Numerical Methods for Monte Carlo Device Simulation

by

Jennifer Anne Lloyd

B.S., Massachusetts Institute of Technology (1989)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1992

Signature of Author _____
Department of Electrical Engineering and Computer Science
May 12, 1992

Certified by _____
Jacob White
Thesis Supervisor

Accepted by _____
Cambell L. Searle
Chairman, Departmental Committee on Graduate Students

Numerical Methods for Monte Carlo Device Simulation

by

Jennifer Anne Lloyd

Submitted to the Department of Electrical Engineering and Computer Science
on May 13, 1992, in partial fulfiilment of the
requirements for the degree of
Master of Science

Abstract

This thesis investigates several new numerical techniques for Monte Carlo device simulation. A method is presented in which the time integration of the semi-classical motion equations is computed with an implicit multistep method. Additionally, an electric field calculation is proposed in which the particle-particle forces and the particle-doping forces are computed separately. Simulations were used to verify the improvement that both of these methods make over standard explicit mesh-based methods. Results from the simulations are presented and evaluated.

Thesis Supervisor: Jacob White

Associate Professor of Electrical Engineering and Computer Science

Acknowledgments

Funding for this research was provided by the National Science Foundation contract MIP-8858764 A02 and the Defense Advanced Research Projects Agency contract N00014-91-J-1698. Many thanks to these organizations for making this thesis possible.

I wish to thank my thesis advisor, Professor Jacob White, for originally suggesting my thesis topic. The direction and advice he provided were invaluable. Ken Szajda, Ignacio McQuirk, and Rajni Aggarwal were of great assistance, providing important comments and advice on the research as it progressed. Thanks also go to Einar Rønquist, Joel Phillips, and Keith Nabors for all of their technical assistance, without which this thesis would not have been possible.

Above all, special thanks to my family, who have been so supportive of my academic pursuits over the years.

Contents

List of Figures

1

Introduction

Semiconductor device simulation entails numerically solving the partial differential equations associated with carrier transport in a device. Since the mathematical analysis of devices is quite complex, device simulation has always been an invaluable and important tool for semiconductor circuit and device developers. The classical semiconductor equations are coupled, and must be solved numerically because closed form solutions can be derived only in a limited number of relatively simple cases. Semiconductor device simulators are therefore widely applied as tools for both prediction and verification; device simulation can be used for new technology development, process design, extraction of parameters for lumped models, and simulation of critical circuits.

The Monte Carlo method is widely applied to the simulation of charge transport in semiconductor devices, which is accurately modeled by the Boltzmann Transport Equation (BTE) coupled with Poisson's equation. The Monte Carlo method yields a numerical solution to the BTE in the form of a distribution of the possible states of the system of particles. There are two major advantages to this approach: First, it provides realistic treatment of small-geometry effects when compared with standard simulations which use the Drift-Diffusion model or other simplifications of the BTE [1, 2, 3, 4, 5, 6]. Additionally, the Monte Carlo method allows all of the physical phenomena present in a semiconductor to be represented in terms of the band structure and scattering mechanisms for that semiconductor. Such a representation makes these simulations manageable from a programming point of view, since the scattering mechanisms can easily be added or deleted from the simulation model. There is one major drawback, however; although the Monte Carlo method provides these advantages over standard semiconductor simulations, it is computationally expensive. Considering methods for improving the numerical

efficiency of Monte Carlo methods is therefore of practical importance.

Ensemble Monte Carlo simulation involves following a large number of particles in time and space through a semiconductor and accumulating statistics on the overall behavior of the ensemble. As particles move, the electric field due to the mobile charges changes. The field must be recomputed frequently from Poisson's equation, which relates the charge distribution to the electric field. Both the time and space discretization of this self-consistent electric field calculation are key numerical issues.

This thesis investigates the implicit discretization of the semiclassical equations of motion. This discretization should stabilize the simulation such that larger timesteps than those used in explicit simulators can be used, and the number of costly electric field calculations can be reduced. Additionally, improving the accuracy of the electric field calculations using direct particle-particle methods is investigated.

The thesis begins by describing the details of the Monte Carlo simulation algorithm and the physical models that are used to represent a semiconductor device. A brief review of the literature describing Monte Carlo applications is presented, followed by a review of the details of standard electric field calculations in Monte Carlo simulation.

The third chapter discusses the difficulties with the standard numerical methods for Monte Carlo simulation that are described in the Background chapter. The problem with grid based approaches to electric field calculation is presented, and improvements are suggested. Additionally, the difficulty with using explicit integration techniques is discussed and a possibly improved implicit integration scheme is proposed.

The fourth chapter describes the simulation system in which all of the experiments for this research were performed. The dimensions and parameters of the physical model are described. A description of the specific numerical methods used in the experiments is given, as well as the details of the standard and proposed simulation algorithms. Additionally, a description of the computation of average quantities in the simulation system is described.

The fifth chapter presents the results from a series of experiments that were used to make a comparison between an implicit and explicit discretizations of the semiclassical equations. The results of these simulations, which demonstrated that the implicit methods yield more stable behavior than explicit methods, are evaluated and compared with predicted results.

The final chapter summarizes the work in this thesis, focusing on the applicability of these new numerical techniques to Monte Carlo simulations. The results obtained from the simulation

experiments are evaluated in this context. Further extensions of the simulation enhancements and other suggestions for future research are also outlined.

2

Background

2.1 Introduction

The Monte Carlo method can be used to simulate many important devices; it can give insight into physical effects that are not easily modeled in standard simulators. Two important features of Monte Carlo simulators are both the physical models employed and the method of electric field computation used. This chapter discusses both of these issues. First, the basic algorithm and physical models used in Monte Carlo simulators are detailed, then standard electric field computations are described.

2.2 Overview of the Monte Carlo Method

Monte Carlo simulation of charge transport in semiconductor devices involves tracking the behavior of one or several particles in both space and time. These "labelled" particles, which often represent a larger number of physical particles, are subjected to forces due to both the electric field as well as a set of scattering mechanisms. Throughout the simulation, the particle history is recorded so that statistics can be accumulated on overall particle behavior [7].

The Monte Carlo algorithm can be implemented as either a single particle or an ensemble particle simulation. Single particle algorithms are used to simulate steady state behavior; statistics are accumulated by averaging the particle's behavior over the entire simulation time. Algorithms in which an ensemble of particles are simulated are used to study steady state *or* transient behavior, since statistics can be formed at one point in time if enough particles are used. Typically 10,000 or more particles are needed to collect a reasonably accurate set of

statistics [7].

In addition, Monte Carlo simulation has been used to extract bulk parameters such as mobility and momentum relaxation time, which can be used in drift-diffusion [8, 9, 10] or hydrodynamic simulators. It has also been used to study impact ionization [6], high-energy transport [11, 1, 12] and gate and substrate current in MOSFET's.

2.3 Details of the Monte Carlo Method

The following section describes the physical models used in Monte Carlo simulations. These include both the semiclassical model for the movement of a Bloch state and the models for the random scattering processes in silicon that describe transitions between Bloch states. In addition, the actual algorithm for computing device behavior is outlined. With such a broad range of application, optimization of Monte Carlo simulators and simulation algorithms take on a special importance.

2.3.1 Semiclassical Particle Motion

In general an electron can be described by a wave function $\Psi(\vec{r})$. Under time-independent assumptions, the wave function will satisfy the following eigenvalue differential equation (Schrödinger's time-independent equation):

$$\left[-\frac{\hbar^2}{2m}\nabla^2 + V(\vec{r})\right]\Psi(\vec{r}) = \mathcal{E}\Psi(\vec{r}) \tag{2.1}$$

where $\hbar$ is Planck's constant, m is the particle mass, and $\mathcal{E}$ is the particle's eigenenergy. The prefactor of $\Psi(\vec{r})$ is called the Hamiltonian operator, and is a sum of both kinetic and potential energy operators; the potential energy operator is $V(\vec{r})$ and the kinetic energy operator is $-\frac{\hbar^2}{2m}\nabla^2$. (This is derived from substituting the momentum operator, $\hat{p} = \frac{\hbar}{\mathrm{i}}\nabla$ into the kinetic energy operator, $\frac{\hat{p}^2}{2m}$.) The eigenfunction $\Psi(\vec{r})$ can be interpreted such that $|\Psi(\vec{r})|^2$ is the probability of finding the particle in a differential volume element at $\vec{r}$. For a free particle where $V(\vec{r}) = 0$, the wave function is $\Psi(\vec{r}) = e^{i\vec{k}\cdot\vec{r}}$ and the associated eigenenergy is $\mathcal{E} = \frac{\hbar^2\vec{k}^2}{2m}$, where $\vec{k}$ is the particle's wavevector [13].

An electron in a crystal is subjected to a periodic potential arising from the atomic structure of the crystal itself. Under the assumptions of the effective mass theorem, the electron will obey

the isotropic effective mass Schrödinger equation (time-independent):

$$\left[-\frac{\hbar^2}{2m^*}\nabla^2 + V(\vec{r})\right]\Psi(\vec{r}) = \mathcal{E}\Psi(\vec{r}) \tag{2.2}$$

where $V(\vec{r})$ is the crystal potential and m^* is the particle's effective mass [14]. The periodicity of the lattice potential imposes a periodicity on the electron wavefunctions. Under such conditions, the electron is called a Bloch electron, and the associated Bloch wavefunctions are described by:

$$\Psi_{n\vec{k}}(\vec{r}) = e^{i\vec{k}\cdot\vec{r}}u_{n\vec{k}}(\vec{r}) \tag{2.3}$$

where $u_{n\vec{k}}(\vec{r})$ is a function with the same periodicity as the underlying Bravais lattice. The subscripts n and $\vec{k}$ refer to the band index and the wavevector, respectively, of the Bloch electron. The wavevector $\vec{k}$ is confined to the first Brouillin zone.

The properties of the Bloch electron that are pertinent to the semiclassical approximation are as follows. First, it can be readily shown that the Bloch wavefunctions are periodic in the reciprocal lattice vector of the Bravais lattice. Second, Bloch electrons can be described by an average momentum, or crystal momentum, $\hbar\vec{k}$. It is important to note that for Bloch electrons, the wavefunction $\Psi_{nk}(\vec{r})$ is not a momentum eigenstate; the actual momentum of the Bloch electron is found by applying the momentum operator, $\hat{p} = \frac{\hbar}{i}\nabla$, to the Bloch eigenstates of Equation 2.3:

$$\frac{\hbar}{i}\nabla\Psi_{n\vec{k}} = \hbar\vec{k}\Psi_{n\vec{k}} + e^{i\vec{k}\cdot\vec{r}}\frac{\hbar}{i}\nabla u_{n\vec{k}}(\vec{r}) \tag{2.4}$$

Finally, the possible electronic levels of the Bloch electrons can be represented by the band structure, $\mathcal{E}_n(\vec{k})$. For a given band index n, $\mathcal{E}_n(\vec{k})$ relates the energy of a particle to its wavevector.

In classical mechanics, the position and momentum of a particle can be described by Hamilton's conjugate equations:

$$\frac{\partial\vec{r}}{\partial t} = \frac{\partial H}{\partial\vec{p}} \tag{2.5}$$

$$\frac{\partial\vec{p}}{\partial t} = -\frac{\partial H}{\partial\vec{r}} \tag{2.6}$$

where $\vec{r}$ is the particle position and $\vec{p}$ is the particle momentum, and H is the Hamiltonian of the system. Applying Hamilton's equations to a quantum mechanical system of Bloch waves gives the semi-classical equations of motion. The semiclassical approximation to particle motion in a crystal in the absence of collisions assumes that a particle in a periodic potential field can be described simultaneously by a position vector $\vec{r}$, a wavevector $\vec{k}$ and a band index, n, which is

a constant of the particle motion. The time evolution of both the position and wavevectors is given by:

$$\frac{d\vec{r}}{dt} = \vec{v}_n(\vec{k}) = \frac{1}{\hbar}\nabla_{\vec{k}}\mathcal{E}_n(\vec{k}) \tag{2.7}$$

$$\hbar\frac{d\vec{k}}{dt} = -q\left(\vec{E} + \frac{1}{c}\vec{v}\times\vec{B}\right) \tag{2.8}$$

where q is the magnitude of the electronic charge, c is the velocity of light in the material, and $\vec{E}$ is the electric field on the particle. Equation 2.8 reduces to:

$$\hbar\frac{d\vec{k}}{dt} = -q\vec{E} \tag{2.9}$$

in the absence of a magnetic field.

2.3.2 Scattering Processes in Semiconductors

The semiclassical equations of motion yield a description of particle motion in the absence of collisions. In a semiconductor, however, particles are subjected to both elastic and inelastic collisions due to many processes, including optical phonons, acoustic phonons, and ionized impurities. These collisions will perturb the motion of the carriers. The collision processes can be modeled such that they are described by both a scattering rate (which is a function of carrier energy, position, etc.) and a rule for computing the new Bloch state with a new wavevector $\vec{k}'$ and a new energy $\mathcal{E}_{n'}(k')$ from an old Bloch state with a wavevector $\vec{k}$ and energy $\mathcal{E}_n(k)$. In a simulation environment, these transitions between Bloch states are assumed to happen instantaneously.

The scattering rates are formulated by considering weak perturbations to the periodic potential of the lattice, $V(\vec{r},t)$, that are a function of both time and space. From time-dependent perturbation theory, the transition rate $S(\vec{k},\vec{k}')$ is derived for electrons from state $|n,\vec{k}>$ to state $|n',\vec{k}'>$. The probability of finding an electron in state $|n',\vec{k}'>$ after time t is written as:

$$|a_{n'\vec{k}'}(t)|^2 = \left(\frac{|<n,\vec{k}|H'|n',\vec{k}'>|^2}{\hbar^2}\right)\frac{4sin^2 w_{n'n}\frac{t}{2}}{w^2_{n'n}} \tag{2.10}$$

where:

$$<n,\vec{k}|H'|n',\vec{k}'> = \int u^*_{n\vec{k}}(\vec{r})H'(\vec{r},t)u_{n'\vec{k}'}(\vec{r})d\vec{r} \tag{2.11}$$

and $w_{n'n}$ is the Bohr frequency between states $|n,\vec{k}>$ and $|n',\vec{k}'>$. $H'(\vec{r},t)$ is the perturbation Hamiltonian, and $<n,\vec{k}|H'|n',\vec{k}'>$ is the time dependent matrix element. In order to get a rate of transition probability between two states $|n,\vec{k}>$ and $|n',\vec{k}'>$, the probability $|a_{n'\vec{k}'}(t)|^2$ must

be integrated over the available energy states $\delta\left[\mathcal{E}_{n'}(k') - \mathcal{E}_n(k)\right]$. Performing the integration yields:

$$S(k,k') \cong \frac{2\pi}{\hbar}| < n,\vec{k}|H'|n',\vec{k}' > |^2 \delta\left[\mathcal{E}_{n'}(k) - \mathcal{E}_n(k)\right] \tag{2.12}$$

which is more familiarly known as Fermi's Golden Rule [15]. Given a model for the perturbation, $H'(\vec{r},t)$, matrix elements can be calculated and substituted into Equation 2.12 in order to give a scattering rate as a function of energy and other parameters (such as doping and local electron concentration).

Given a set of scattering mechanisms, scattering rate selection is a random process. For single particle algorithms, scattering rates are typically selected using the "self-scattering" method [16]. For ensemble algorithms where this is not practical, a scattering mechanism with scattering time τ is chosen if a generated random number r is less than $\frac{h}{\tau}$, where h is the particle's free-flight time. This is based on a differential form of the scattering rate, where the probability of scattering in an incremental amount of time h can be approximated by:

$$P_{sc}(t+h) \approx \frac{h}{\tau}. \tag{2.13}$$

It has been shown that this method of selection is only valid for timesteps approximately one-tenth of τ. Final $\vec{k}$ states after scattering are chosen by computing the new energy of the particle, and then finding the new wavevector from the rules associated with the scattering mechanism.

2.3.3 Band Structure Modeling

The band structure of the semiconductor to be simulated is crucial since the semiclassical approximation assumes that it is known. The focus of this research is silicon-based devices. Realistic band structures derived using the emperical pseudopotential method have been incorporated into Monte Carlo simulations, and need to be used in order to correctly simulate devices[16]. However, using the emperical band structures is costly in terms of storage and interpolations.

For this research, an analytical description of the band structure is used instead. Assuming spherical, parabolic bands, the energy of an electron around the conduction band minima is:

$$\mathcal{E}_c(\vec{k}) = \frac{\hbar^2\vec{k}^2}{2m^*} \tag{2.14}$$

where m^* is a scalar effective mass. This band structure can be corrected for non-parabolicity, such that:

$$\mathcal{E}_c(\vec{k})(1+\alpha\mathcal{E}) = \frac{\hbar^2\vec{k}^2}{2m^*} \tag{2.15}$$

where α is a non-parabolicity factor. From this, the velocity is easily derived:

$$\vec{v}_c(\vec{k}) = \frac{\hbar\vec{k}}{m^*(1+2\alpha\mathcal{E})} \tag{2.16}$$

The previous two equations account analytically for deviations from the spherical bands far from the minima of the conduction band.

2.3.4 The Monte Carlo Algorithm

In order to simulate particle motion, the semiclassical motion Equations 2.7 and 2.8 are typically discretized so that

$$\vec{x}(t+h) - \vec{x}(t) = h \cdot \vec{v}(t+h) \tag{2.17}$$

$$\hbar(\vec{k}(t+h) - \vec{k}(t)) = -h \cdot q \cdot \vec{E}(t) \tag{2.18}$$

where h is the free-flight or drift time, q is the magnitude of the electronic charge, and $\vec{v}$ is the particle velocity. Note that $\vec{v}(t+h)$ is computed from the band structure dispersion relationship given in Equation 2.7. This type of discretization is explicit, since the only information used to calculate the new wavevector and position is from previous timepoints.

Given the above discretization of the equation of motion, one time-loop of an ensemble Monte Carlo simulation would proceed as follows: first, move all of the particles in momentum space for some flight time h and update their wave-vectors according to the discretized motion Equation 2.18. Compute the new velocities and energies of the particles according to the semiconductor band structure. Move the particles in space, and update their positions with the new computed velocities according the the second discretized motion Equation 2.17. At the end of the free flight, determine which particles will scatter and compute their new wavevectors depending on which randomly determined scattering process will occur. The scattering mechanisms are modeled in terms of carrier energy and momentum so that a new state of the carrier can be determined. At the end of the time-loop, the electric field must be recomputed, since the charge density inside the device has now changed. This time loop repeats until the simulation ending time is reached. During each time loop, information on the carrier position, velocity, energy, etc. is recorded. The data collected over the flight of the particle is then averaged to form a set of statistics [7].

In order to properly simulate a device, boundary conditions for the particles must be incorporated into the simulation. Typically, in the vicinity of ohmic contacts particles are either injected or removed from the simulation in such a way that charge neutrality is maintained in some region around the contact. Additionally, reflecting boundary conditions are used at non-contacted boundaries. Boundary conditions for the potential and the gradient of the potential are included in the electric field computation, which is described in the next section.

2.4 Electric Field Computations in Monte Carlo Simulation

In general, electrostatic potential in a device satisfies Poisson's equation, namely:

$$\nabla^2\psi = -\frac{\rho}{\epsilon} = -q\frac{(N_d^+ - N_a^- + p - n)}{\epsilon} \tag{2.19}$$

where ψ is the scalar potential, ρ is the net charge density, $N_d^+ - N_a^- = N$ is the net ionized doping concentration, n and p are the electron and hole concentrations, respectively, and ϵ is the dielectric constant for the semiconductor. In an ensemble Monte Carlo simulation, the holes are usually ignored, so that Equation 2.19 reduces to:

$$\nabla^2\psi = -\frac{\rho}{\epsilon} = -q\frac{(N - n)}{\epsilon} \tag{2.20}$$

Assuming that N and n can be assigned to mesh points i, Poisson's equation can be discretized and solved on a finite-difference mesh. The second derivative of any quantity defined on the mesh can be approximated in terms of mesh quantities in several ways. A centered finite difference approximation to the second derivative of ψ with respect to x at node i,j is:

$$\frac{\partial^2\psi}{\partial x^2} = \frac{\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j}}{(\Delta x)^2} \tag{2.21}$$

where Δx is the mesh spacing in the x direction. A uniform mesh is assumed. In two- and higher dimensions, the Laplacian operator is the sum of the second partial derivatives in all dimensions, so that Poisson's equation in two dimensions, for example, would become:

$$\nabla^2\psi = \frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} = \frac{\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j}}{(\Delta x)^2} + \frac{\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}}{(\Delta y)^2} = -q\frac{(N_i - n_i)}{\epsilon} \tag{2.22}$$

where N_i and n_i represent the doping concentration and the charge density that have been projected onto node i.

Since Equation 2.22 can be written for all nodes (i,j), a system of algebraic equations results. For simulation of devices, two boundary conditions on the potential exist. First, at ideal

ohmic contact nodes, the potential is fixed (Dirchlet boundary conditions). Second, at other device edges, the electric field must vanish, i.e. the gradient of ψ is zero (Neumann boundary conditions). When these boundary conditions are included, the linear system of equations can be solved using several methods, such as sparse Gaussian elimination or conjugate gradient iteration. For this research, an ICCG (incomplete Cholesky conjugate gradient) iteration was chosen for the three-dimensional mesh-based simulations in order to avoid allocating memory for large matrices.

The mesh potentials found by solving Equation 2.22 at all the nodes are interpolated into electric field forces. By definition, the electric field and scalar potential are related by:

$$\vec{E} = -\nabla\psi. \tag{2.23}$$

The electric field can be translated into a force on a particle by multiplying by the particle's charge. (For example, the force on an electron is $-q\vec{E}$.)

In Equation 2.22, n_i is formed by spatially mapping the mobile charged particles onto the mesh using a nearest-grid-point (NGP) or cloud-in-a-cell (CIC) approach [17]. Both of these methods are shown pictorially in Figure 2-1. The NGP method entails simply associating each particle with its closest mesh point to form the charge density. The electric field is computed on the grid as described by Equation 2.25, and the electric field acting on each particle is the electric field of the particle's nearest grid point. As an example of NGP weighting, assume a uniform mesh with spacing $\Delta x, \Delta y$. Let a particle of strength q be located at a point (x, y), where x is between grid points i and $i+1$ and y is between grid points j and $j+1$. If (x, y) is closest to node i, j, (i.e. $x - x_j < \frac{\Delta x}{2}$ and $y - y_j < \frac{\Delta y}{2}$), then the charge density due to this particle would be:

$$\rho_m = -\frac{q}{\epsilon}\delta_{i,j} \tag{2.24}$$

where ρ_m denotes the charge density projected on the mesh, and $\delta_{i,j}$ is the Kronecker delta function, defined to have a value of 1 at node i, j and zero elsewhere. From this charge density, the simulator can calculate the potentials at all the nodes and the electric field $\vec{E}_{i,j}$ can be computed using a centered difference approximation. The electric field on a two-dimensional mesh at node i, j is:

$$\vec{E} = (E_x, E_y) = -\left(\frac{\psi_{i+1,j} - \psi_{i-1,j}}{2\Delta x}, \frac{\psi_{i,j+1} - \psi_{i,j-1}}{2\Delta y}\right) \tag{2.25}$$

again assuming a uniform mesh. One obvious difficulty of this method is that the charge density

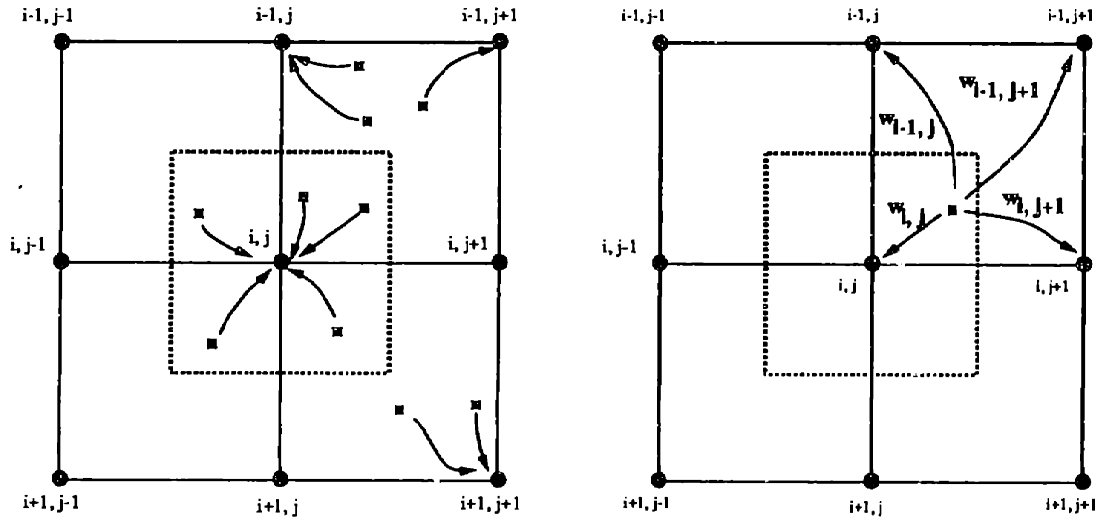

FIGURE 2-1: NGP and CIC mesh assignment schemes.

ρ_m is not smooth. In addition, the electric field is piecewise constant, meaning that a particle crossing a grid line will encounter a discontinuity in electric field.

The CIC method improves upon the accuracy of NGP by spatially weighting the charges onto their four surrounding mesh points to form the charge density. Electric fields are spatially weighted back onto the particles in exactly the inverse way. In order to reduce errors for CIC, the following weights are assigned for each of the four nodes surrounding a particle in a two-dimensional mesh:

$$\begin{aligned} W_{i,j} &= \left(1 - \frac{x - x_j}{\Delta x}\right)\left(1 - \frac{y - y_j}{\Delta y}\right) \\ W_{i+1,j} &= \left(\frac{x - x_j}{\Delta x}\right)\left(1 - \frac{y - y_j}{\Delta y}\right) \\ W_{i,j+1} &= \left(1 - \frac{x - x_j}{\Delta x}\right)\left(\frac{y - y_j}{\Delta y}\right) \\ W_{i+1,j+1} &= \left(\frac{x - x_j}{\Delta x}\right)\left(\frac{y - y_j}{\Delta y}\right) \end{aligned} \tag{2.26}$$

Clearly the sum of these weights must be unity to maintain overall charge conservation. Thus to find the charge density on the mesh, ρ_m, due to the particle at (x, y), assign the charge of the particle times the weight of the node to each of the four surrounding nodes. Assuming that each of the surrounding nodes can be labeled p where $p \in \{(i,j),(i,j+1),(i+1,j),(i+1,j+1)\}$, then:

$$\rho_m = -\frac{q}{\epsilon}\sum_p W_p \delta_p. \tag{2.27}$$

After the potential has been calculated, the electric field can be interpolated at the actual particle locations. With $\vec{E}$ computed at the nodes from Equation 2.25, these electric fields are interpolated from the mesh to these locations, so that at the point (x, y):

$$\vec{E}(x,y) = \sum_p W_p \vec{E}_p. \tag{2.28}$$

Specifically, the electric field is linearly interpolated, so that for a two-dimensional calculation, the contribution to a particle's electric field from one of the four surrounding nodes $p = (i, j)$ is computed from the node potentials as:

$$\begin{aligned} E_x &= \frac{\psi_{i+1,j} - \psi_{i,j}}{\Delta x} + \left(\frac{y - y_j}{\Delta y}\right)\left[\frac{(\psi_{i+1,j+1} - \psi_{i,j+1})}{\Delta x} - \frac{(\psi_{i+1,j} - \psi_{i,j})}{\Delta x}\right] \\ E_y &= \frac{\psi_{i,j+1} - \psi_{i,j}}{\Delta y} + \left(\frac{x - x_j}{\Delta x}\right)\left[\frac{(\psi_{i+1,j+1} - \psi_{i+1,j})}{\Delta y} - \frac{(\psi_{i,j+1} - \psi_{i,j})}{\Delta y}\right] \end{aligned} \tag{2.29}$$

Although more computationally expensive, the CIC method produces a much smoother representation of the charge density than the NGP methods. In addition, when particles move between cells, they will not experience discontinuities in force, as the electric field is now piecewise linear between cells (as far as the particles are concerned). These schemes are easily extended to three dimensions by including the weights for four nodes from the third dimension.

Of course, there are some difficulties encountered in these mesh based methods. Both the finite difference approximations to the Laplacian operator and the electric field introduce errors. In addition, the charge assignment to the mesh can also introduce errors in the forces computed on the particles. These limitations will be discussed further in the next chapter.

3

Analysis of Monte Carlo Computations

3.1 Introduction

The Monte Carlo method provides many advantages over standard methods for device simulation because detailed physics can be incorporated into the simulator. However, the method is computationally expensive, due in part to the electric field calculations that must be performed for ensembles of particles. This section describes the motivation for the proposed improvements to the standard numerical techniques utilized in Monte Carlo device simulation.

3.2 Integration of Semi-Classical Equations

A numerical improvement to Monte Carlo simulation involves integrating the semiclassical equations of motion 2.7 and 2.8 implicitly instead of explicitly. The idea of using implicit methods to integrate the particle equations of motion has been used successfully in plasma simulations [18, 19, 20, 21], where the physical model for particle motion is a simplification of that used for device modeling. In plasma simulations, it is desirable to take timesteps that are large compared with the plasma frequency. Since implicit methods have a looser restriction on the timestep than explicit methods, an implicit method can be used to stabilize such simulations. The following sections will examine the stability of explicit versus implicit time discretization schemes, and the applicability of the results to Monte Carlo simulation.

3.2.1 Stability and Accuracy of Linear Multistep Methods

Linear multistep methods can be used to numerically solve initial value problems of the form $y_t = f(y(t), t)$, such as the equations of motion used in Monte Carlo simulation. The 1-step methods of concern are forward Euler, backward Euler and the trapezoidal rule, which are given in the following equations:

$$y_{n+1} = y_n + hf_n \tag{3.1}$$

$$y_{n+1} = y_n + hf_{n+1} \tag{3.2}$$

$$y_{n+1} = y_n + \frac{h}{2}(f_n + f_{n+1}) \tag{3.3}$$

where $f_n = f(y_n, t_n)$. The numerical solution to the initial value problem is found at a set of discrete timepoints t_n. For the above fixed timestep methods, the timepoints are defined to be $t_n = hn$, where h is the timestep, and n is any nonnegative integer. The numerical solution is described by the sequence of values $y_0 \ldots y_n$ at the points t_n. An explicit multistep method is defined to be one for which y_n is computed only from information at previous timepoints. An implicit multistep method, on the other hand, also uses information from future timepoints. Clearly the forward Euler method is an explicit scheme, and both backward Euler and the trapezoidal rule are implicit schemes. In general, implicit methods require solving a nonlinear equation at each timepoint.

A multistep method is stable if, when applied to a stable problem, the computed values of $y(t)$ remain bounded as $t \to \infty$. For any scalar problem of the form $y_t = f(y(t), t) = \lambda y + f(t)$, Gear defines stability as follows: "A multistep method is absolutely stable for those values of $h\lambda$ where the roots of the characteristic polynomial are ≤ 1 in absolute value"[22]. In other words, for any value of $h\lambda$ producing a root of the characteristic polynomial which is > 1, the component of the numerical solution associated with that root grows with time and the solution is unbounded.

The regions of stability for the three integration methods of interest, Equations 3.1, 3.2 and 3.3, are easily derived. The regions of $h\lambda$ corresponding to absolute stability can be plotted in the complex plane. For forward Euler, the absolute stability region is a unit circle centered at $-\frac{1}{2}$. For backward Euler, the absolute stability region is all points in the complex plane excluding the unit circle centered at $\frac{1}{2}$, and for the trapezoidal rule, the absolute stability region is the entire left half plane. Both the backward Euler and the trapezoidal methods are called A-stable because the regions of absolute stability include the left-half plane.

In conjunction with stability, multistep methods can also be characterized by their accuracy. This is often expressed in terms of a local truncation error (LTE), which is defined to be the multistep formula of interest evaluated with the exact solution to $y(t) = f(y(t), t)$. A Taylor series expansion of $y(t+h)$ and $f(y(t+h)) = y'(t+h)$ can be used to derive the following results:

$$\begin{aligned} LTE_{fe} &= \frac{h^2}{2}y''(t) + \frac{h^3}{6}y'''(t) + \dots \\ LTE_{be} &= -\frac{h^2}{2}y''(t) - \frac{h^3}{3}y'''(t) + \dots \\ LTE_{trap} &= -\frac{h^3}{12}y'''(t) - \frac{h^4}{24}y''''(t) + \dots \end{aligned} \tag{3.4}$$

Thus both forward Euler and backward Euler are first-order methods with error $O(h^2)$ and the trapezoidal rule is a second-order method with error $O(h^3)$.

Although the trapezoidal rule and the backward Euler scheme both have the same $O(h^3)$ accuracy, the backward Euler scheme presents difficulties in its application to Monte Carlo simulation due to its energy loss properties.

3.2.2 Implicit Integration of Monte Carlo Equations

Section 2.3.4 described a Monte Carlo algorithm using the explicitly discretized equations of motion 2.18 and 2.17. For this case, a new wavevector $\vec{k}^{n+1}$ is computed from the electric field at time t_n, $\vec{E}^n$. Knowing the value of $\vec{k}^{n+1}$, the new velocity $\vec{v}^{n+1}$ is found from the band structure information, and the new particle position is computed. Using a fixed timestep backward Euler discretization of the semiclassical equations yields:

$$\hbar(\vec{k}(t+h) - \vec{k}(t)) = h \cdot q \cdot \vec{E}(t+h) \tag{3.5}$$

$$\vec{r}(t+h) - \vec{r}(t) = h \cdot \vec{v}(t+h) \tag{3.6}$$

so that the the new $\vec{k}^{n+1}$ is computed from the new field $\vec{E}^{n+1}$, the new velocity $\vec{v}^{n+1}$ is computed from the material band structure, and the new particle position is fully implicit. Note that the electric field $\vec{E}^{n+1}$ is a function of the particle positions $\vec{r}^{n+1}$.

Equations 3.5 and 3.6 must be solved numerically at every timestep. Since the equations are nonlinear, a fixed point iteration or a Newton's method can be used. Both of these methods present difficulties. To ensure convergence of the fixed point iteration, it is usually necessary

to use as small a timestep as would be used in an explicit method. Instead using a Newton's method requires calculating derivatives. Additionally, the size of the problem to be solved, which is on the order of the number of particles in the simulation, can often be large, so that an iterative method, such as a non-linear relaxation technique, is a practical alternative. This type of relaxation would require using Newton's method inside the relaxation loop in order to solve the 3-dimensional problem for each particle.

3.3 Electric Field Calculations

The grid-based methods for computing electric fields that were described in the previous chapter will be discussed in detail in terms of the errors they introduce into the solution of Poisson's equation. The finite difference approximations introduced in Section 2.4 are evaluated. This is followed by a description of an alternate direct-force scheme for computing the electric field which involves breaking the electric field computation into two pieces, one due to the mobile charges and one due to the ionized impurity charges.

3.3.1 Error Estimates for Finite-Difference Approximations

The finite difference approximations to the first and second derivatives, given in Equation 2.21 and 2.25, can be evaluated in terms of the errors they present. Finite difference schemes stem from the definition of a derivative:

$$\left.\frac{\partial u}{\partial x}\right|_{x_0,y_0} = \lim_{\Delta x \to \infty} \frac{u(x_0+\Delta x, y_0) - u(x_0, y_0)}{\Delta x} \tag{3.7}$$

By using the forward and backward Taylor series expansion of a function about the point (x_0, y_0) we can find the finite-difference first derivative and its corresponding error series:

$$\frac{\partial u}{\partial x} = \frac{u(x_0+\Delta x, y_0) - u(x_0-\Delta x, y_0)}{2\Delta x} - \frac{\partial^3 u}{\partial x^3}\frac{\Delta x^2}{3!} - \dots \tag{3.8}$$

The first term of this series is the centered finite-difference first derivative which can be used to approximate the electric field in Equation 2.25. The remaining terms constitute the truncation error which is $O(\Delta x)^2$.

The centered finite difference approximation to the second derivative can similarly be found to be:

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x_0+\Delta x) - 2u(x_0, y_0) + u(x_0-\Delta x, y_0)}{(\Delta x)^2} - 2\frac{\partial^4 u}{\partial x^4}\frac{(\Delta x)^2}{4!} - \dots \tag{3.9}$$

Again, the first term is the numerical approximation to the derivative and the remaining terms describe the error series, which is $O(\Delta x^2)$.

Using the finite difference schemes proposed here, the local error in the potential will be on the order of the grid spacing squared in all dimensions. In order to obtain a reasonable solution to Poisson's equation, small grid sizes are needed. (Other issues are involved here, for example the carrier concentration or doping concentration gradient can effect the accuracy of the solution also.)

In their application to Monte Carlo simulation, mesh-based methods present their main difficulties in the charge assignment and force interpolation schemes previously discussed, rather than in the errors arising from the finite-difference approximations. Because these methods smooth out the charge-density, the local particle interactions may not be correctly captured. This is important in applications of Monte Carlo simulation where the particle-particle scattering is included.

3.4 Direct Force Scheme

Instead of using a mesh based method to compute the electric field, a scheme in which direct-forces are computed is proposed. The force on a particle is decomposed into contributions from the other mobile charges, the fixed ionized impurities and the charge on the contacts that represents the potential boundary conditions.

3.4.1 Particle-Particle Forces

The interaction between two particles of charge q_m and $q_m^{'}$ is found from the free-space Green's function. The scalar potential at $\vec{r}$ due to a particle of charge q_m located at the origin is:

$$\psi(\vec{r}) = \frac{q_m}{4\pi\epsilon_0|\vec{r}|} \tag{3.10}$$

and from the potential, the electric field at $\vec{r}$ due to the particle is:

$$\vec{E} = -\nabla\psi = \frac{q_m}{4\pi\epsilon_0}\frac{\vec{r}}{|\vec{r}|^3} \tag{3.11}$$

More generally, if the particle of charge q_m is located at $\vec{r}_0 = (x_0, y_0, z_0)$, it will create the following field at $\vec{r}' = (x', y', z')$:

$$\vec{E} = \frac{q_m}{4\pi\epsilon_0}\frac{(\vec{r}' - \vec{r}_0)}{[(x' - x_0)^2 + (y' - y_0)^2 + (z' - z_0)^2]^{\frac{3}{2}}} \tag{3.12}$$

so that it will impose the following force on a particle of charge $q_m^{'}$ located at $\vec{r}^{'}$:

$$\vec{F} = \frac{q_m q_m^{'}}{4\pi\epsilon_0} \frac{(\vec{r}^{'} - \vec{r}_0)}{[(x^{'} - x_0)^2 + (y^{'} - y_0)^2 + (z^{'} - z_0)^2]^{\frac{3}{2}}} \tag{3.13}$$

In order to compute the force on one particle, the contributions from each of the other charges must be summed. Clearly this calculation is cumbersome, being $O(N^2)$, where N is the number of particles.

3.4.2 Ionized Impurity Forces

The interactions between the doping charge and the mobile charged particles can be computed by performing a numerical volume integral over the volume containing the uniform charge. Given a volume charge with dimensions (a, b, c), the following integral must be evaluated to find the potential at a point $\vec{r} = (x, y, z)$:

$$\Psi(\vec{r}) = \frac{1}{4\pi\epsilon_o} \int_{-\frac{c}{2}}^{+\frac{c}{2}} \int_{-\frac{b}{2}}^{+\frac{b}{2}} \int_{-\frac{a}{2}}^{+\frac{a}{2}} \frac{N(x^{'}, y^{'}, z^{'}) dx^{'} dy^{'} dz^{'}}{[(x - x^{'})^2 + (y - y^{'})^2 + (z - z^{'})^2]^{\frac{1}{2}}} \tag{3.14}$$

where N is the ionized impurity concentration. Once the potential is found, the electric field forces on a particle can be computed numerically using a divided difference method. The numerical evaluation of the potential integral can be executed in two ways. One approach is that of discretizing the volume into a set of panels of sheet charge, for which an analytical solution for the potential is known, and computing the volume charge interaction with a single charge by summing the over all the panels. This method presents difficulties due to the singularities in the analytical solution, and a trade-off exists between requiring a small number of panels to compute the electric field accurately and the corresponding increase in the possibility of encountering problems with the singularity. A second method involves using a Gauss-Legendre quadrature method for solving the integral, along with a scheme for subtracting out the singularity in the integral. Both of these methods and their relative merits and difficulties are presented in Appendix A.

4

Simulation Algorithms

4.1 Introduction

This chapter presents a series of experiments used to verify the theory discussed previously. These experiments involve Monte Carlo simulation using the classical equations of motion. The framework for these simulations is outlined. Each of the experiments presents a time-dependent problem, which can be integrated in several ways. Both implicit and explicit discretizations of the relevant motion equations are considered. The accuracy and stability of all the simulations are discussed.

4.1.1 Experimental System

The simulation domain used for study is a system in which particles are simulated in three-dimensions. The charges are subjected to electric field forces due to the other mobile charges and the fixed ionized impurity charges, as well as the potential boundary conditions on the contacts.

The transport model for the particles is based on Newton's laws of motion, which are:

$$\frac{d\vec{r}_i(t)}{dt} = \vec{v}(t) \tag{4.1}$$

$$\frac{d\vec{v}_i(t)}{dt} = \frac{\vec{F}}{m} \tag{4.2}$$

where $\vec{r}_i$ is the i^{th} particle position, $\vec{v}_i$ is the i^{th} particle's velocity, m is the particle's mass, and $\vec{F}$ is the force acting on the particle. Assuming a parabolic band structure, the simulation of the classical motion equations is easily related to the simulation of the semi-classical motion

equations. For the semiclassical case, a particle's wavevector is found from the velocity:

$$\vec{k} = \vec{v}\frac{m^*}{\hbar^2} \tag{4.3}$$

4.1.2 Time Evolution of Experimental System

In order to solve the Equations 4.1 and 4.2 numerically, they must be discretized. For a forward Euler discretization, the equations become simply:

$$\vec{v}_i(t+h) = \vec{v}_i(t) - \frac{hq}{m}\vec{E}(r_1(t)\cdots r_n(t)) \tag{4.4}$$

$$\vec{r}_i(t+h) = \vec{r}_i(t) + h\vec{v}_i(t)) \tag{4.5}$$

for the i^{th} particle. Since there are two equations to be solved, the velocity can be updated first, yielding the following modified forward Euler formulation:

$$\vec{v}_i(t+h) = \vec{v}_i(t) - \frac{hq}{m}\vec{E}(r_1(t)\cdots r_n(t)) \tag{4.6}$$

$$\vec{r}_i(t+h) = \vec{r}_i(t) + h\vec{v}_i(t+h)) \tag{4.7}$$

These equations must be solved at every timestep for the new position and velocity of each of the two particles. Note that the electric field $\vec{E}(t)$ is a straightforward calculation since it only depends on the position of all of the particles at time t, $\vec{r}_1(t)\cdots\vec{r}_n(t)$. Both of the above formulations are explicit.

Using the backward Euler discretization scheme on Equations 4.1 and 4.2 yields:

$$\vec{r}_i(t+h) = \vec{r}_i(t) + h\vec{v}_i(t+h) \tag{4.8}$$

$$\vec{v}_i(t+h) = \vec{v}_i(t) - \frac{hq}{m}\vec{E}(r_1(t+h)\cdots r_n(t+h)) \tag{4.9}$$

and for the trapezoidal rule we get:

$$\vec{r}_i(t+h) = \vec{r}_i(t) + \frac{h}{2}[\vec{v}_i(t) + \vec{v}_i(t+h)] \tag{4.10}$$

$$\vec{v}_i(t+h) = \vec{v}_i(t) - \frac{hq}{2m}\left[\vec{E}(r_1(t)\cdots r_n(t)) + \vec{E}(r_1(t+h)\cdots r_n(t+h))\right] \tag{4.11}$$

Both of these sets of equations are implicit non-linear equations and must be solved at each time-step. Since there is a non-linear system of equations associated with each particle, a relaxation technique is utilized to iterate for the particle position updates at each timestep. The Gauss-Seidel relaxation scheme used in these experiments works as follows: consider each

of the particles individually. For each particle, solve for the position update vector assuming that all of the other particles are fixed in space. Then update that particle's position and go to the next particle. Assume for the next particle that the previous particle has moved; recalculate the necessary forces. Continue this iteration through all of the particles until the relaxation converges, which occurs if the particles' positions change an amount less than some tolerance, ϵ.

Within the relaxation scheme, the update for each individual particle requires solving a three-dimensional problem, namely representing the updates in particle position in all three dimensions, x, y and z. Since this problem is also non-linear, a 3-dimensional Newton's method is employed. An N-dimensional Newton's method is used to solve N non-linear equations $\vec{F}(\vec{u}_i)$ for a vector $\vec{u} \in \Re^N$ such that:

$$|\vec{F}(\vec{u}_i)| = 0. \tag{4.12}$$

This method requires solving the following equation for each Newton iteration:

$$J_F \vec{\delta u} = -F(\vec{u}) \tag{4.13}$$

where J_F is a Jacobian matrix of derivatives and $\vec{\delta u}$ is the vector of corrections to $\vec{u}$. A description of this method and the Jacobian terms is given in in Appendix B.

The nonlinear problem to be solved is based on the discretization. For the classical motion equations, the 3-dimensional non-linear problem for the i^{th} particle is:

$$\begin{aligned} \vec{F}(r_i(t+h)) &= \vec{r}_i(t+h) - \vec{r}_i(t) \\ &\quad -h\left[\vec{v}_i(t) - \frac{hq}{m}\vec{E}(r_1^k(t+h)\cdots r_{i-1}^k(t+h), r_i^{k-1}(t+h)\cdots r_n^{k-1}(t+h))\right] = 0 \end{aligned} \tag{4.14}$$

for the backward Euler discretization and is:

$$\begin{aligned} \vec{F}(r_i(t+h)) &= \vec{r}_i(t+h) - \vec{r}_i(t) \\ &\quad -\frac{h}{2}\left[2\vec{v}_i(t) - \frac{hq}{2m}\left(\vec{E}(r_1^k(t)\cdots r_{i-1}^k(t), r_i^{k-1}(t)\cdots r_n^{k-1}(t))\right)\right. \\ &\quad \left. - \frac{hq}{2m}\left(\vec{E}(r_1^k(t+h)\cdots r_{i-1}^k(t+h), r_i^{k-1}(t+h)\cdots r_n^{k-1}(t+h))\right)\right] = 0 \end{aligned} \tag{4.15}$$

for the trapezoidal discretization, where k represents the relaxation iteration number. A description of the Jacobians associated with Equations 4.14 and 4.15 are also given in Appendix B.

The details of the simulation procedure for the explicit direct-force computation are given in Algorithm 1, and the details of the Newton-relaxation simulation algorithm for the implicit

Algorithm 1: Explicit direct-force algorithm.
Initialize particle positions, $\vec{r}_i^0$, velocities $\vec{v}_i^0$.
For k = 1 to number_of_timesteps {
    For i = 1 to number_of_particles {
        Compute E_i contribution from the doping charge.
        Compute E_i contribution from the mobile charges.
        Update velocity, $\vec{v}_i(t+h) = \vec{v}_i(t) + \frac{hq}{m}\vec{E}_i$.
        Update position, $\vec{r}_i(t+h) = \vec{r}_i(t) + h\vec{v}_i(t+h)$.
    }
}

(trapezoidal) direct-force computation is given in Algorithm 2. Comparisons were made with the traditional explicit CIC algorithm which is given in Algorithm 3. Clearly the differences in Algorithms 1 and 3 are only in the electric field computation. The implicit CIC routine, shown in Algorithm 4, is similar to Algorithm 3, with the two direct force electric field calculations substituted with the CIC calculation. Note that in using an implicit CIC algorithm, the CIC computation will lie within both the relaxation and Newton iterations, so that a matrix solve will have to be computed within both loops.

4.2 Simulation Details

The simulation domain is a 0.1 μm cube, and in the simulation experiments 50, 100, or 300 particles are used to represent the mobile charge density. The ionized impurity interactions for the direct-force algorithms are computed using the Gauss quadrature method described in Appendix A. It was determined that 10 quadrature points were required to form the basic set of quadrature weights and abscissas, which means that $(10)^3$ evaluations per calculation are made. The scattering was ignored for the purposes of removing any energy-damping mechanisms which would cover the instabilities in the numerics.

4.2.1 Boundary Conditions

Typically, Monte Carlo simulations must allow for two types of boundaries: ohmic contacts and the side walls of the device. The non-contact surfaces are treated as purely reflecting boundaries. The algorithms used here do not account for ohmic contacts. Particles are simply reflected from all boundaries.

Algorithm 2: Implicit direct-force algorithm.
Initialize particle positions, $\vec{r}_i^0$, velocities $\vec{v}_i^0$.
For k = 1 to number_of_timesteps of size h {
    For i = 1 to number_of_particles N
        Compute explicit position, $\vec{r}_i(t+h) = \vec{r}_i(t) + h\vec{v}_i$.
    For j = 1 to number_of_relaxation_iterations {
        For i = 1 to number_of_particles N {
            For n = 1 to number_of_Newton_iterations {
                Compute E_i contribution from the doping charge.
                Compute E_i contribution from the mobile charges.
                Compute Jacobian matrix from equations B.9 and B.10.
                Compute Newton right hand side from equation 4.15.
                Compute position update, $\vec{\delta}_i$.
                Update position, $\vec{r}_i(t+h) = \vec{r}_i(t+h) + \vec{\delta}_i$.
                Correct for boundary conditions.
            }
        }
    }
    For i = 1 to number_of_particles N
        Update velocity, $\vec{v}_i(t+h) = \vec{v}_i(t) - \frac{hq}{2m}\left[\vec{E}_i(t) + \vec{E}_i(t+h)\right]$.
}

Algorithm 3: Explicit cloud-in-a-cell (CIC) algorithm.
Initialize particle positions, $\vec{r}_i^0$, velocities $\vec{v}_i^0$.
For k = 1 to number_of_timesteps {
    For i = 1 to number_of_particles {
        Compute particle weights onto the mesh.
        Compute contribution of particle i to the mesh charge density.
    }
    Solve the system of mesh equations for potential Φ with ICCG.
    For i = 1 to number_of_particles {
        Interpolate mesh potential to get the electric field E_i.
        Update particle velocity, $\vec{v}_i(t+h) = \vec{v}_i(t) + \frac{hq}{m}\vec{E}_i$.
        Update particle position, $\vec{r}_i(t+h) = \vec{r}_i(t) + h\vec{v}_i(t+h)$.
    }
}

Algorithm 4: Implicit CIC algorithm.
Initialize particle positions, $\vec{r}_i^0$, velocities $\vec{v}_i^0$.
For k = 1 to number_of_timesteps of size h {
    For i = 1 to number_of_particles N
        Compute explicit position, $\vec{r}_i(t+h) = \vec{r}_i(t) + h\vec{v}_i$.
    For j = 1 to number_of_relaxation_iterations {
        For i = 1 to number_of_particles N {
            For n = 1 to number_of_Newton_iterations {
                For m = 1 to number_of_particles {
                    Compute particle weights onto the mesh.
                    Compute contribution of particle m to the mesh charge density.
                }
                Solve the system of mesh equations for potential Φ with ICCG.
                Interpolate mesh potential to get the electric field E_i.
                Compute Jacobian matrix from equations B.9 and B.10.
                Compute Newton right hand side from equation 4.15.
                Compute position update, $\vec{\delta}_i$.
                Update position, $\vec{r}_i(t+h) = \vec{r}_i(t+h) + \vec{\delta}_i$.
                Correct for boundary conditions.
            }
        }
    }
    For i = 1 to number_of_particles N
        Update velocity, $\vec{v}_i(t+h) = \vec{v}_i(t) - \frac{hq}{2m}\left[\vec{E}_i(t) + \vec{E}_i(t+h)\right]$.
}

It is important to note that the implementation of the boundary conditions can change depending on the discretization scheme used. For an explicit calculation, particles will either be removed from the simulation if they leave the device through an ohmic contact or reflected from a wall if they leave the device through a non-contact boundary. With an implicit calculation, the issue of the boundary becomes more difficult, since the particle position is refined on every numerical iteration of the relaxation and the Newton algorithm. In all cases, the boundary conditions are implemented such that if the first guess at the particle position is outside the device, the particle position will be refined at each relaxation/Newton iteration before it is placed back in or removed from the simulation domain.

The reflecting boundaries are implemented as follows: If the projected position of a particle is outside the simulation domain, the following algorithm is used to determine the new position of the particle. The intersection point between the line formed by the old and new position with the plane of the boundary is determined. The particle is positioned at that point. To ensure a perfect reflection, the normal component of the velocity relative to the surface the particle hit, is negated. This way the angle of incidence is the same as the angle of reflection and no energy is lost at the wall.

Finding the intersection point with the boundary involves finding the time relative to the total timestep, at which the particle hits the wall. The particle trajectory between the current and projected positions is given by the parametric equations:

$$\begin{aligned} x(t) &= x_0 + (x_1 - x_0)t \\ y(t) &= y_0 + (y_1 - y_0)t \\ z(t) &= z_0 + (z_1 - z_0)t \end{aligned} \tag{4.16}$$

where (x_0, y_0, z_0) is the old position of the particle and (x_1, y_1, z_1) is the new position lying outside the simulation domain. The time of intersection with each bounding plane is computed by solving the parametric equations for t. The intersection points with the planes at $y = \pm y_{max}$ for example are determined by solving the equation:

$$t = \frac{\pm y_{max} - y_0}{y_1 - y_0}. \tag{4.17}$$

Four values of t are computed, corresponding to the four bounding planes $y = \pm y_{max}$ and $z = \pm z_{max}$. Any value of t which is not in the interval $[0, 1]$ indicates that no intersection with the corresponding bounding plane occurs in this timestep. Of the remaining values of t, the minimum one is selected since that indicates the plane of first intersection.

4.2.2 System Simulation Parameters

Since the algorithms presented above will be compared on the basis of accuracy and stability, the simulations are evaluated in terms of average temperature and current. The system temperature T is computed by averaging over all of the particles, as follows:

$$\frac{1}{2}m_i^*\langle v_i - \langle v_i\rangle\rangle^2 = \frac{3}{2}k_B T_i \tag{4.18}$$

where k_B is Boltzmann's constant, $\langle v\rangle$ is the average velocity of the particle and v is the velocity of the particle. The average velocity of the i^{th} particle is computed so that at time t':

$$\langle v_i\rangle = \frac{\sum_{t=0}^{t'} v_i(t)}{t'}. \tag{4.19}$$

With no applied field the average velocity eventually should become zero. It will always be close to zero if a large number of particles are used, so that the temperature is determined primarily by the average random velocity of the particles.

Simulation accuracy is measure from the quantity I, which is a normalized measure of the particle "current". Instead of counting particles that enter and leave the simulation domain through contacts, the number of particles that are reflected off opposing boundaries of the domain, denoted P and N, are counted. In the Results chapter, the current refers to the difference between the particle counts P and N divided by the number of simulation timesteps k. This "charge" per unit time then defines a normalized current as follows:

$$I = \frac{P - N}{k}. \tag{4.20}$$

For the problem solved in this thesis, the average current should be zero. In a transient calculation, this calculation would probably be "windowed" over some number of timesteps, so that the current would reflect an instantaneous rather than an average quantity.

5

Results

This chapter presents results from simulations of the test system described in the previous chapter. By varying simulation parameters, such as numerical timestep and the number of mobile charges, comparisons can be made between implicit and explicit time integration techniques. Results are described in terms of terminal currents, average velocity, and temperature.

5.1 A One-Particle Example

A simple one-dimensional set-up can be created to demonstrate the differences between various multi-step methods. Consider integrating the motion of a mobile unit charge positioned off center between two fixed boundary charges. Ideally the mobile particle will oscillate in the space between the two boundaries. In a system with no energy loss implemented, the oscillation should be of constant amplitude, since the total energy of the particle is determined by the potential energy it has the beginning of the simulation. This system is depicted in Figure 5-1.

Figure 5-2 shows the unit particle's position between the two boundaries as a function of time using an explicit forward Euler integration method for several different numerical timesteps.

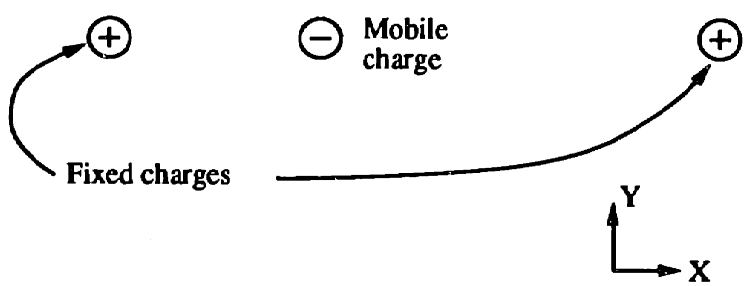

FIGURE 5-1: Physical setup for single particle experiments.

As expected, for large timesteps, the oscillation will grow in time, indicating that the charge is picking up energy due to the instability of the numerical method. For small enough timesteps, the numerical solution to the particle's motion will approach that of the ideal solution, so that by decreasing the timestep, the rate of growth of the motion amplitude can be slowed.

Figure 5-3 shows the particle motion when integrated with the backward Euler method. Clearly the solution demonstrates no instability for large timesteps, however the decay in the amplitude of oscillation is due to the energy-loss property of the backward Euler method. As with the forward Euler case, the accuracy will improve with decreasing timestep, and will approach the exact solution. The solution found using the trapezoidal rule, also an implicit method, is given in Figure 5-4. The solution is very close to the exact solution for all of the esteps used in this demonstration, as expected, since the numerical method is both energy-preserving and stable for any timestep. For extremely large timesteps, the trapezoidal solution becomes inexact, as shown in Figure 5-5.

5.2 Explicit MC Simulation

By using explicit methods to integrate particle motion, it is expected that some system parameter should indicate whether there is growth in the overall energy in the system. The average temperature of a system can be found over the simulation time using the technique described in Chapter 4, and in some sense is a measure of this overall energy. Figures 5-6,5-7, and 5-8 show the ensemble temperatures over time for a system with 50, 100 and 300 particles, respectively. This temperature is normalized to a good approximation of the exact answer, which is the temperature found using a very small timestep.

As expected, the temperature shows growth over time, with the speed of growth increasing with a corresponding increase in timestep size. For a small enough timestep, the solution is stable and approaches the exact answer. Some degree of noise on the solution is due to the small number of particles being used in the simulation. This noise does decrease as the number of particles is increased. However the growing nature of the temperature for large timesteps does not improve with increasing number of particles.

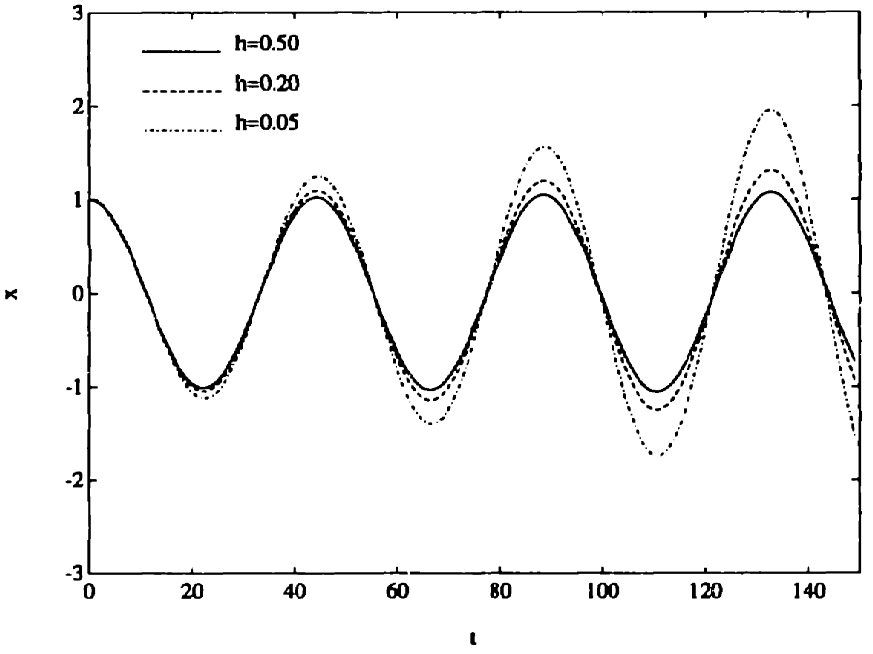

FIGURE 5-2: Position of a single particle integrated with the forward Euler method.

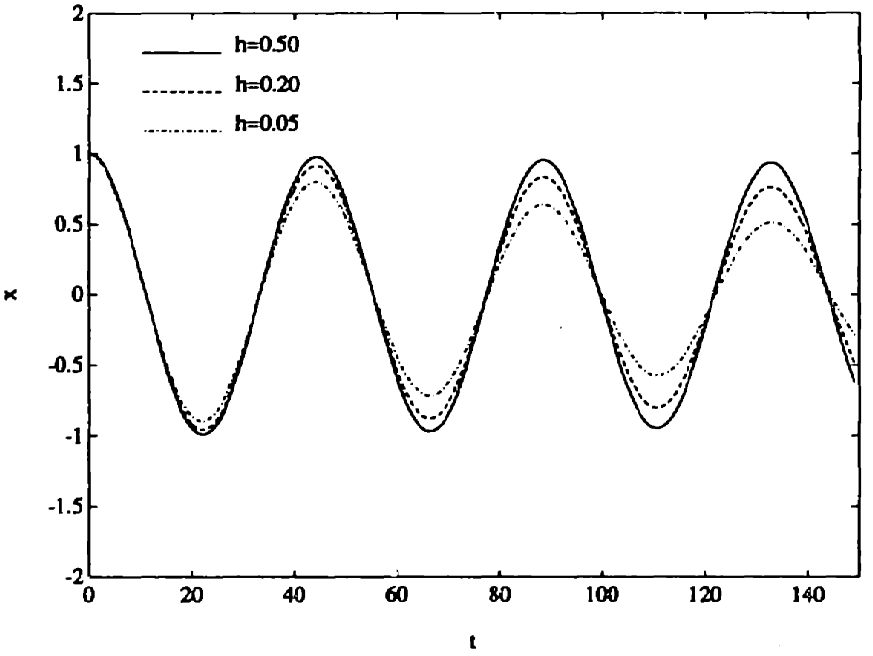

FIGURE 5-3: Position of a single particle integrated with the backward Euler method.

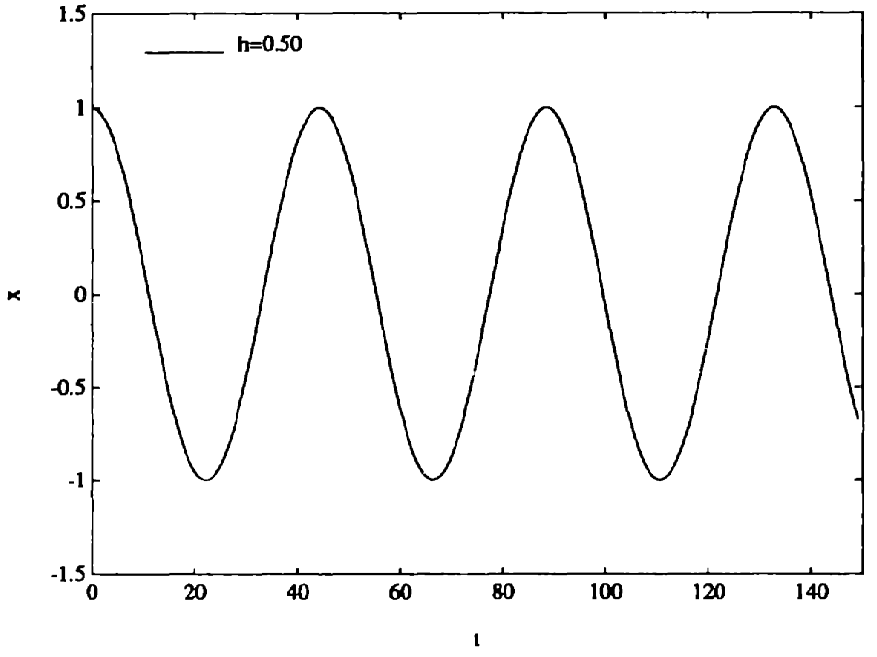

FIGURE 5-4: Position of a single particle integrated with the trapezoidal method.

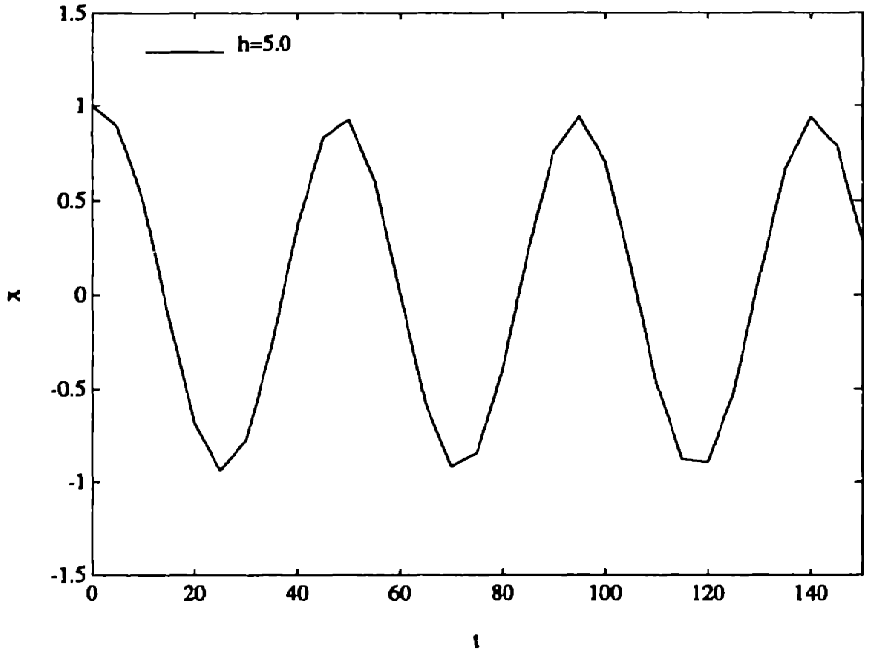

FIGURE 5-5: Position of a single particle integrated with the trapezoidal method for a very large timestep.

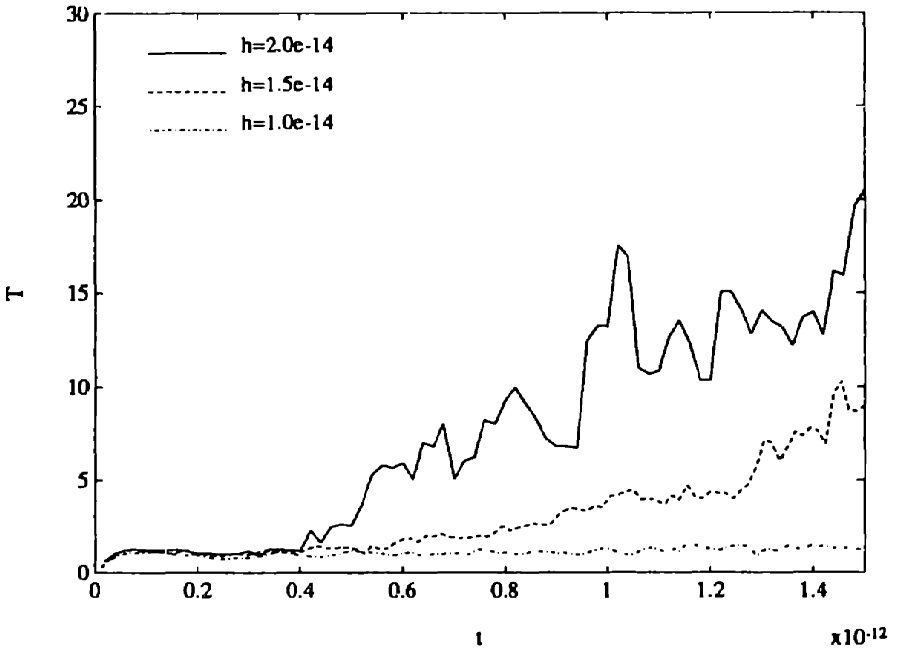

FIGURE 5-6: Normalized average temperature over time for a 50 particle ensemble using explicit integration.

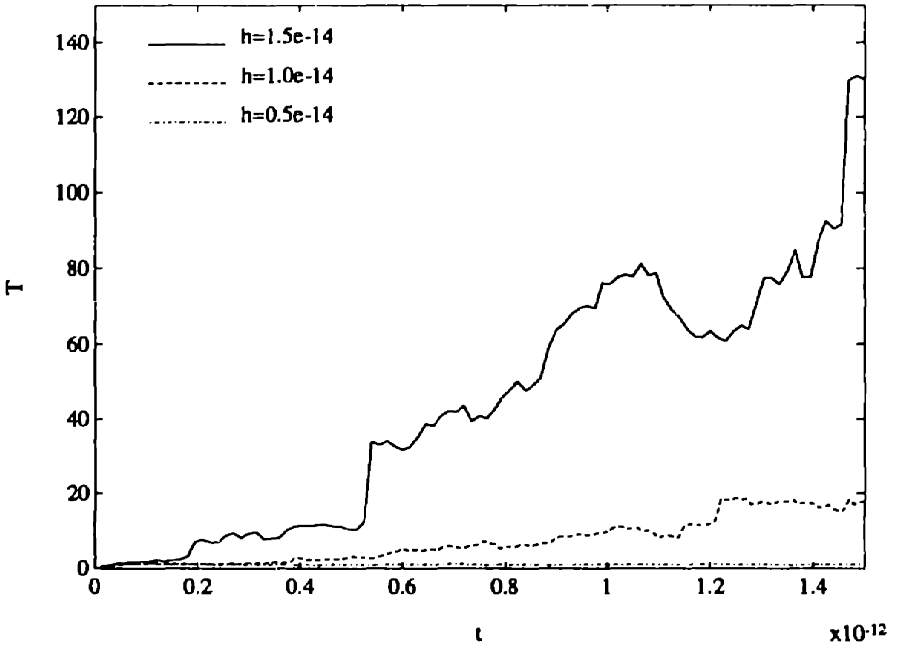

FIGURE 5-7: Normalized average temperature over time for a 100 particle ensemble using explicit integration.

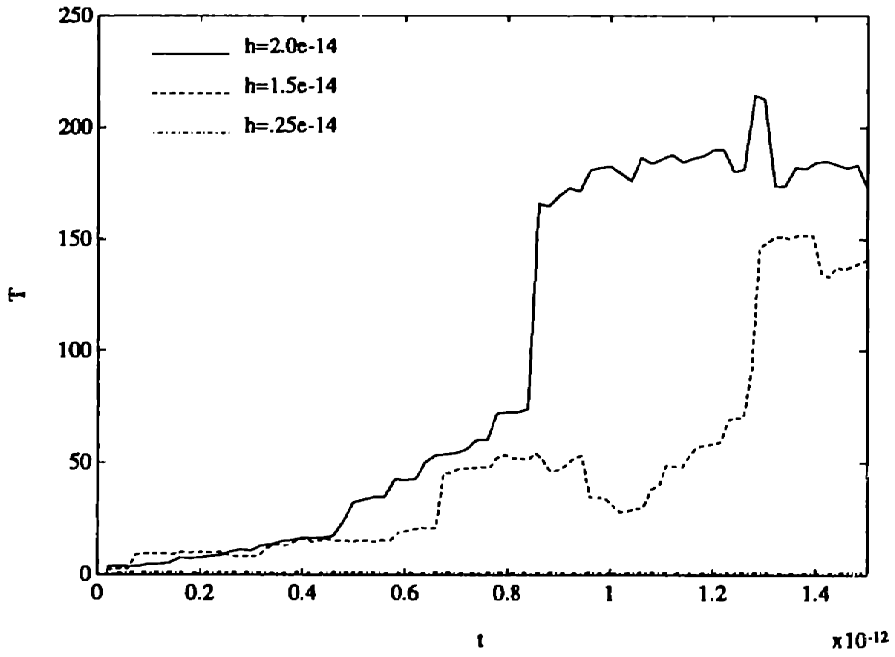

FIGURE 5-8: Normalized average temperature over time for a 300 particle ensemble using explicit integration.

5.3 Implicit MC Simulation

By using an implicit method for the time integration, such as the trapezoidal rule, the ensemble motion should be energy-preserving. The temperature is expected to demonstrate no growth over time beyond the initial settling time. Indeed, as Figures 5-9 and 5-10 indicate, the temperature of the ensemble is fairly steady over time. By decreasing the timestep, the variations in the solution can be reduced, although some noise is unavoidably caused by the small number of particles in the system. This is indicated by the deviation in the solution from the mean in the two figures. Figure 5-11 shows that a large number of particles is needed to improve the accuracy in the solution and reduce the size of the fluctuations from the mean of the temperature.

5.4 Accuracy of the Methods

A rough indication of the accuracy of the simulation can be found by counting the number of particles that are reflected off the opposing walls of the box. Dividing the difference in these counts over time by the number of simulation timesteps, we have a normalized approximation

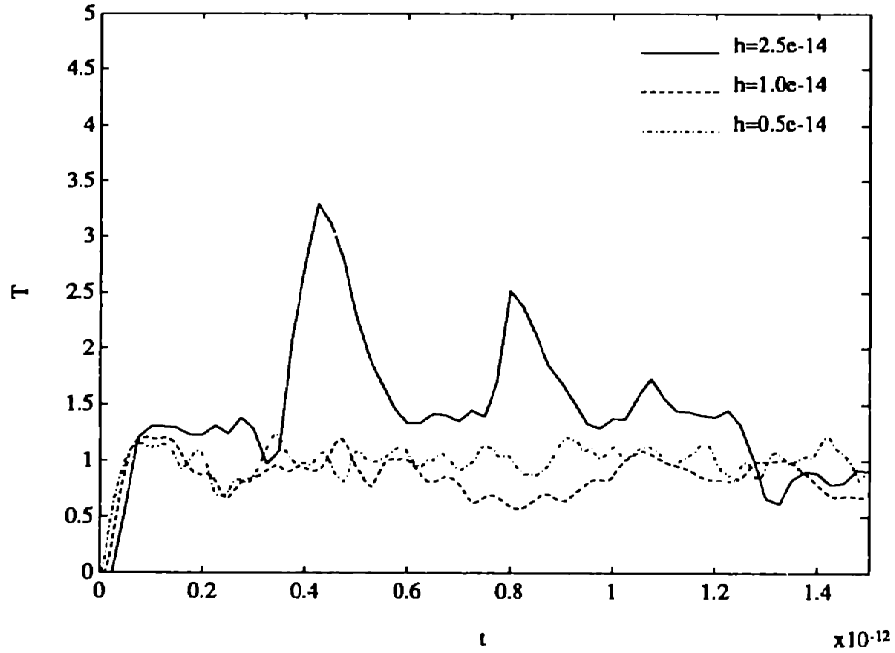

FIGURE 5-9: Normalized average temperature over time for a 50 particle ensemble using trapezoidal integration.

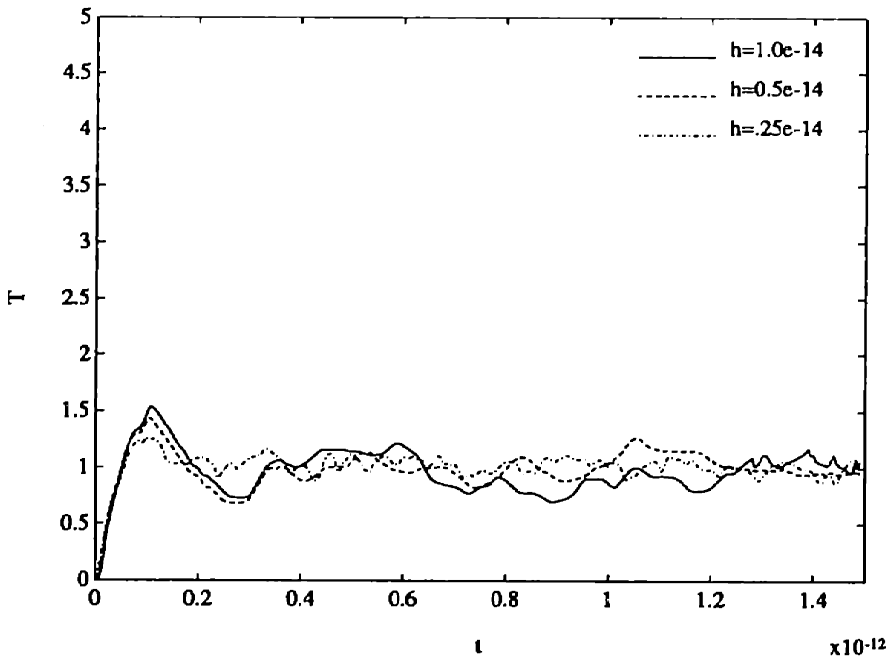

FIGURE 5-10: Normalized average temperature over time for a 100 particle ensemble using trapezoidal integration.

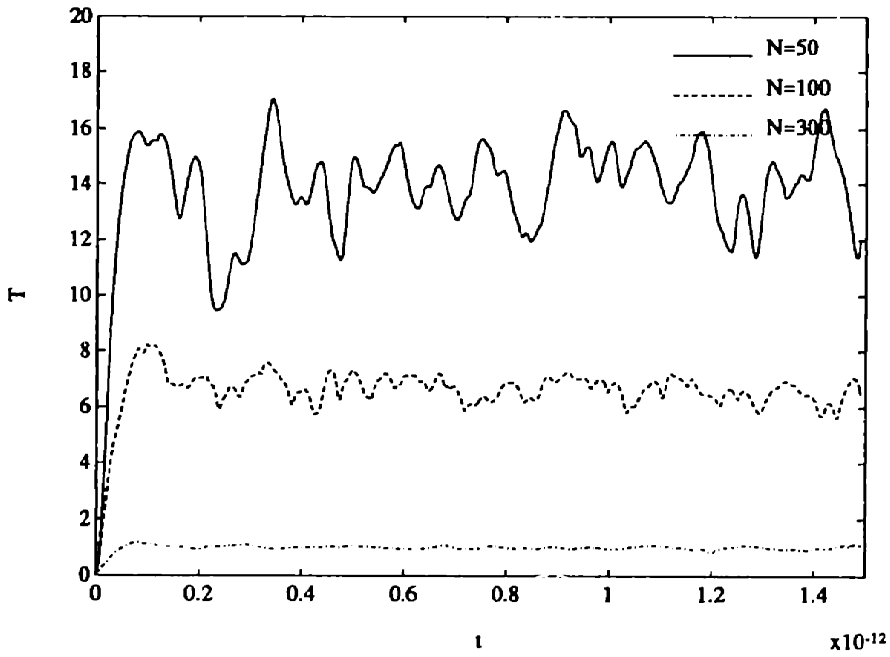

FIGURE 5-11: Normalized average temperature over time for a particle ensemble using trapezoidal integration and timestep h=0.5e-14.

of the current if those sides were contacted. Figures 5-12, 5-13 and 5-14 show this calculation over the simulation time of an explicit simulation for ensembles containing 50, 100, and 300 particles respectively.

The accuracy of an explicit method will improve with decreasing timestep. As the timesteps are decreased the currents deviate less and less from the exact solution (of zero), indicating a more accurate simulation. As expected, using larger ensemble of particles also improves the accuracy of this result.

The current derived using implicit methods showed an improvement in terms of accuracy over those calculated using explicit methods. Figure 5-15 shows the best calculated explicit current, and the current calculated with three trapezoidal solutions using different timesteps. The difference between explicit and implicit methods can be seen more clearly in Figures 5-16, 5-17, and 5-18, each of which show both the explicit and the implicit calculation of current for timesteps 2.5e-14, 1.0e-14 and 0.5e-14, respectively, for a 50 particle system.

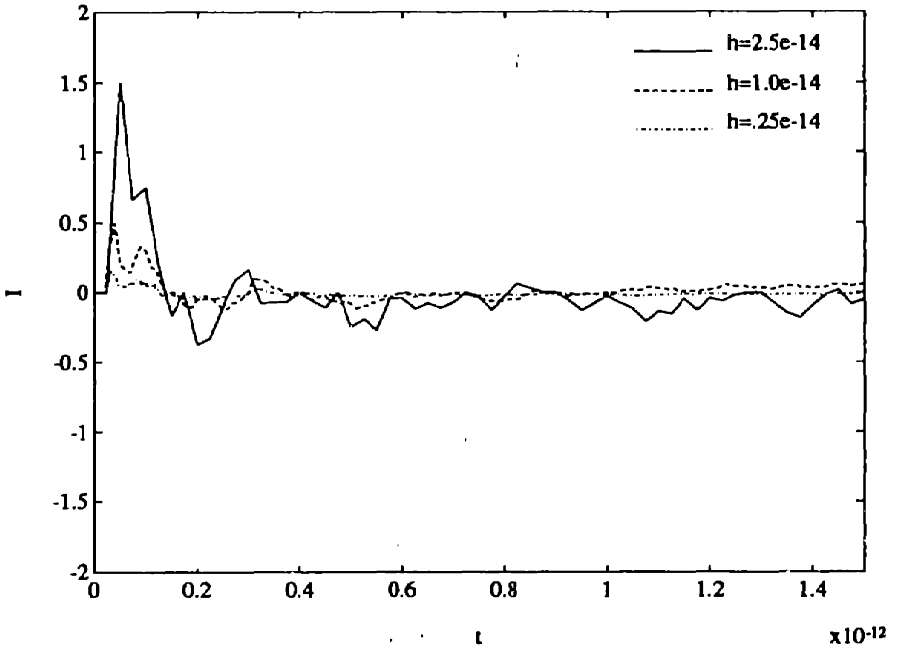

FIGURE 5-12: Normalized "current" over time for a 50 particle ensemble using explicit integration.

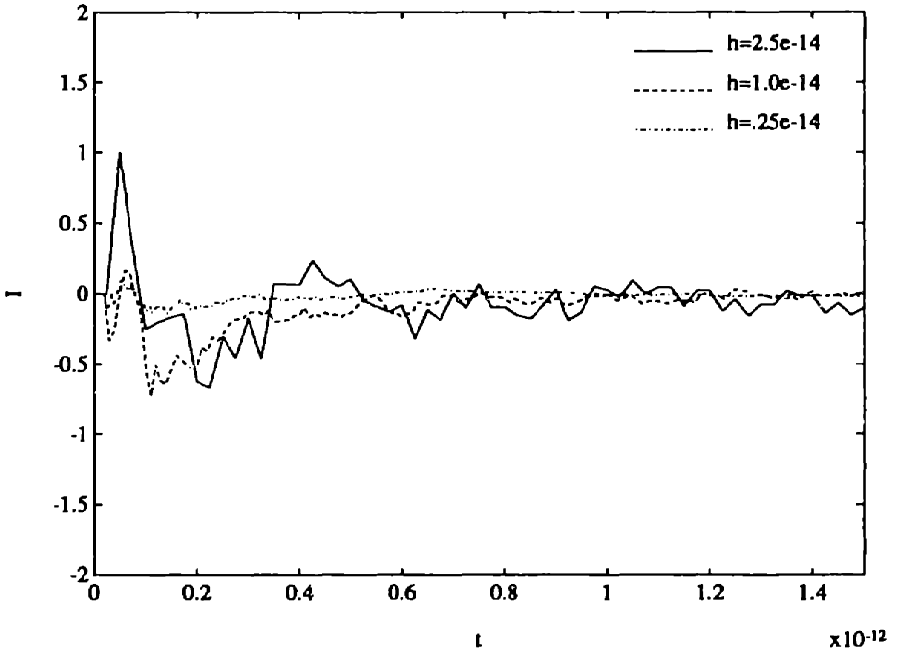

FIGURE 5-13: Normalized "current" over time for a 100 particle ensemble using explicit integration.

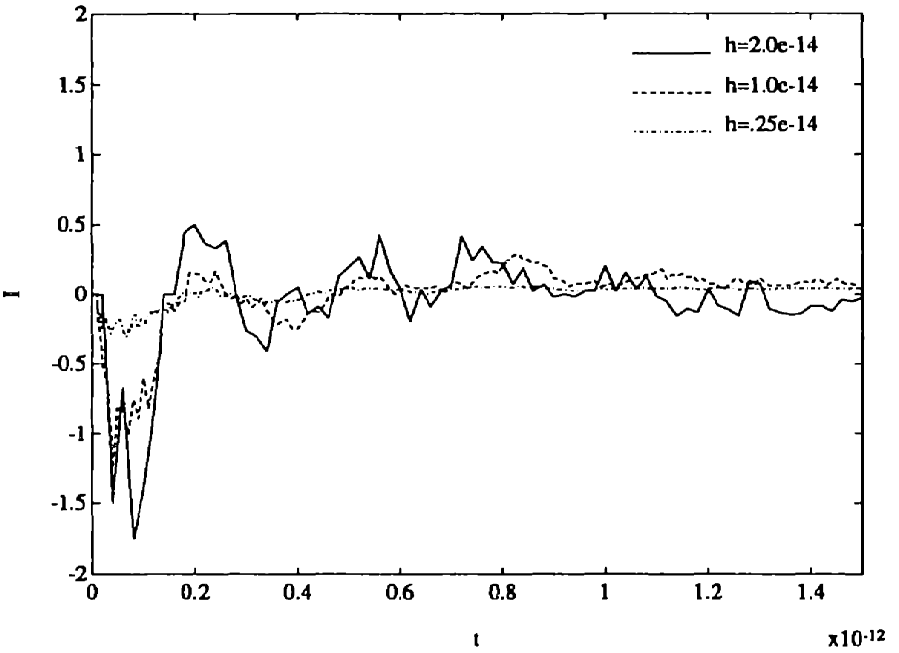

FIGURE 5-14: Normalized "current" over time for a 300 particle ensemble using explicit integration.

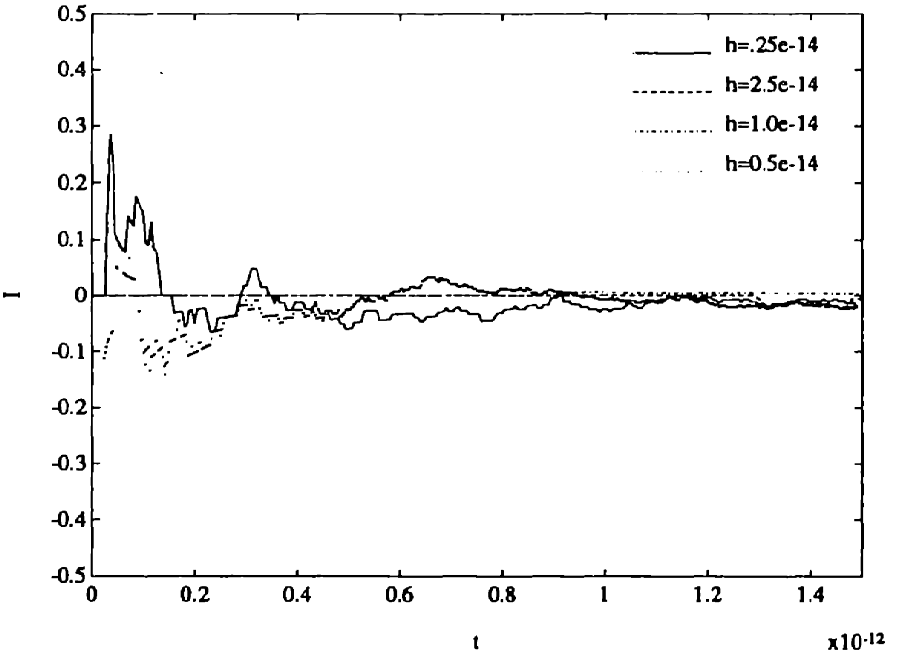

FIGURE 5-15: Normalized "current" over time for a 50 particle ensemble. The explicit solution is shown in a solid line. All other solutions results from trapezoidal integration.

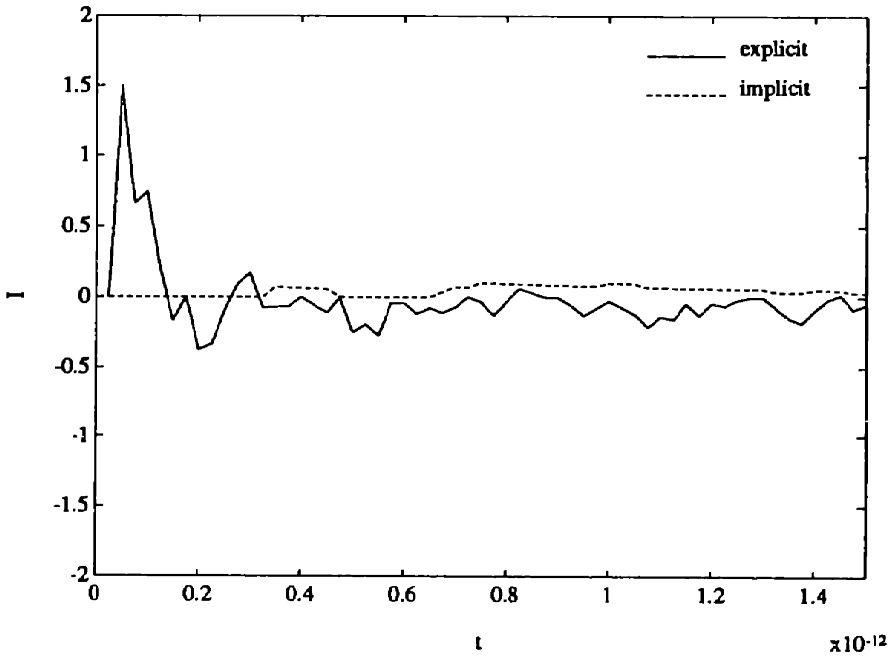

FIGURE 5-16: Normalized "current" over time for a 50 particle ensemble for h=2.5e-14.

5.5 Convergence of Numerical Algorithms

The algorithms presented in the previous chapter for solving the implicit system of equations all involve several numerical techniques, namely a relaxation and a Newton's method, to compute the particle updates, and a quadrature method to solve for the potential integral due to the volume charge. The convergence of these algorithms is discusses here.

The relaxation algorithm, which is used to solve for the particle positions in the implicit formulations, typically requires 3 to 4 passes over all of the particles to converge with a position resolution of 1 part in 10^4. A Newton method is contained within the relaxation, in order to solve for the individual particle updates. This converges quadratically when all Jacobian terms are included, and only one Newton iteration per step of the relaxation iteration is required. To save computation time, the off-diagonal Jacobian terms can be approximated; however, this scheme requires 2 Newton iterations per relaxation step.

In order to correctly compute the volume integral of the potential and the electric field, the Gaussian quadrature method requires about 10 points per dimension. With this number of quadrature points, the potential and electric field close to the domain edges, which present the most difficulty, can be correctly evaluated. (When the evaluation point is close to the edge, less of the potential is found with the analytic solution.) The fact that difference locations in

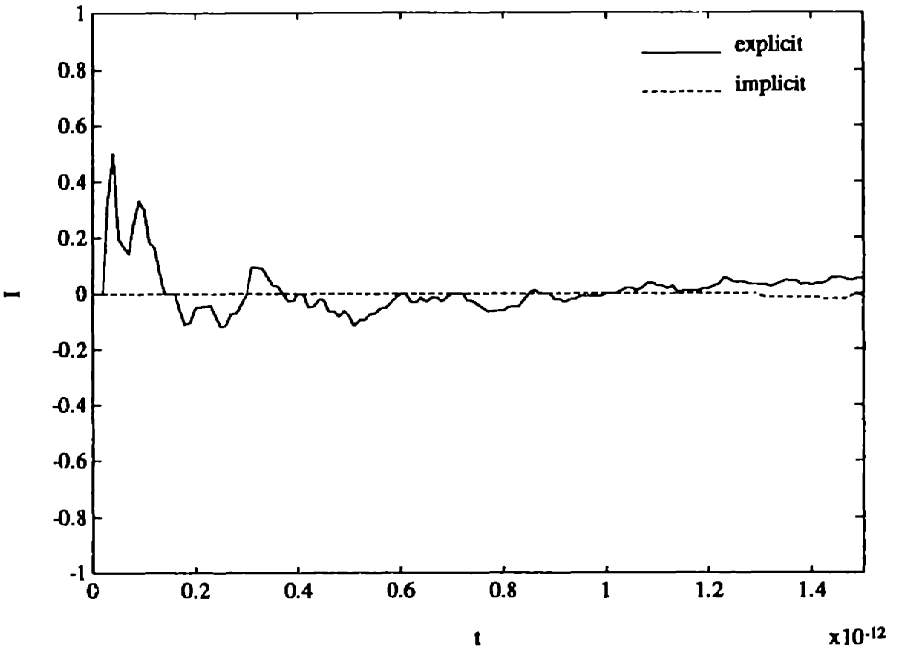

FIGURE 5-17: Normalized "current" over time for a 50 particle ensemble for h=1.0e-14.

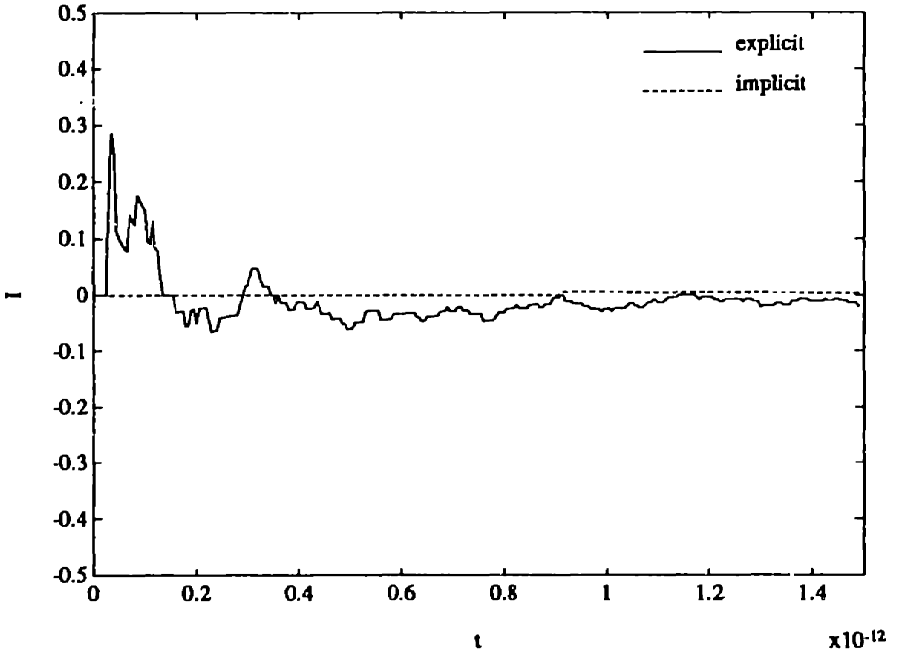

FIGURE 5-18: Normalized "current" over time for a 50 particle ensemble for h=0.5e-14.

the volume require different numbers of quadrature points was not exploited, however, future versions of the simulator would either utilize this fact or tabulate the integrals.

6

Conclusion

This thesis was concerned with studying methods for improving the numerical efficiency of Monte Carlo device simulation. Several numerical techniques for integrating the equations of motion of the simulation particles were investigated in the context of using direct electric field calculations to compute the particle forces. In particular two simulators were implemented: The first used an explicit scheme to compute the particle velocities and positions. The second used an implicit scheme, the trapezoidal method, in which both the velocities and positions are calculated using information from both future and current timesteps. These methods are discussed below.

6.1 Summary of Results

The first experiments of this research were designed to compare implicit and explicit methods of integrating particle motion. Preliminary results indicate that for a single charge system, implicit methods (specifically the trapezoidal method) can accurately track the particle's position, whereas the explicit simulation produced a growth in the particle's position and energy over time.

For simulations of an ensemble of particles, results were expected in which implicit methods would show less timestep dependence and more accuracy than explicit methods. In order to make the comparison, the total energy of the system over time was examined. The energy metric used was the average temperature of the particle ensemble. It was demonstrated that for explicit methods, the total energy of the system grows without bound for large timesteps, although for small enough timesteps, the solution will approach that of the ideal solution. However, by using implicit methods, the total energy of the system remained stable over time,

so that the temperature was bounded.

These results verify that by using an explicit method, the electric field calculated to determine the particle's movement can easily be incorrect, since the field will actually vary over the timestep period for large enough timesteps. An implicit method on the other hand, essentially makes a correction to this error by using the average electric field over the time-step, thus making it more un-likely that a particle will ever get too close (and see a large force from) another particle. The combined relaxation/Newton technique used to solve the implicit problem was found to be robust, and converges within a few iterations of the relaxation (combined with 1 Newton iteration.)

It was demonstrated that increasing the number of simulation particles did not change the stability of the time-integration problem, but did smooth out the variations in the simulation results (e.g. temperature and current.) A rather large number of particles was actually required to accurately simulate these systems, although a small number of particles can be used to show the trends in the numerical methods.

Some qualitative comparisons were made between the CIC methods and the direct-force methods, although no really detailed simulations allowing for a more quantitative comparison were run. As proposed, the CIC methods smooth-out or miss some of the local forces between particles. For a large number of simulations, the number of particles were counted which landed within a distance of each other that was about one tenth of the CIC mesh-spacing. In all cases this count was higher in the direct-force simulations than in the CIC simulations, indicating that the CIC methods might miss important local interactions. The following table gives some data relating the percentage increase in particle interactions using CIC methods over direct methods for different number of total particles.

N	%
50	62%
100	44%
300	16%
500	14%
1000	10%

6.2 Future Work

Because Monte Carlo methods encapsulate such a broad range of numerical techniques, there are many issues involved in the numerics of Monte Carlo device simulation. Some of them have been discussed in this thesis, and many remain to be researched. Numerical issues involving the physics, such as details involving scattering integrals or band structures, were not pursued.

Some issues to be resolved directly from this thesis include methods for improving the speed at which the direct particle-particle forces are calculated. Algorithms exist for this purpose; specifically multipole accelerated field calculations can be performed in time $O(N)$ time rather than $O(N^2)$ time obtained for the direct method [23, 24]. Additionally, the potential integral used to calculated the doping-particle forces, presently uses 10^3 quadrature points, or potential evaluations, per integral. Since the integral is only a function of space, it can be tabulated in order to save computation time.

Although explicit and implicit methods have been compared in the context of direct-force electric field calculations, a more quantitative comparison between the explicit CIC (standard) approach and the implicit direct-force calculation needs to be completed. Qualitatively, the benefits of the direct-force calculation are apparent, since local forces are better represented. However, a more complete comparison would be insightful. Additionally, investigating higher-order explicit schemes might yield methods that are less computationally expensive than the implicit methods.

Finally, the methods suggested in this thesis still must be incorporated into a full Monte Carlo simulator which includes the physics that have been excluded here. The speed enhancements described above would need to be completed first, in order to make this feasible.

Bibliography

[1] E. Venturi, E. Sangiorgi, R. Brunetti, W. Quade, C. Jacboni, and B. Ricco. An Efficient Monte Carlo Simulator for High-Energy Electrons and Holes in MOSFET's. In *Proceedings of the IEDM*, 1990.

[2] Franco Venturi, R. Kent Smith, Enrico Sangiorgi, Mark R. Pinto, and Bruno Ricco. A General Purpose Device Simulator Coupling Poisson and Monte Carlo Transport with Applications to Deep Submicron MOSFET's. *IEEE Transactions on Computer-Aided Design*, 8(4), April 1989.

[3] Steven E. Laux and M.V.Fischetti. Monte-Carlo Simulation of Submicrometer Si n-MOSFET's at 77 and 300 K. *IEEE Electron Device Letters*, 9(9), September 1988.

[4] Massimo V. Fischetti and Steven E. Laux. Monte Carlo Analysis of Electron Transport in Small Semiconductor Devices including Band-structure and Space-charge effects. *Physical Review B*, 38(14), November 1988.

[5] A. Al-Omar and J.P. Krusius. Self-Consistent Monte Carlo study of high-field carrier transport in graded heterostructures. *Journal of Applied Physics*, 62(9), November 1987.

[6] Jack M. Higman. *Theoretical Studies of High Field Electron Transport in Silicon Devices.* PhD thesis, University of Illinois at Urbana-Champaign, 1989.

[7] Carlo Jacoboni and Lino Reggiani. The Monte Carlo Method for the Solution of Charge Transport in Semiconductors with Applications to Colvalent Materials. *Reviews of Modern Physics*, 55(3), July 1983.

[8] Doreen Y. Cheng, Chang G. Hwang, and Robert W. Dutton. PISCES-MC: A Multiwindow, Multimethod 2-D Device Simulator. *IEEE Transactions on Computer-Aided Design*, 7(9), September 1988.

[9] C.G. Hwang, R.W. Dutton, J.M. Higman, and K. Hess. Coupled Monte Carlo - Drift Diffusion Analysis of Hot Electron Effects in MOSFET's. *IEEE Trans. Electron Devices*, ED-34(11), November 1987.

[10] C. Canali, C. Jacoboni, F. Nava, G. Ottoviani, and A. Alberigi-Quaranta. Electron Drift Velocity in Silicon. *Physcial Review B*, 12(4), August 1975.

[11] Jr.. A. Phillips and P.J. Price. Monte Carlo calculations on hot electron energy tails. *Applied Physics Letters*, 30(10), May 1977.

[12] J. Tang. *Theoretical Studies of High Field, High Energy Transport in Gallium Arsenide, Silicon, and Heterostructures.* PhD thesis, University of Illinois at Urbana-Champaign, 1983.

[13] Neil W. Ashcroft and N. David Mermin. *Solid State Physics.* W. B. Saunders Company, New York, 1976.

[14] A.C.Smith, J.F.Janak, and R.B.Adler. *Electronic Conduction in Solids.* McGraw-Hill, New York, 1967.

[15] Mildred S. Dresselhaus. *Solid State Phyics Part II: Optical Properties of Solids, 6.732 Class Notes.*

[16] Marvin L. Cohen and T.K. Bergstresser. Band Structures and Pseudopotential Form Factors for Fourteen Semiconductors of the Diamond and Zinc-blende Structures. *Physical Review*, 141(2), January 1966.

[17] R. W. Hockney and J. W. Eastwood. *Computer Simulation using Particles.* Adam Hilger, New York, 1988.

[18] A. Bruce Langdon, Bruce I. Cohen, and Alex Friedman. Direct Implicit Large Time-Step Particle Simulation of Plasmas. *Journal of Computational Physics*, 51:107–138, 1983.

[19] Bruce I. Cohen, A. Bruce Langdon, Dennis W. Hewett, and Richard J. Procassini. Performance and Optimization of Direct Implicit Particle Simulation. *Journal of Computational Physics*, 81:151–168, 1989.

[20] J. Denavit. Time-Filtering Particle Simulations with $\omega_{dt} >> 1$. *Journal of Compuational Physics*, 42:337–366, 1981.

[21] Rodney J. Mason. Implicit Moment Particle Simulation of Plasmas. *Journal of Computational Physics*, 41:233–244, 1981.

[22] C. William Gear. *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[23] V. Rohklin L. Greengard. The Rapid Evaluation of Potential Fields in Particle Systems. *Journal of Computational Physics*, 73, 1987.

[24] V. Rohklin. Rapid Solution of Integral Equations of Classical Potential Theory. *Journal of Computational Physics*, 60:187–207, 1985.

A

Potential Due to a Uniformly Charged Volume

The three-dimensional integral for the potential due to a volume of uniform charge is discussed in Chapter 2. To find the potential at a point (x, y, z) due to a volume of unit charge located at the origin with dimensions (a, b, c) the following integral must be evaluated:

$$\Psi(x,y,z) = \frac{1}{4\pi\epsilon}\int_{-\frac{c}{2}}^{+\frac{c}{2}}\int_{-\frac{b}{2}}^{+\frac{b}{2}}\int_{-\frac{a}{2}}^{+\frac{a}{2}}\frac{dx'dy'dz'}{[(x-x')^2+(y-y')^2+(z-z')^2]^{\frac{1}{2}}} = \frac{1}{4\pi\epsilon}\int_V \frac{1}{r} \qquad \text{(A.1)}$$

A.1 Panel Method

This integral can be approximated by discretizing the volume into a set of panels of size $(\pm\frac{a}{2}, \pm\frac{b}{2}, 0)$. Assuming that the panels are also uniformly distributed with charge such that the total charge in the system is divided equally among the panels, the volume integral now becomes a sum over all of the panels:

$$\Psi(x,y,z) \approx \frac{c}{N}\frac{1}{4\pi\epsilon}\sum_{i=1}^{N}\left[\int_{-\frac{b}{2}}^{+\frac{b}{2}}\int_{-\frac{a}{2}}^{+\frac{a}{2}}\frac{dx'dy'}{[(x-x')^2+(y-y')^2+(z-z_i)^2]^{\frac{1}{2}}}\right] \qquad \text{(A.2)}$$

where N is the number of panels, $\frac{c}{N}$ represents the spacing between the panels in the x-y plane, and z_i describes the location of the center of mass of the i^{th} panel in the z direction. The enclosed two-dimensional integral contained in Equation A.2 is:

$$\Psi'(x,y,z) = \int_{-\frac{b}{2}}^{+\frac{b}{2}}\int_{-\frac{a}{2}}^{+\frac{a}{2}}\frac{dx'dy'}{[(x-x')^2+(y-y')^2+(z-z_i)^2]^{\frac{1}{2}}} \qquad \text{(A.3)}$$

The closed-form solution of this integral for the i^{th} panel and evaluated at the point (x, y, z)

$$\Psi'(x,y,z) \quad = \quad \left(y+\frac{b}{2}\right) ln\frac{(x+\frac{a}{2})+R^1}{(x-\frac{a}{2})+R^3}$$

$$
\begin{aligned}
&+ \quad \left(y-\frac{b}{2}\right) ln \frac{\left(x-\frac{a}{2}\right)+R^4}{\left(x+\frac{a}{2}\right)+R^2} \\
&+ \quad \left(x+\frac{a}{2}\right) ln \frac{\left(y+\frac{b}{2}\right)+R^1}{\left(y-\frac{b}{2}\right)+R^2} \\
&+ \quad \left(x-\frac{a}{2}\right) ln \frac{\left(y-\frac{b}{2}\right)+R^4}{\left(y+\frac{b}{2}\right)+R^3} \\
+(z-z_i) &\left[tan^{-1} \frac{\left(y-\frac{b}{2}\right)\left(x+\frac{a}{2}\right)}{(z-z_i)\,R^2} - tan^{-1} \frac{\left(y+\frac{b}{2}\right)\left(x+\frac{a}{2}\right)}{(z-z_i)\,R^1} \right. \\
&\left. + tan^{-1} \frac{\left(y+\frac{b}{2}\right)\left(x-\frac{a}{2}\right)}{(z-z_i)\,R^3} - tan^{-1} \frac{\left(y-\frac{b}{2}\right)\left(x-\frac{a}{2}\right)}{(z-z_i)\,R^4} \right]
\end{aligned}
\tag{A.4}
$$

where R^1, R^2, R^3 and R^4 are:

$$
\begin{aligned}
R^1 &= \left[\left(x+\frac{a}{2}\right)^2 + \left(y+\frac{b}{2}\right)^2 + (z-z_i)^2 \right]^{\frac{1}{2}} \\
R^2 &= \left[\left(x+\frac{a}{2}\right)^2 + \left(y-\frac{b}{2}\right)^2 + (z-z_i)^2 \right]^{\frac{1}{2}} \\
R^3 &= \left[\left(x-\frac{a}{2}\right)^2 + \left(y+\frac{b}{2}\right)^2 + (z-z_i)^2 \right]^{\frac{1}{2}} \\
R^1 &= \left[\left(x-\frac{a}{2}\right)^2 + \left(y-\frac{b}{2}\right)^2 + (z-z_i)^2 \right]^{\frac{1}{2}}
\end{aligned}
\tag{A.5}
$$

A.2 Gaussian Quadrature Method

The panel method described above presents difficulties due to the singularity present when the point of evaluation is arbitrarily close (but not exactly on) a panel. While a large number of panels is desired for accuracy, this fine discretization establishes a situation where the evaluation points are likely to be extremely close to a panel. Applied to this particular problem, where the electric field is also of interest, this difficulty is exaggerated, since the potential must be computed very accurately in order to provide an accurate electric field calculation.

Instead of discretizing the volume into sheets of charge, quadrature methods can be used to approximate the integral given in Equation A.1. Define the function $f(x', y', z')$ as the integrand $\frac{1}{r}$ given in Equation A.1. Then the three-dimensional integral for a volume of unit

charge density is approximated by the N-point quadrature formula as:

$$\Psi(x,y,z) \approx \frac{1}{4\pi\epsilon}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} aw_i bw_j cw_k f(ax_i, bx_j, cx_k) \tag{A.6}$$

where x_i are the abscissas for the N-point Gauss-Legendre quadrature method defined on the interval $[1,-1]$, the w_i are the corresponding quadrature weights, and a, b, and c are the dimensions of the volume.

This form of the integral is an extension and scaling of the basic N-point Gauss quadrature formula, which gives an approximation for the one-dimensional integral of $w(x)f(x)$ as:

$$\int_a^b w(x)f(x)dx \approx \sum_i^N w_i f(x_i) \tag{A.7}$$

The abscissas, or evaluation points, x_i are the roots of the degree N polynomial from a set of orthogonal polynomials, and the corresponding weights are w_i. The weight $w(x)$ determines the type of Gauss quadrature used and thus the set of orthogonal polynomials from which the abscissas are derived. In general, for any admissible weight function $w(x)$ on [a,b] with corresponding orthonormal polynomials $p_n^*(x)$, if the zeros of $p_n^*(x)$ are $a < x_1 < x_2 < ... < x_n < b$, then positive constants can be found such that

$$\int_a^b w(x)f(x)dx \approx \sum_i^N w_i f(x_i) \tag{A.8}$$

is exact if f is a polynomial of degree less than 2N. The weights are explicitly given by:

$$w_i = -\frac{k_{n+1}}{k_n}\frac{1}{p_{n+1}^*(x_i)p_n^{*\prime}(x_i)}, i = 1...n \tag{A.9}$$

where k_n is the leading coefficient for $p_n^*(x)$, and $p_n^{*\prime}(x)$ is the derivative of $p_n^*(x)$. Assuming that the function f is $2N$ times continuously differentiable, the error from a quadrature approximation will be:

$$E_n(f) = \int_a^b w(x)f(x)dx - \sum_{i=1}^{N} w_i f(x_i) = \frac{f^{2N}(\zeta)}{(2N!)k_n^2} \tag{A.10}$$

for some $\zeta \in [a,b]$.

Gauss-Legendre quadrature uses orthogonal polynomials constructed from $w(x) = 1$ on the interval $[-1,1]$, which are the Legendre polynomials denoted by $P_n(x)$. The weights used specifically for Gauss-Legendre quadrature are given by:

$$w_i = \frac{-2}{P_{n+1}(x_i)P_n'(x_i)}, i = 1...n \tag{A.11}$$

The polynomials $P_n(x)$ can be constructed from the recursion $(n+1)P_{n+1}-(2n+1)P_n+nP_{n-1} = 0$ with $P_0(x) = 1$ and $P_1(x) = x$. From Equation A.10 the N-point Gauss-Legendre method will produce a local error of

$$E_n(f) = \frac{(b-a)^{2N+1}(n!)^4}{(2N+1)\left[(2N)!\right]^3} f^{(2N)}(\zeta) \tag{A.12}$$

over the interval $[a, b]$.

These Quadrature methods of Gauss type are different from standard Newton-Cotes type formulas in that non-equally spaced abscissas are used so that the N-point formula has 2N degrees of freedom, and can thus solve exactly the integral of any polynomials of order $2N-1$ or less.

Since the volume of interest will often contain the singularity of $\frac{1}{r}$, the singularity can be subtracted out and corrected for by integrating the function:

$$\begin{aligned} f^*(x) &= \frac{1}{r}, r > r_{min} \\ f^*(x) &= g(r) = \frac{3}{2r_{min}} - \frac{r^2}{2r_{min}^3}, r < r_{min} \end{aligned} \tag{A.13}$$

The fitting function, $g(r)$ is designed so that f^* and it's derivative will be continuous at $r = r_{min}$. Once the integral of f^* is evaluated, a correction can be made with a closed form expression since

$$\int_0^R \frac{1}{r} dV = 2\pi R^2 \tag{A.14}$$

and

$$\int_0^R g(r) dV = \frac{2}{5}\pi r_{min}^2 \tag{A.15}$$

Thus the integral we are interested in is computed as follows:

$$\int_V \frac{1}{r} dV = \int_V f^* + 2\pi(r_{min})^2 - \frac{2}{5}\pi(r_{min})^2 \tag{A.16}$$

The following tables show the results of taking the volume integral $\int_V \frac{1}{r}$ with both Gaussian quadrature and with Simpson's quadrature rule at three different points of evaluation. With each method, both $\int \frac{1}{r}$ is computed as well as $\frac{d}{dx}\int \frac{1}{r}$. (The * denotes that using more quadrature abscissas will not improve the accuracy of the result.) The volume of unit charge density, V, extends over the interval $[-1, 1]$ in three dimensions. Note that the total charge $Q = 8.0$ so that far from the volume the integral should evaluate to $\frac{8}{r}$. In the tables, N denotes the number of quadrature points.

Gauss for $r = 8$			Simpson for $r = 8$		
N	$\int \frac{1}{r}$	$\frac{d}{dx}\int \frac{1}{r}$	N	$\int \frac{1}{r}$	$\frac{d}{dx}\int \frac{1}{r}$
2	.999906	.124961	3	.1	.125019
3	.999043	.124984*	5	.999947	.124986
			7	.999847*	.124986
			9		.124984*

Gauss for $r = 2$			Simpson for $r = 2$		
N	$\int \frac{1}{r}$	$\frac{d}{dx}\int \frac{1}{r}$	N	$\int \frac{1}{r}$	$\frac{d}{dx}\int \frac{1}{r}$
2	3.91809	1.81345	3	4.01392	2.06983
3	3.95262	1.88952	5	3.9515	1.88811
4	3.95012	1.888	7	3.95071	1.88992
5	3.9504	1.88967	9	3.95047	1.88956
6	3.95037*	1.88941	11	3.95041	1.8895
7		1.88945*	13	3.95039	1.88947
			15	3.95038*	1.88945*

Gauss for $r = 0.5$		Simpson for $r = 0.5$	
N	$\int \frac{1}{r}$	N	$\int \frac{1}{r}$
2	7.83	3	10.1797
3	9.14998	5	8.90141
4	8.64986	7	8.9737
5	8.95558	9	8.99268
6	9.0086	11	8.98097
10	8.98217	13	8.97927
15	8.98079*	15	8.98071

A.3 Electric Field Calculation

Once we have a way of evaluating the potential, the electric field at any point in space can be computed using finite difference approximations. The electric fields are computed numerically

from:

$$\begin{aligned}\vec{E}(\vec{r}) &= -\nabla\Psi(\vec{r}) \\ &\approx -\left[\frac{\Psi(x+\delta x,y,z)-\Psi(\vec{r})}{\delta x},\frac{\Psi(x,y+\delta y,z)-\Psi(\vec{r})}{\delta y},\frac{\Psi(x,y,z+\delta z)-\Psi(\vec{r})}{\delta z}\right]\end{aligned} \tag{A.17}$$

where δx, δy, and δz are numbers which are small compared with the dimensions of the problem being simulated. Solving for the electric field contributions in three dimensions requires computing the potential at four evaluation points, representing perturbations in each dimension plus the original evaluation point.

B

Jacobians for Implicit Time Integration Schemes

Chapter 4 discusses implicit time-integration schemes for solving the motion equations for a particle. The position update for a particle is computed using a relaxation scheme, so for each particle, it is assumed that all of the other charge in the system is fixed. The actual position update is dictated by the discretization scheme. Since the update vector is three-dimensional, a three-dimensional Newton's method is employed.

In general, an N-dimensional Newton's method is used to solve N non-linear equations $\vec{F}(\vec{u}_i)$ for a vector $\vec{u} \in \Re^N$ such that:

$$|\vec{F}(\vec{u}_i)| < \epsilon \approx 0 \tag{B.1}$$

The method is iterative, and for Newton iteration number i, $\vec{u}_i$ is updated by a vector $\vec{\delta}$ from:

$$\mathbf{J}\vec{\delta} = -\vec{F} \tag{B.2}$$

such that the solution will become:

$$\vec{u}_{i+1} = \vec{u}_i + \vec{\delta} \tag{B.3}$$

If the solution vector $\vec{u}_{i+1}$ satisfies Equation B.1, then the simulation is complete; otherwise, this updating procedure is repeated until a convergent solution is found.

In Equation B.2, **J** is the Jacobian matrix containing partial derivatives of the functions described by $\vec{F}$ with respect to all of the parameters. In detail,

$$\mathbf{J} = \begin{bmatrix} \frac{\partial F_1}{\partial u_1} & \frac{\partial F_1}{\partial u_2} & \cdots & \frac{\partial F_1}{\partial u_N} \\ \frac{\partial F_2}{\partial u_1} & & & \\ \vdots & & \ddots & \\ \frac{\partial F_N}{\partial u_1} & & & \frac{\partial F_N}{\partial u_N} \end{bmatrix} \tag{B.4}$$

The non-linear problems to be solved at every timestep are repeated here. For the backward Euler scheme,

$$\vec{F} = \vec{r}(t+h) - \vec{r}(t) - h\left[\vec{v}(t) - \frac{hq}{m}\vec{E}(t+h)\right] = 0 \tag{B.5}$$

and for the trapezoidal scheme,

$$\vec{F} = \vec{r}(t+h) - \vec{r}(t) - \frac{h}{2}\left[2\vec{v}(t) - \frac{hq}{2m}\left(\vec{E}(t) + \vec{E}(t+h)\right)\right] = 0 \tag{B.6}$$

The associated Jacobians for Equation 4.14 are:

$$\frac{\partial F_i}{\partial r_i}(t+h) = 1 + \frac{h^2 q}{m}\frac{\partial E_i(t+h)}{\partial r_i} \tag{B.7}$$

for the diagonal terms, and:

$$\frac{\partial F_i}{\partial r_j}(t+h) = \frac{h^2 q}{m}\frac{\partial E_i(t+h)}{\partial r_j} \tag{B.8}$$

for the off-diagonal ($i \neq j$) terms. For Equation 4.15, the diagonal terms are:

$$\frac{\partial F_i}{\partial r_i}(t+h) = 1 + \frac{h^2 q}{4m}\frac{\partial E_i(t+h)}{\partial r_i} \tag{B.9}$$

and the off-diagonal terms are:

$$\frac{\partial F_i}{\partial r_j}(t+h) = \frac{h^2 q}{4m}\frac{\partial E_i(t+h)}{\partial r_j} \tag{B.10}$$

The electric field E_i is the electric field component in the i^{th} dimension for a particle located at $\vec{r}$ due to all of the other charges in the system.

Since the system described in Chapter 4 is represented by three types of charges, the various derivatives given in the equations above must be computed for all of the charges. The contributions from each of the charges are then superimposed to give the net interaction with the charge. For the mobile charges and the contact charges, represented by point charges located various positions, $\vec{r}^{*}$, the derivatives can be computed analytically. The electric field

contributions from all of the point charges in the system, given by Equation 3.12, are added to give:

$$E_i = \sum_k \frac{q_k}{4\pi\epsilon} \frac{r_i - r_i^k}{|\vec{r} - \vec{r}^k|^3} \tag{B.11}$$

In both cases the essential derivatives of the electric field are:

$$\frac{\partial E_i}{\partial r_i} = \sum_k \frac{q_k}{4\pi\epsilon} \left[\frac{1}{|\vec{r} - \vec{r}^k|^3} - \frac{3}{|\vec{r} - \vec{r}^k|^5} (r_i - r_i^k)|r_i - r_i^k| \right] \tag{B.12}$$

$$\frac{\partial E_i}{\partial r_j} = \sum_k \frac{q_k}{4\pi\epsilon} \left[-\frac{3}{|\vec{r} - \vec{r}^k|^5} (r_j - r_j^k)|r_j - r_j^k| \right] \tag{B.13}$$

Since the interactions between the mobile charges and the ionized impurity charge is computed numerically, the electric field derivatives from this calculation which are required for the Jacobian terms must also be computed numerically. Contributions to the Jacobians from this piece of the electric field are calculated from the potential, and added to the contributions from the mobile charge part of the electric field. This then yields the total Jacobian terms. Note that computing these numerical derivatives with finite-difference techniques requires computing the potential at 7 locations.