

\[
\text{temp1} = \text{abs}(\text{norm}(p\_error)/\text{norm}(po));
\]

\[
\text{if} (\text{temp1} > \text{etol}) \\
\quad \text{flag1} = 1;
\]

\[
\text{else} \\
\quad \text{flag1} = 0;
\]

\[
\text{end}
\]

\[
\text{temp2} = \text{abs}(\text{norm}(n\_error)/\text{norm}(no));
\]

\[
\text{if} (\text{temp2} > \text{etol}) \\
\quad \text{flag2} = 1;
\]

\[
\text{else} \\
\quad \text{flag2} = 0;
\]

\[
\text{end}
\]

\[
\text{temp3} = \text{abs}(\text{norm}(v\_error)/\text{norm}(saio));
\]

\[
\text{if} (\text{temp3} > \text{etol}) \\
\quad \text{flag3} = 1;
\]
else
    flag3 = 0;
end

it = it + 1;
end

% This loop calculates the energy field at each device point once the final device potential is found.
for (M=1:1:L-1)
    E(M) = energy_func(M,h,saio);
end

% Figures 2-4 plot the zero-applied bias device potential and carrier concentrations for a step junction PN device.
figure(2)
plot(x.*10^-2,po,x.*10^-2,no)
xlim([0 length])
% ylim([0 2E17])
title('carrier densities - equilibrium','FontSize',16)
figure(3)
plot(x.*10^-2,saio)
xlim([0 length])
title('potential - equilibrium','FontSize',16)
ylim([0 1.0])
figure(4)
plot(x.*10^-2,E)
xlim([0 length])
title('Energy - euilibrium','FontSize',16)

time = toc(TSTART); % Stops timer for the equilibrium solution
time
TSTART = tic; % Starts a timer for applied bias solution
time

% Applied external voltage (forward bias)
Vl = 0.5; % Applies a new external bias of 0.5 volts
Vr = 0.0; % Right contact is still grounded
V1 = 0.0; % Previous applied external voltage
% Re-define Trial Values for applied external voltage; this equation
% is a simple proportional relation to move the new "trial" values
% with respect to the old applied voltage. The equilibrium solution
% is used as the trial values for the applied voltage solution.
for (N=2:1:L) %#ok<*N04LP>
    saio(N) = (1-((Vl-Vl)/(saio(L)-saio(1))))*saio(N)
    +((Vl-Vl)/(saio(L)-saio(1)))*saio(L);
    yo{N} = [po(N);no(N);saio(N)];
end

% Re-define Boundary Conditions for applied external voltage; The
% doping profile hasn't changed, and so therefore only the device
% potential boundary conditions need updating.
saio(1) = V1;
saio(L) = Vr + Vt*log(no(L)/(ni)) - phi_p;
yo{1} = [po(1);no(1);saio(1)];
yo{L} = [po(L);no(L);saio(L)];

% This was used during development to extract the values to a text
% file
% for(N=1:1:L)
%   out (N, 1) = x(N);
%   out(N, 2) = po(N);
%   out(N, 3) = no(N);
%   out(N, 4) = saio(N);
% end
%dlmwrite('output.txt', out);

% This plots the updated device trial potential; it was mainly used
% during development to ensure the starting trial potential was
% correct.
figure(5)
plot(x.*10^-2, saio)
xlim([0 length])
title('saio - applied voltage trial potential', 'FontSize',16)
% ylim([-1.2 1.2])

% Reset iteration number
it = 0; % Iteration number re-initialized to zero
itl = 15; % Max iteration limit to find solution within given
% tolerance; this can be changed if solution is not
% converging in the allotted iterations
%The remaining portion of this code, down through the while loop is
%exactly as above, and is performing the recursive algorithm on the
%tri-diagonal matrix to calculate the error at each grid point and
%continuing until the solution has converged or limit is reached.
N=2;
Bpr{2} = b_func(N,h,hpr,po,no,Ntot,saio);
Cpr{2} = c_func(N,h,hpr,po,no,Ntot,saio);
Fpr{2} = f_func(N,h,hpr,po,no,Ntot,gamma,saio);

for (N=3:1:L-1)
    Apr{N} = a_func(N,h,hpr,po,no,Ntot,saio);
    Bpr{N} = b_func(N,h,hpr,po,no,Ntot,saio) - Apr{N} * inverse_func(Bpr{N-1})*Cpr{N-1};
    Cpr{N} = c_func(N,h,hpr,po,no,Ntot,saio);
    Fpr{N} = f_func(N,h,hpr,po,no,Ntot,gamma,saio) - Apr{N} * inverse_func(Bpr{N-1})*Fpr{N-1};
end

for (N=L-1:-1:2)
    delta_y{N} = inverse_func(Bpr{N})*Fpr{N} - inverse_func(Bpr{N})*Cpr{N}*delta_y{N+1};
end

for (N=1:1:L)
    p_error(N) = delta_y{N}(1);
    n_error(N) = delta_y{N}(2);
    v_error(N) = delta_y{N}(3);
end

for (N=1:1:L)
    y{N} = yo{N} + 1.0*delta_y{N};
    po(N) = y{N}(1);
    no(N) = y{N}(2);
    saio(N) = y{N}(3);
    yo{N} = [po(N);no(N);saio(N)];
end

temp1 = abs(norm(p_error)/norm(po));

if (temp1 > etol)
    flag1 = 1;
else
    flag1 = 0;
end

temp2 = abs(norm(n_error)/norm(no));
if (temp2 > etol)
    flag2 = 1;
else
    flag2 = 0;
end

temp3 = abs(norm(v_error)/norm(s aio));
if (temp3 > etol)
    flag3 = 1;
else
    flag3 = 0;
end

it = it + 1;

while(((flag1 || flag2 || flag3) == 1) && (it < itl))
    N=2;
    Bpr{2} = b_func(N,h,hpr,po,no,Ntot,s aio);
    Cpr{2} = c_func(N,h,hpr,po,no,Ntot,s aio);
    Fpr{2} = f_func(N,h,hpr,po,no,Ntot,gamma,s aio);

    for (N=3:1:L-1)
        Bpr{N} = b_func(N,h,hpr,po,no,Ntot,s aio)...
            - a_func(N,h,hpr,po,no,Ntot,s aio)...
            inverse_func(Bpr{N-1})*Cpr{N-1};
        Cpr{N} = c_func(N,h,hpr,po,no,Ntot,s aio);
        Fpr{N} = f_func(N,h,hpr,po,no,Ntot,gamma,s aio)...
            - a_func(N,h,hpr,po,no,Ntot,s aio)...
            inverse_func(Bpr{N-1})*Fpr{N-1};

    end

    for (N=L-1:-1:2)
        delta_y{N} = inverse_func(Bpr{N})*Fpr{N}...
            - inverse_func(Bpr{N})*Cpr{N}*delta_y{N+1};

    end

    for (N=1:1:L)
        p_error(N) = delta_y{N}(1);
        n_error(N) = delta_y{N}(2);
        v_error(N) = delta_y{N}(3);
    end

    for (N=1:1:L)
        y{N} = yo{N} + 1.0*delta_y{N};
\[ po(N) = y[N](1); \]
\[ no(N) = y[N](2); \]
\[ saio(N) = y[N](3); \]
\[ yo{N} = [po(N);no(N);saio(N)]; \]
\end

\[ temp1 = \text{abs}(\text{norm}(p\_error)/\text{norm}(po)); \]
\[ \text{if} \ (temp1 > \text{etol}) \]
\[ \quad \text{flag1} = 1; \]
\[ \text{else} \]
\[ \quad \text{flag1} = 0; \]
\[ \text{end} \]

\[ temp2 = \text{abs}(\text{norm}(n\_error)/\text{norm}(no)); \]
\[ \text{if} \ (temp2 > \text{etol}) \]
\[ \quad \text{flag2} = 1; \]
\[ \text{else} \]
\[ \quad \text{flag2} = 0; \]
\[ \text{end} \]

\[ temp3 = \text{abs}(\text{norm}(v\_error)/\text{norm}(saio)); \]
\[ \text{if} \ (temp3 > \text{etol}) \]
\[ \quad \text{flag3} = 1; \]
\[ \text{else} \]
\[ \quad \text{flag3} = 0; \]
\[ \text{end} \]

\[ it = it + 1; \]
\end

\%Defined memory space for the SRH terms used to calculate current\nU(1:L)=0;

\%Energy field is re-calculated based on applied voltage\nfor (M=1:1:L-1)\n\quad E(M) = energy\_func(M,h,saio);\n\end

\%SRH Recombination is calculated across the device in order to\n\%calculate the device current for the give applied voltage.\nfor (N=2:1:L-1)\n\quad U(N) = srh\_func(N,po,no,Ntot);\n\end
Based on the device equations, the current can be calculated by numerically integrating the SRH Recombination values at each grid point. A simple trapezoidal integration technique was used. This is only useful for forward bias situations. Generation terms need to be included for reverse bias current draw calculations.

\[ J_{tot} = q \left( \frac{\delta}{2} \left( U(1) + U(L) \right) + \delta \text{sum}(U(2:L-1)) \right) \]

Finally figures 6-8 plots the device potential and carrier concentrations for the given applied bias.

figure(6)
plot(x.*10^-2,saio)
xlim([0 length])
ylim([0 1.0])
title('potential - applied voltage', 'FontSize', 16)

figure(7)
plot(x.*10^-2,E)
xlim([0 length])
title('Energy - applied voltage', 'FontSize', 16)

figure(8)
plot(x.*10^-2,po,x.*10^-2,no)
xlim([0 length])
% ylim([0 2E17])
title('carrier densities - applied voltage', 'FontSize', 16)

timel = toc(TSTART);  
% Stops the clock for the applied bias solution time
function mu_out = mu_func(M,h,po,no,Ntot,saio,I)
%This function returns the carrier concentration mobility based on
%the index, I and grid point M, passed into the function.
%Currently this function only returns the low E-field mobility as a
%constant and for the hole mobility an I of 1 is used and for
%electron mobility an I of -1 is used. The rest of the equations and
%constants below are currently not used, but could be to develop the
%mobility based on total concentration and E-field.
%M - Device grid point of interest
%h - Grid mesh spacing
%po - Hole concentration
%no - Electron concentration
%Ntot - Total concentration
%saio - Device potential
%I - Index used to define which mobility, hole or electron, to
%output

% N=M;

%Mobility Parameters
Nrefp = 6.3E16; % cm^-3
alphap = 0.76; %fitting parameter
betap = 1; %fitting parameter
mu_maxp = 495; % cm^2/V-sec
mu_minp = 47.7; % cm^2/V-sec
Ecp = 1.95E4; % V/cm

Nrefn = 8.5E16; % cm^-3
alphan = 0.72; %fitting parameter
betan = 2; %fitting parameter
mu_maxn = 1330; % cm^2/V-sec
mu_minn = 65; % cm^2/V-sec
Ecn = 8E3; % V/cm

% fp = ((po(N)*no(N))^((1/2)/(2.04*Nrefp)))^alphap;
% fn = ((po(N)*no(N))^((1/2)/(2.04*Nrefn)))^alphan;

fp = 0;
fn = 0;

% if(I == 1)
% mu_out =(((mu_maxp-u_minp))/((1+(Ntot(M)/Nrefp)^alphap+fp))... 
% +mu_minp)*(1/(1+(energy_func(M,h,saio)/Ecp)^betap)^(1/betap));
% elseif(I == -1)
% mu_out =(((mu_maxn-u_minn))/((1+(Ntot(M)/Nrefn)^alphan+fn))... 
% +mu_minn)*(1/(1+(energy_func(M,h,saio)/Ecn)^betan)^(1/betan));
% end
if (I == 1)
    mu_out = 495;
elseif (I == -1)
    mu_out = 1330;
end

function nieff_out = nieff_func(T)
%Returns the effective intrinsic carrier concentration based on the 
%provided device temperature.

k = 8.62E-5;
EG300 = 1.08;
EG_alpha = 4.73E-4;
EG_beta = 636;
NC300 = 2.8E19;
NV300 = 1.04E19;

EG = EG300 + EG_alpha*((300^2/(300+EG_beta)) - (T^2/(T+EG_beta)))
NC = (T/300)^(3/2)*NC300;
NV = (T/300)^(3/2)*NV300;

assignin('base','EG',EG);
assignin('base','NC',NC);
assignin('base','NV',NV);

nieff_out = sqrt(NC*NV)*exp(-EG/(2*k*T));
function a = a_func(N,h,hpr,po,no,Ntot,saio)
%This function calculates a 3x3 matrix based on the grid point
%index, M, passed into the function.
%This matrix, along with the b,c, and f_func makes up the
%Tri-diagonal matrix for each device point a when combined can be
%solved for the error using the recursive tri-diagonal solution
%algorithm.
% N - Device grid point of interest
% h - Grid mesh spacing
% hpr - Auxiliary grid spacing; same as h for uniform device
% po - Hole concentration
% no - Electron concentration
% Ntot - Total concentration
% saio - Device potential

%The main 3x3 matrices (a,b,c,f) are calculated upon the main grid
points N, but the sub-equations are calculated on the auxiliary grid
points. This is why M is set equal to N.
M=N;

a(1,1)= -(mu_func(M,h,po,no,Ntot,saio,1)/hpr(N))...
         *(lambda_func(M-1,saio,1)/h(M-1));

a(1,2)= 0;

a(1,3)= -(1/(hpr(N)*h(M-1)))*(mu_func(M-1,h,po,no,Ntot,saio,1)*...
           po(N-1)*difflambda_func(M-1,saio,1)+ mu_func(M-
           1,h,po,no,Ntot,saio,1)*...po(N)*difflambda_func(M-1,saio,-1));

a(2,1)= 0;

a(2,2)= -(mu_func(M,h,po,no,Ntot,saio,-1)/hpr(N))...
         *(lambda_func(M-1,saio,-1)/h(M-1));

a(2,3)= -(1/(hpr(N)*h(M-1)))*(mu_func(M-1,h,po,no,Ntot,saio,-
           1)*no(N-1)... *difflambda_func(M-1,saio,-1)+ mu_func(M-1,h,po,no,Ntot,saio,-
           1)... *no(N)*difflambda_func(M-1,saio,1));

a(3,1)= 0;

a(3,2)= 0;

a(3,3)= 1/(h(M-1)*hpr(N));
end
function b = b_func(N,h,hpr,po,no,Ntot,saio)
%This function calculates a 3x3 matrix based on the grid point
index, M, passed into the function.
%This matrix, along with the a,c, and f_func makes up the tri-
%diagonal matrix for each device point a when combined can be solved
%for the error using the recursive tri-diagonal solution algorithm.
% N - Device grid point of interest
% h - Grid mesh spacing
% hpr - Auxiliary grid spacing; same as h for uniform device
% po - Hole concentration
% no - Electron concentration
% Ntot - Total concentration
% saio - Device potential

%The main 3x3 matrices (a,b,c,f) are calculated upon the main grid
points N, but the sub-equations are calculated on the auxiliary grid
points. This is why M is set equal to N.
M=N;

eo = 8.854E-14; %Permittivity Constant
er = 11.7; %Relative Permittivity of Silicon
es = eo * er; %Permittivity of Silicon
q = 1.602E-19; %Electron Charge

b(1,1)= ( 1/hpr(N) )*((mu_func(M,h,po,no,Ntot,saio,1)/h(M))...
*lambda_func(M,saio,1)-(mu_func(M-1,h,po,no,Ntot,saio,1)/h(M-1))...
*lambda_func(M-1,saio,-1))+ diffrh_func(N,po,no,Ntot,1);

b(1,2)= diffrh_func(N,po,no,Ntot,-1);

b(1,3)= ( 1/hpr(N) )*((mu_func(M,h,po,no,Ntot,saio,1)/h(M))...
*(po(N)*difflambda_func(M,saio,1)+po(N+1)...
*difflambda_func(M,saio,-1))...%
(mu_func(M-1,h,po,no,Ntot,saio,1)/h(M-1))* (po(N-1)...
*difflambda_func(M-1,saio,1)+po(N)*difflambda_func(M-1,saio,-1)));

b(2,1)= -diffrh_func(N,po,no,Ntot,1);

b(2,2)= ( 1/hpr(N) )*((mu_func(M,h,po,no,Ntot,saio,-1)/h(M))...
*lambda_func(M,saio,-1)-(mu_func(M-1,h,po,no,Ntot,saio,-1)/h(M-1))...
*lambda_func(M-1,saio,1))-diffrh_func(N,po,no,Ntot,-1);

b(2,3)= ( 1/hpr(N) )*((mu_func(M,h,po,no,Ntot,saio,-1)/h(M))...
*(no(N)*difflambda_func(M,saio,-1)+no(N+1)...
*difflambda_func(M,saio,1))...%
(mu_func(M-1,h,po,no,Ntot,saio,-1)/h(M-1))* (no(N-1)...

101
\*\text{diff\lambda\_func}(M-1, saio, -1) + \text{no}(N) \* \text{diff\lambda\_func}(M-1, saio, 1)));

b(3,1) = q/es;

b(3,2) = -q/es;

b(3,3) = -(1/hpr(N)) \* (1/h(M-1) + 1/h(M));

\text{end}
function c = c_func(N,h,hpr,po,no,Ntot,saio)
%This function calculates a 3x3 matrix based on the grid point
%index, M, passed into the function.
%This matrix, along with the a,b, and f_func makes up the tri-
%diagonal matrix for each device point a when combined can be solved 
%for the error using the recursive tri-diagonal solution algorithm.
% N - Device grid point of interest
% h - Grid mesh spacing
% hpr - Auxiliary grid spacing; same as h for uniform device
% po - Hole concentration
% no - Electron concentration
% Ntot - Total concentration
% saio - Device potential

%The main 3x3 matrices (a,b,c,f) are calculated upon the main grid
%points N, but the sub-equations are calculated on the auxiliary
%grid points. This is why M is set equal to N.
M=N;

c(1,1)= (mu_func(M,h,po,no,Ntot,saio,1)/hpr(N))... 
    *(lambda_func(M,saio,-1)/h(M));

c(1,2)= 0;

c(1,3)= -(1/(hpr(N)*h(M)))*(mu_func(M,h,po,no,Ntot,saio,1) ... 
    *po(N)*difflambda_func(M,saio,1)+mu_func(M,h,po,no,Ntot,saio,1)... 
    *po(N+1)*difflambda_func(M,saio,-1));

c(2,1)= 0;

c(2,2)= (mu_func(M,h,po,no,Ntot,saio,-1)/hpr(N))... 
    *(lambda_func(M,saio,1)/h(M));

c(2,3)= -(1/(hpr(N)*h(M)))*(mu_func(M,h,po,no,Ntot,saio,-1)... 
    *no(N)*difflambda_func(M,saio,-1)+mu_func(M,h,po,no,Ntot,saio,-1)... 
    *no(N+1)*difflambda_func(M,saio,1));

c(3,1)= 0;

c(3,2)= 0;

c(3,3)= 1/(h(M)*hpr(N));
end
function f = f_func(N,h,hpr,po,no,Ntot,gamma,saio)
%This function calculates a 1x3 matrix based on the grid point
%index, M, passed into the function.
%This matrix, along with the a,b, and c_func makes up the tri-
%diagonal matrix for each device point a when combined can be solved
%for the error using the recursive tri-diagonal solution algorithm.
% N - Device grid point of interest
% h - Grid mesh spacing
% hpr - Auxiliary grid spacing; same as h for uniform device
% po - Hole concentration
% no - Electron concentration
% Ntot - Total concentration
% gamma - Doping concentration
% saio - Device potential

%The main 3x3 matrices (a,b,c,f) are calculated upon the main grid
%points N, but the sub-equations are calculated on the auxiliary
%grid points. This is why M is set equal to N.
M=N;

eo = 8.8548-14;  %Permittivity Constant
er = 11.7;       %Relative Permittivity of Silicon
es = eo * er;    %Permittivity of Silicon
q   = 1.6028-19; %Electron Charge

gamma1 = 1/(h(M-1)*hpr(N));
gamma2 = -(1/hpr(N))*(1/h(M-1) + 1/h(M));
gamma3 = 1/(h(M)*hpr(N));

f(1,1)= -(1/(q*hpr(N)))*(current_func(M,h,po,no,Ntot,saio,1) -
- current_func(M-1,h,po,no,Ntot,saio,1)) -
srh_func(N,po,no,Ntot);

f(2,1)= -(1/(q*hpr(N)))*(current_func(M,h,po,no,Ntot,saio,-1) -
- current_func(M-1,h,po,no,Ntot,saio,-1)) +
srh_func(N,po,no,Ntot);

f(3,1)= (-q/es)*(gamma(N)+po(N)-no(N)) - gamma1*saio(N-1) -
- gamma2*saio(N) - gamma3*saio(N+1);

assignin('base','gamma1',gamma1);
assignin('base','gamma2',gamma2);
assignin('base','gamma3',gamma3);
end
function current_out = current_func(M,h,po,no,Ntot,saio,I)
%This function calculates and returns the hole or electron current
%density based on which index, I is passed into the function.
%The current and hole density equations come from the numerical
%equations established in order to solve the physical differential
%equations. It uses the lambda and mobility functions as well.
% M - Device grid point of interest
% h - Grid mesh spacing
% po - Hole concentration
% no - Electron concentration
% Ntot - Total concentration
% saio - Device potential
% I - Index used to define which current hole or electron,
% to output

q = 1.602E-19; %Electron Charge

%M - Device grid point of interest
%h - Grid mesh spacing
%po - Hole concentration
%no - Electron concentration
%Ntot - Total concentration
%saio - Device potential
%I - Index used to define which current hole or electron,
%to output

%The main 3x3 matrices (a,b,c,f) are calculated upon the main grid
%points N, but the sub-equations are calculated on the auxiliary
%grid points. This is why N is set equal to M.
N=M;

if(I == 1) 
current_out = (q/h(M))*(mu_func(M,h,po,no,Ntot,saio,1)) ... 
*lambda_func(M,saio,1)*po(N)+ 
mu_func(M,h,po,no,Ntot,saio,1) ... 
*lambda_func(M,saio,-1)*po(N+1));
elseif(I == -1) 
current_out = (q/h(M))*(mu_func(M,h,po,no,Ntot,saio,-1)) ... 
*lambda_func(M,saio,-1)*no(N)+ mu_func(M,h,po,no,Ntot,saio,-1) ... 
*lambda_func(M,saio,1)*no(N+1));
end

-----------------------------------------------

function beta_out = beta_func(M, saio)
%This function is only a called function, usually by another
%function, in order to calculate the beta parameter which is found
%in some of the numerical equations, for example, a,b,c,d_func,
%current_func, etc. Physically beta has no meaning and is only a
%variable.
Vt = 0.02586;
theta = 1/Vt;

beta_out = theta*(saio(M)-saio(M+1));

end
function lambda_out = lambda_func(M, saio, I)
% This function is only a called function, usually by another
% function, in order to calculate the lambda parameter which is found
% in some of the numerical equations, for example, a, b, c, d_func,
% current_func, etc. Physically lambda has no meaning and is only a
% variable. For calculation of the lambda parameter the Bernoulli
% function was used. For very small numbers, the physical equation
% breaks down and therefore must be expanded so that larger terms can
% be neglected. This is basically one use of a series expansion.

Vt = 0.02586;
theta = 1/Vt;

temp = beta_func(M, saio);

if (I == 1)
    if (abs(temp) < 1E-5)
        lambda_out = (1/theta)*(1/(1-(1/2)*temp+(1/6)*temp^2));
    else
        lambda_out = (1/theta)*(temp / (1 - exp(-temp)));
    end
elseif (I == -1)
    if (abs(temp) < 1E-5)
        lambda_out = -(1/theta)*(1/(1+(1/2)*temp+(1/6)*temp^2));
    else
        lambda_out = (1/theta)*(temp / (1 - exp(temp)));
    end
end

function difflambda_out = difflambda_func(M, saio, I)
% This function is only a called function, usually by another
% function, in order to calculate the derivative of the lambda
% parameter which is found in some of the numerical equations, for
% example, a, b, c, d_func.

temp = sign(I)*beta_func(M, saio);

if (abs(temp) < 1E-5)
    difflambda_out = 0.5 + sign(I)*(temp/6);
else
    difflambda_out = (1/(1-exp(-temp)))*(1-temp/(exp(temp)-1));
end
function E_out = energy_func(M,h,saio)
% This function calculates the energy field at a given device point
% based on the given grid point, mesh spacing, and device potential.
% The E-field is simply the difference between two grid point
% potentials divided by the mesh spacing.

N=M;

E_out = -(saio(N+1)-saio(N))/h(M);
end

function invBpr_out = inverse_func(Bpr)
% This function is used to calculate the inverse of a 3x3 matrix; the
% built in Matlab function det is used to calculate the determinate
% of the matrix

invBpr_out(1,1) = (Bpr(2,2)*Bpr(3,3) - Bpr(3,2)*Bpr(2,3))/det(Bpr);
invBpr_out(1,2) = (Bpr(3,2)*Bpr(1,3) - Bpr(1,2)*Bpr(3,3))/det(Bpr);
invBpr_out(1,3) = (Bpr(1,2)*Bpr(2,3) - Bpr(1,3)*Bpr(2,2))/det(Bpr);

invBpr_out(2,1) = (Bpr(3,1)*Bpr(2,3) - Bpr(2,1)*Bpr(3,3))/det(Bpr);
invBpr_out(2,2) = (Bpr(1,1)*Bpr(3,3) - Bpr(3,1)*Bpr(1,3))/det(Bpr);
invBpr_out(2,3) = (Bpr(2,1)*Bpr(1,3) - Bpr(1,1)*Bpr(2,3))/det(Bpr);

invBpr_out(3,1) = (Bpr(2,1)*Bpr(3,2) - Bpr(3,1)*Bpr(2,2))/det(Bpr);
invBpr_out(3,2) = (Bpr(3,1)*Bpr(1,2) - Bpr(1,1)*Bpr(3,2))/det(Bpr);
invBpr_out(3,3) = (Bpr(1,1)*Bpr(2,2) - Bpr(2,1)*Bpr(1,2))/det(Bpr);
end
function gen_out = gen_func(N,h,hpr,po,no,Ntot,saio)

M = N;

q = 1.602E-19; %Electron Charge

alpha_p0 = 2.25E7;
E_p0 = 3.26E6;
alpha_n0 = 3.80E6;
E_n0 = 1.75E6;
m=1;

ma = h(M)/(2*hpr(N));
mb = h(M-1)/(2*hpr(N));

E = ma*energy_func(M-1,h,saio) + mb*energy_func(M,h,saio);
alpha_p = alpha_p0*exp(-(E_p0/abs(E))^(m));
alpha_n = alpha_n0*exp(-(E_n0/abs(E))^(m));

J_p = ma*current_func(M-1,h,po,no,Ntot,saio,1) +
    mb*current_func(M,h,po,no,Ntot,saio,1);
J_n = ma*current_func(M-1,h,po,no,Ntot,saio,-1) +
    mb*current_func(M,h,po,no,Ntot,saio,-1);

gen_out = (1/q)*(alpha_p*abs(J_p) + alpha_n*abs(J_n));

end
function gen_out = diffgen_func(N,h,hpr,po,no,Ntot,saio,I)

q = 1.602E-19; %Electron Charge

alpha_p0 = 2.25E7;
E_p0 = 3.26E6;
alpha_n0 = 3.80E6;
E_n0 = 1.75E6;
m=1;

M=N;

ma = h(M)/(2*hpr(N));
mb = h(M-1)/(2*hpr(N));

E = ma*energy_func(M-1,h,saio) + mb*energy_func(M,h,saio);
alpha_p = alpha_p0*exp(-(E_p0/abs(E))^
alpha_n = alpha_n0*exp(-(E_n0/abs(E))^

J_p = ma*current_func(M-1,h,po,no,Ntot,saio,1) +
J_n = ma*current_func(M-1,h,po,no,Ntot,saio,-1) +

G11 = (ma/h(M-1))*alpha_p*(E_p0/E^2)*(abs(J_p)/q)*sign(E);
G12 = (ma*alpha_p)/h(M-1)*mu_func(M-1,h,po,no,Ntot,saio,1)*po(N-
   mu_func(M-1,h,po,no,Ntot,saio,1)*po(N)*difflambda_func(M-
G13 = (ma/h(M-1))*alpha_n*(E_n0/E^2)*(abs(J_n)/q)*sign(E);
G14 = (ma*alpha_n)/h(M-1)*mu_func(M-1,h,po,no,Ntot,saio,-1)*no(N-
   mu_func(M-1,h,po,no,Ntot,saio,-1)*no(N)*difflambda_func(M-

G31 = (mb/h(M))*alpha_p*(E_p0/E^2)*(abs(J_p)/q)*sign(E);
G32 = (mb*alpha_p)/h(M)*mu_func(M,h,po,no,Ntot,saio,1)*po(N)*difflambda
   mu_func(M,h,po,no,Ntot,saio,1)*po(N+1)*difflambda_func(M,saio,2)*si
G33 = (mb/h(M))*alpha_n*(E_n0/E^2)*(abs(J_n)/q)*sign(E);
G34 = (mb*alpha_n)/h(M)*mu_func(M,h,po,no,Ntot,saio,1)*no(N)*difflambda

109
\[ rnu\_func(M,h,po,no,Ntot,saio,l)\times no(N+l)\times difflambda\_func(M,saio,1) \times \text{sign}(J_n); \]

\[ G21 = - G11 - G31; \]
\[ G22 = \alpha_p \times ((mb/h(M)) \times (rnu\_func(M,h,po,no,Ntot,saio,l) \times po(N) \times difflambda\_func(M,saio,1) \times no(N+l) \times difflambda\_func(M,saio,2)) - (ma/h(M-1)) \times (rnu\_func(M-1,h,po,no,Ntot,saio,l) \times po(N-1) \times difflambda\_func(M-1,saio,1) \times no(N-1) \times difflambda\_func(M-1,saio,2)) \times \text{sign}(J_p); \]

\[ G23 = - G13 - G33; \]
\[ G24 = \alpha_n \times ((mb/h(M)) \times (rnu\_func(M,h,po,no,Ntot,saio,l) \times no(N) \times difflambda\_func(M,saio,3) \times no(N+l) \times difflambda\_func(M,saio,4)) - (ma/h(M-1)) \times (rnu\_func(M-1,h,po,no,Ntot,saio,-1) \times no(N-1) \times difflambda\_func(M-1,saio,3) \times no(N-1) \times difflambda\_func(M-1,saio,4)) \times \text{sign}(J_n); \]

assignin ('base', 'rna', rna);
assignin ('base', 'mb', mb);
assignin ('base', 'E', E);
assignin ('base', 'alpha_p', alpha_p);
assignin ('base', 'alpha_n', alpha_n);
assignin ('base', 'J_p', J_p);
assignin ('base', 'J_n', J_n);
assignin ('base', 'G11', G11);
assignin ('base', 'G12', G12);
assignin ('base', 'G13', G13);
assignin ('base', 'G14', G14);
assignin ('base', 'G21', G21);
assignin ('base', 'G22', G22);
assignin ('base', 'G23', G23);
assignin ('base', 'G24', G24);
assignin ('base', 'G31', G31);
assignin ('base', 'G32', G32);
assignin ('base', 'G33', G33);
assignin ('base', 'G34', G34);

if (I == 1)
    gen_out = (ma/h(M-1)) \times \text{mu\_func}(M-1,h,po,no,Ntot,saio,l) \times \text{lamda\_func}(M-1,saio,1) \times \alpha_p \times \text{sign}(J_p);
elseif (I == 2)
    gen_out = ((ma/h(M-1)) \times \text{mu\_func}(M-1,h,po,no,Ntot,saio,l) \times \text{lamda\_func}(M-1,saio,-1) \times \alpha_p \times \text{sign}(J_p)
elseif (I == 3)
    gen_out = (mb/h(M)) * mu_func(M,h,po,no,Ntot,saio,l) * lambda_func(M,saio,l) * alpha_p * sign(J_p);
elseif (I == 4)
    gen_out = (ma/h(M-1)) * mu_func(M-1,h,po,no,Ntot,saio,-1) * lambda_func(M-1,saio,-1) * alpha_n * sign(J_n);
elseif (I == 5)
    gen_out = ((ma/h(M-1)) * mu_func(M-1,h,po,no,Ntot,saio,-1) * lambda_func(M-1,saio,1) ...
            + (mb/h(M)) * mu_func(M,h,po,no,Ntot,saio,-1) * lambda_func(M,saio,-1)) * alpha_n * sign(J_n);
elseif (I == 6)
    gen_out = (mb/h(M)) * mu_func(M,h,po,no,Ntot,saio,-1) * lambda_func(M,saio,1) * alpha_n * sign(J_n);
elseif (I == 7)
    gen_out = G11 + G12 + G13 + G14;
elseif (I == 8)
    gen_out = G21 + G22 + G23 + G24;
elseif (I == 9)
    gen_out = G31 + G32 + G33 + G34;
end
function srh_out = srh_func(N,po,no,Ntot)
%This function calculates the Shockley Read-Hall Recombination term
%at a given device grid point. Only the current carrier
%concentrations are required to calculate this parameter.
%The SRH term is used in order to calculate the current in a forward
%biased device. Once a solution is found, the SRH terms can be re-
%calculated across the device and a simple numerical integration
%method can be used to calculate the current. Employing these terms
%in the overall tri-diagonal matrix gives a more accurate solution.

ni = 1.45E10; %Intrinsic Carrier Concentration

tau_p0 = 0.2E-7;
tau_n0 = 0.2E-7;
Nsrh_p = 5E15;
Nsrh_n = 5E15;

tau_p = tau_p0/(1+(Ntot(N)/Nsrh_p));
tau_n = tau_n0/(1+(Ntot(N)/Nsrh_n));

srh_out = (po(N)*no(N) - ni^2)/(tau_n*(po(N)+ni) +
tau_p*(no(N)+ni));

end
function diffsrh_out = diffsrh_func(N,po,no,Ntot,I)
%This function calculates the derivative of the SRH terms, as when %the numerical equations are expanded, it requires derivative of the %SRH terms as well. If I = 1, then the partial derivative of the %Recombination term is calculated with respect to the hole %concentration and if I = -1, then it is calculated with respect to %the electron concentration.

ni = 1.45E10; %Intrinsic Carrier Concentration

tau_p0 = 0.2E-7; %Hole lifetime constant
tau_n0 = 0.2E-7; %Electron lifetime constant
Nsrh_p = 5E15;  %Hole SRH Carrier concentration constant
Nsrh_n = 5E15;  %Electron SRH Carrier concentration constant

tau_p = tau_p0/(1+(Ntot(N)/Nsrh_p));  %Calculated hole lifetime
tau_n = tau_n0/(1+(Ntot(N)/Nsrh_n));  %Calculated electron lifetime

if (I == 1)
    diffsrh_out = (no(N) - tau_n*srh_func(N,po,no,Ntot))/
                   (tau_n*(po(N)+ni) + tau_p*(no(N)+ni));
elseif(I == -1)
    diffsrh_out = (po(N) - tau_p*srh_func(N,po,no,Ntot))/
                   (tau_n*(po(N)+ni) + tau_p*(no(N)+ni));
end
REFERENCES


