ELEC 4700 Modelling of Integrated Devices

Assignment 1: Monte-Carlo Modelling of Electron Transport

Due: February 2, 2020

Professor: Tom Smy

Student Name: Rudi Hidvary

Student Number: 101037815

Introduction

Modelling the behavior of electrons in a semiconductor is a way to gain insight into the way they would behave in real life under certain conditions. These models follow user defined rules that should reflect what occurs but as it is a model, always fail to perfectly achieve this goal. However, using a powerful computing program like MATLAB, we can attempt to do this using equations and algorithms that combined, will give a reasonable approximation to the behavior of molecules so that we can see how they operate under certain conditions. Scattering within a semiconductor is common, Monte-Carlo simulation techniques will be used to demonstrate this within the material to see its effects.

In our case, a N-type silicon crystal is being modelled for electron transport. In this material the effective mass is 26% that of an electron in free space and there is an initial temperature which is uniform across the crystal lattice that gives the electrons an initial velocity. that is found using energy transfer between kinetic and thermal energy. The area that the simulation is conducted in is a section of the N-type material that is 200 nm long and 100 nm tall. For simulation purposes, when an electron hits the top or bottom of the simulation area, it will have a perfectly elastic collision where the electron is sent back at the same angle with a negative velocity. If an electron hits either side of the simulation area, it will be sent to the opposite side as if the material loops around and connects back with itself.

Electron Modelling

Before any simulation is achieved the initial positions and velocities of the particles must be found. Since the crystal lattice has a temperature of 300 degrees Kelvin, all of the electrons will begin the simulation with the same velocity. All of the particles will start with the same velocity which will be the average velocity calculated using the energy relationship between kinetic and thermal energies. This simulation has 2 degrees of freedom where energy can be stored, the x direction and the y direction. The simulation is 2-D and there is no rotation of the atoms so there will be information stored in any other form of energy. Using this fact and the known temperature and mass, the initial kinetic energy of each particle can be calculated using the following Equation 1.

$$E = \frac{mv^2}{2} = \frac{d}{2}kT \qquad \text{(Eq. 1)}$$

$$v = \sqrt{\frac{2 * 2kT}{2m}} = \sqrt{\frac{2\left(1.381e-23\,\frac{J}{K}\right)(300\ K)}{(0.26 * 9.11e-31\ kg)}} = 1.8704 * 10^5\ m/s$$

This is the velocity that each of the particles will be initialized with as the temperature is held at a constant 300 degrees kelvin. Each particle is assigned a random angle, and then basic trigonometry is used to send the initial particle off in a random direction all with the same speed.

If the mean time between collisions for any particle is 0.2 picoseconds, then the mean free path can be determined experimentally once the simulation is run. The average velocity of all the particles can be multiplied by the average time between collisions to find the average distance between collisions. MATLAB calculated the MFP roughly as 0.007 m.

The trajectories of the particles can be seen in Figure 1 where each electron is traced as the code iterates through a loop of timesteps of 1 femtosecond long. For this example, the simulation was run with a small number of particles over a long period to see the effect of the boundary conditions, 10 particles over 1 picosecond as well as a second example of 100 particles.

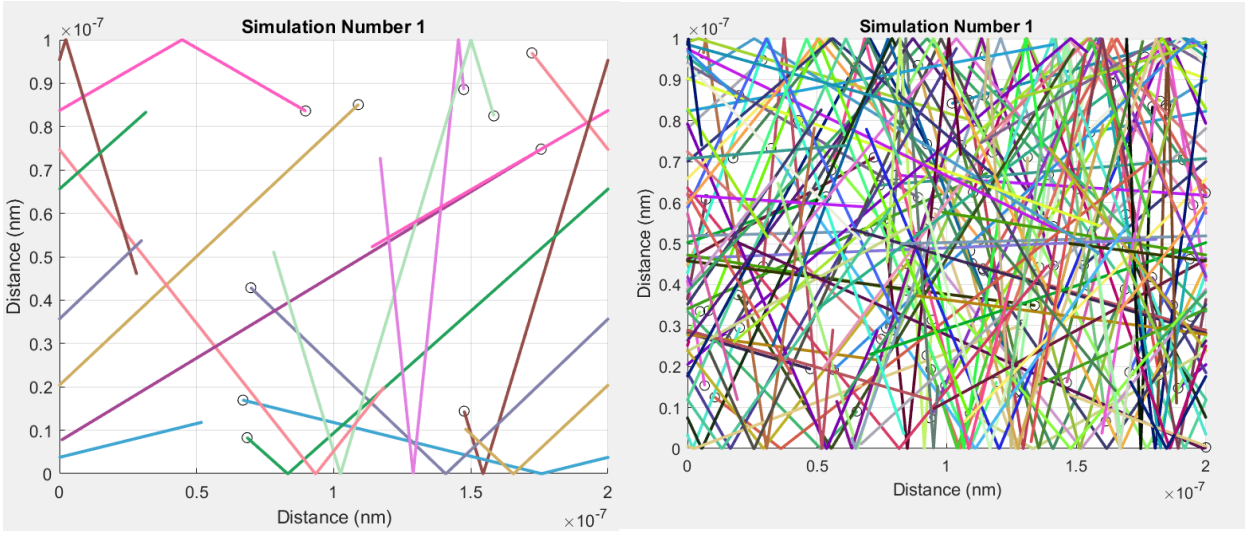

Figure 1: Particle Trajectories for Simulation 1

The temperature of the system over time can be calculated using this average velocity as well. This was done by rearranging Equation 1 from above to find the temperature given the velocity. This plot can be seen in Figure 2 and was a simulation using 100 electrons over 100 femtosecond period.

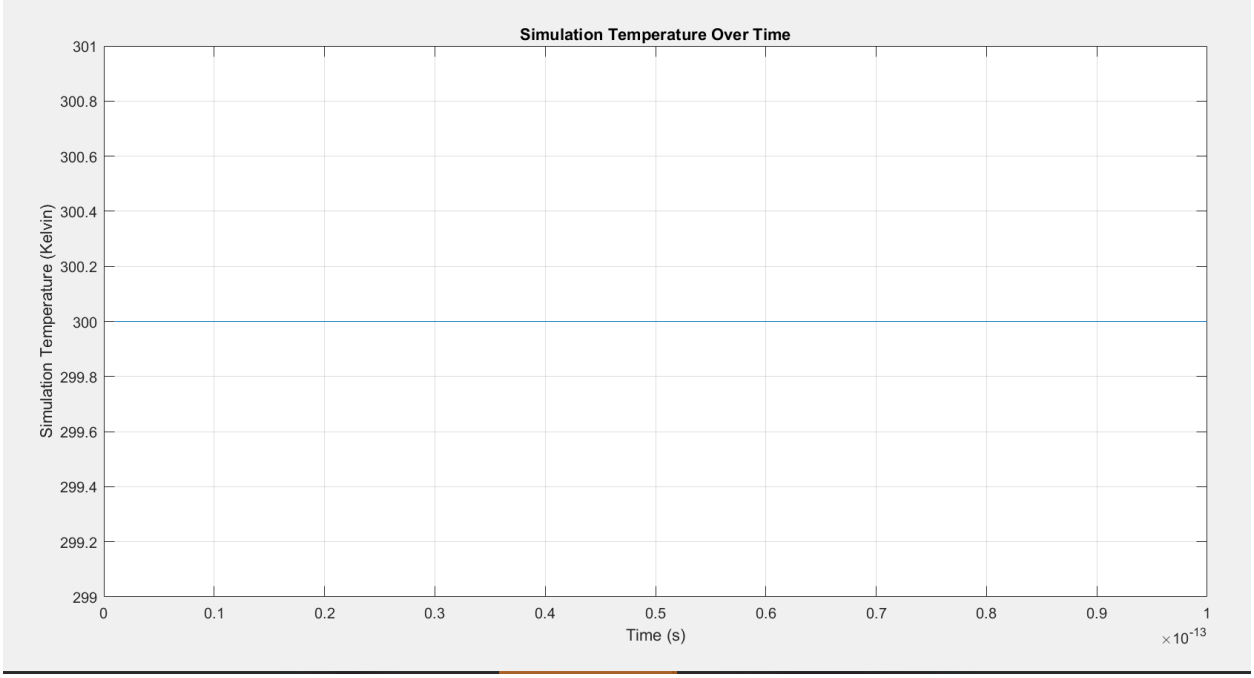

Figure 2: Temperature Plot for Simulation 1

The temperature response of the system is a constant temperature of 300. This is due to the relationship between temperature and velocity. Since all the particles have the same initial thermal velocity and do not change throughout the simulation, the temperature at every step is the same.

Collision with Mean Free Path

To create a more sophisticated model, a distribution of energies is used for the particles so that the velocity of the particle is randomly chosen from it. The distribution that was used in this simulation was a Maxwell-Boltzmann distribution which is made using the square root of the sum of two squared normal distributions. This is then multiplied by a factor so that it is normalized around the thermal velocity that was calculate in Equation 1. To show how that randomly generated distribution follows the Maxwell-Boltzmann distribution, a histogram plot is used to bin the velocities and display them. An example of one of these auto-generated plots can be seen in Figure 3 where 10 million particles are binned into 250 boxes.

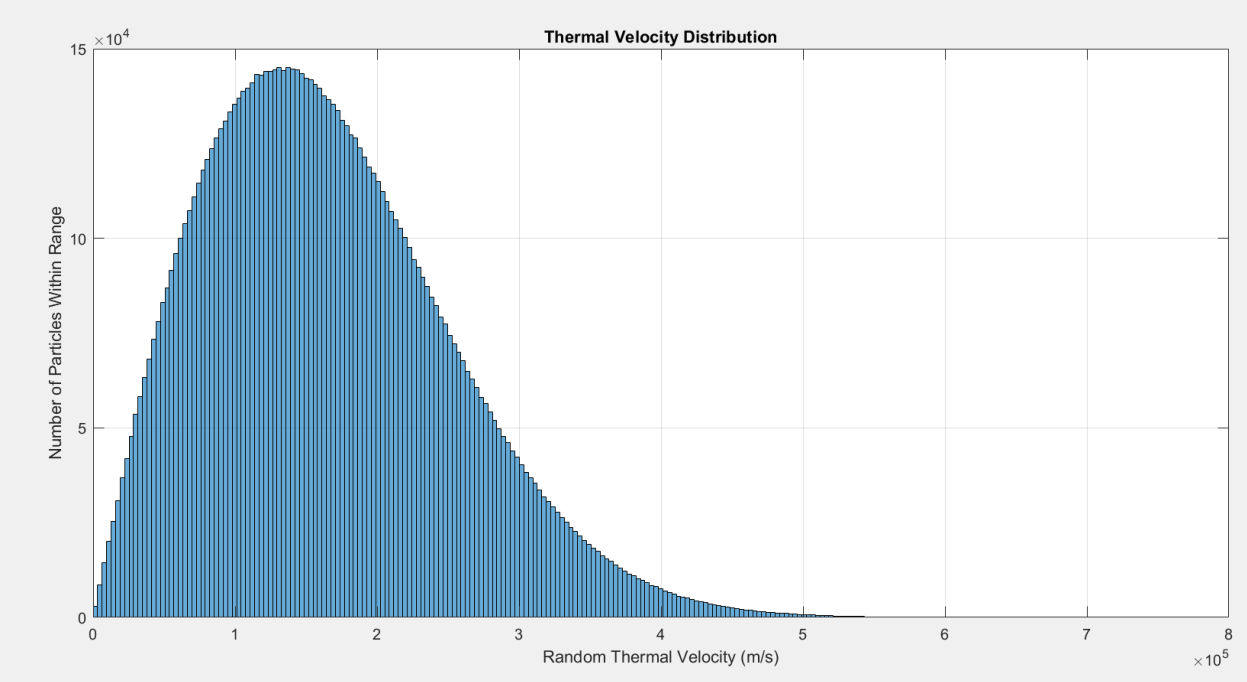

Figure 3: Maxwell-Boltzmann Distribution Random Particle Example

Initializing the particle position is the same as the first simulation, but to get the velocity on each of the particles, each must be assigned one of the random values form the distribution. The random angle is chosen the same way but now using trigonometry and the random values.

Another way to increase the usefulness of the model is to incorporate electron scattering into the simulation. This can be achieved by comparing a randomly generated number to the chance that an electron will scatter over a given time step. Using Equation 2, the scatter probability distribution is calculated using the given mean time of 0.2 picoseconds and the time step which is 1 femtosecond.

$$Pscatter = 1 - e^{-\frac{Timestep}{Tmn}} \quad \text{(Eq. 2)}$$

$$Pscatter = 1 - e^{-\frac{1E-15}{0.2E-12}} = 0.0048975$$

This gives roughly a 0.5 % chance for each particle to scatter at every given timestep. Using this in the loop, at even given time step, all particles have a chance of scattering and then rethermalizing and having a new random velocity assigned to it.

The trajectories of the particles changes when a particle hits a boundary and now it also changes when the electron scatters. To demonstrate the effect this has in the simulation an example of the particle trajectories can be seen in Figure 4 with 10 particles over 1 picosecond as well as 100 particles over the same time period.

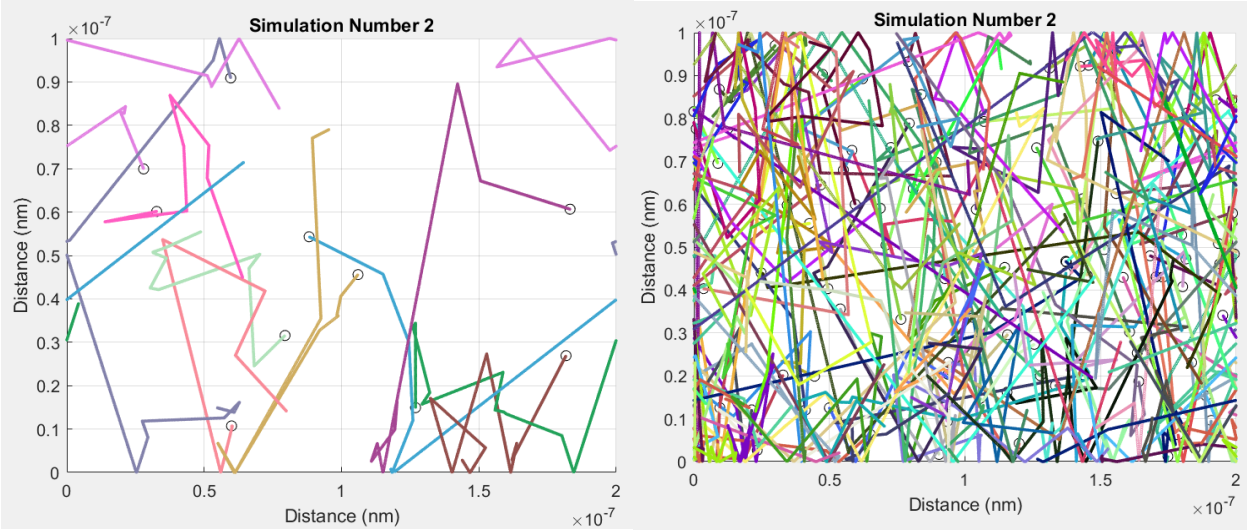

Figure 4: Particle Trajectories for Simulation 2

Now the temperature can be calculated again at each time step to check the average velocity of the particles and use Equation 1 to find the corresponding temperature. In the first simulation, the temperature was set in the beginning and then remained constant throughout the simulation. In this simulation, the initial temperature is set by the distribution assigned to the particles. This will theoretically be 300 degrees kelvin, but if a small number of particles are used, then the initial temperature may be higher or lower than 300. As time progresses, the temperature will change over time due to the rethermalizing particles that have just scattered. This will make the temperature fluctuate, in theory it should fluctuate around 300 Kelvin at the mean of the distribution is set to 300 Kelvin but over short periods of time, more erratic things can take place. Figure 4 shows an example of the temperature over time for a period of 1 picosecond, using 10 000 electrons in the simulation.

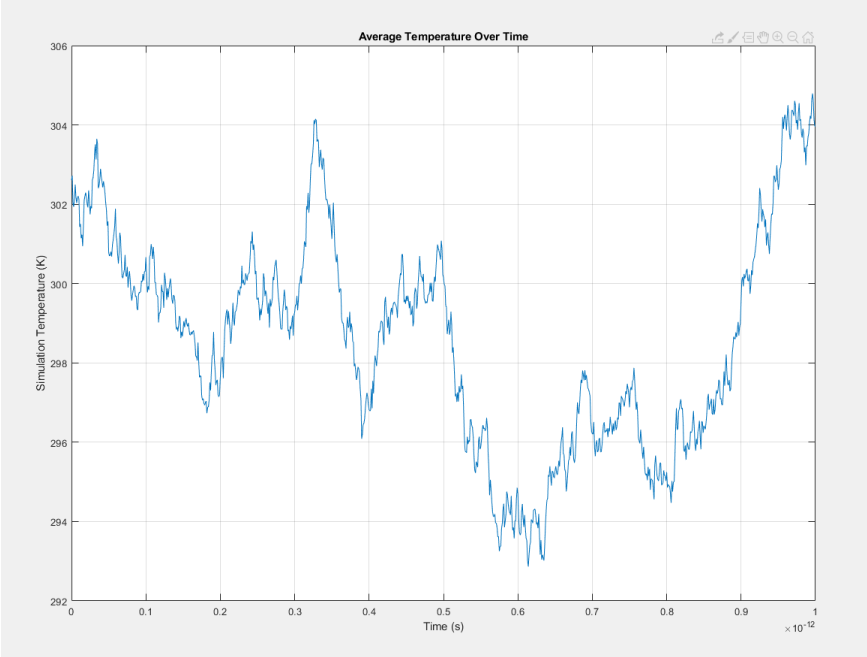

Figure 4: Temperature Plot for Simulation 2

Using the loops in the code, the number of collisions is counted, this can be used to calculate the mean free path and average time between collisions. To do this the collision number is updated in the scattering loop, then this is used in the Equation 3 to find the mean time between collisions.

$$Time\ Between\ Collisions = (Total\ Simulation\ Time * \#\ Particles)/(\#\ Collisions) \quad \text{(Eq. 3)}$$

This number changes for each run of the simulation as the number of collisions is dependant on a random probability that is re generated in each simulation. Now that the average time between collisions is calculated, the mean free path can be found using the average velocity that is also calculated during the simulation. To check to ensure that the simulation is running correctly, a percent error is calculated comparing the correct values for the mean time and mean free path and the calculated ones. An example of a calculated MFP and mean time between collisions can be seen in Figure 5, where the percent error is calculated from the values found in the first section of this report. In this case the simulation was run with 10000 particles over a 1 picosecond timespan to get reasonable accurate results given the random nature of the simulation.

```
QUESTION 2
The average time between collisions was found to be 2.002964e-13 (s)
Comparing this to the given value of 2.000000e-13 (s)
The percent error of the average time is 0.148219 percent
The mean free path was calculated to be 6.939990e-03 (m)
Comparing this to the previously calculated value of 6.996538e-03 (m)
The percent error of the MFP is 0.808232 percent
```

Figure 5: Mean Free Path and Time Between Collision

Enhancements to the Simulation

The simulation is now much more advanced than before, but more improvements can be made. The first is to have restricted regions in the simulation space that act like barriers to the electrons to simulate a small tunneling region. This can be done by initializing a length of the bottleneck region and a height or that region. These two parameters give a bottleneck region that can be varied and made to be any size if both parameters do not exceed the standard simulation size. For the purpose of this report, all of the plots have the same bottleneck size, a length of 80 nm and a height of 20 nm. There are a few problems with this addition to the simulation, the first problem is that no particles can be initialized into these regions, the second problem is boundary conditions of the bottleneck and how to handle the electrons that approach the new restricted regions. To solve the first problem, a loop is used to check if there are particles in the restricted region. Then it reinitializes the particles that remain in the restricted region and loops until there are no particles in the restricted zone. This method can become very calculation intensive if the restricted region is large compared to the regions where electrons can flow, but it also has the advantage of keeping the distribution in the region as random as possible.

The second problem becomes more complicated as the boundaries can be either specular or diffusive when particles interact with them. Specular boundaries are those that the particle has a perfect collision with and the velocity only changes direction with no loss of speed. Diffusive boundaries are those that re-thermalize the particle with a new velocity off the Maxwell-Boltzmann distribution and send them off in a new random direction. The boundary choice is determined at the beginning of the simulation code where a value of 1 will make the boundaries diffusive and a value of 0 will make the boundaries specular. The specular boundaries work much like the top and bottom boundary, where if a particle crosses to the restricted region then the velocity is reversed in whatever dimension that was crossed. For example, if a particle hops the boundary in the x direction, then the velocity in the x direction is reversed and the velocity in the y direction remains the same. For the diffusive boundary, a random angle and velocity Is obtained from the distribution and then assigned to the particle. To ensure that the particle doesn't go further into the restricted region after a boundary collision, the absolute value is taken, and the same method used for the specular boundary

applies. For another example, the particle comes from the left side of the boundary and hits, a new angle and velocity is applied to that particle, since it came from the left the x velocity has to be negative for the particle to escape so the negative of the absolute velocity is taken so that it travels in a direction away from the boundary. To show how each of these boundaries work, the trajectories of a small number of particles was recorded with Figure 6 demonstrating specular boundaries and Figure 7 demonstrating diffusive boundary conditions.

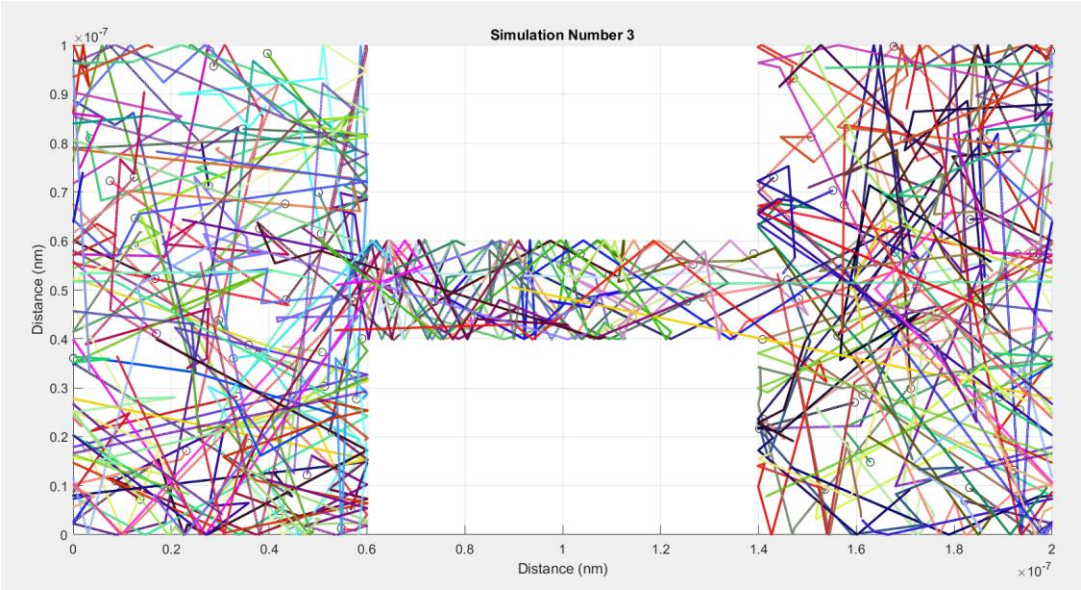

Figure 6: Specular Boundary Conditions

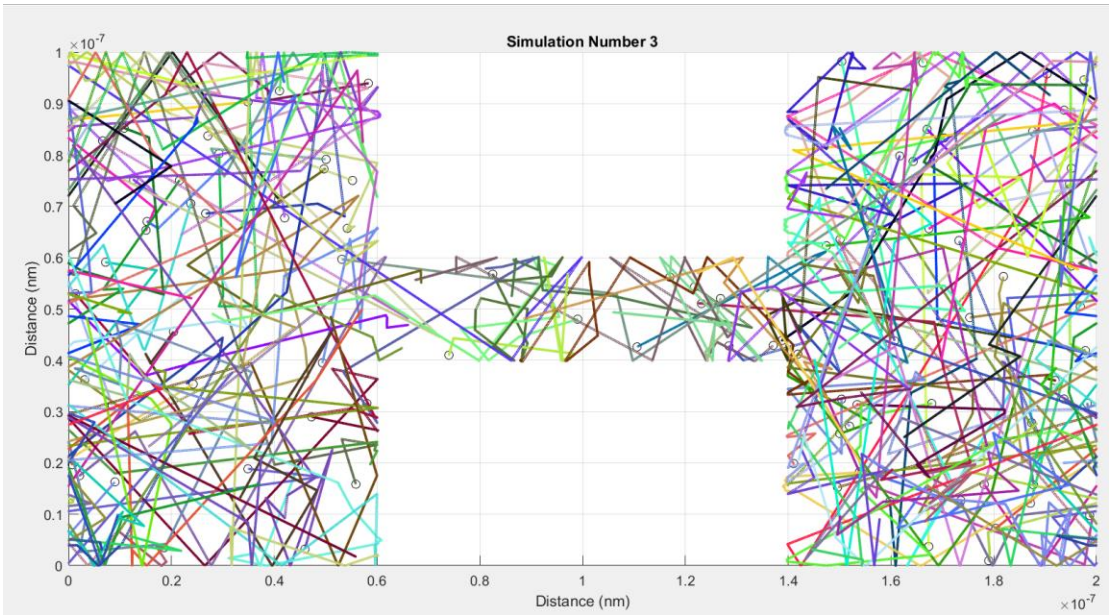

Figure 7: Diffusive Boundary Conditions

To visualize large numbers of particles, the trajectory plot is not very useful. A three-dimensional plot of the final positions of the electrons is useful as it shows where the electrons eventually ended up and the density of electrons in an area. This can be achieved using a three-dimensional histogram plot that bins the electrons in an area and represents the number of electrons per bin. An example of this plot in the simulation can be seen in Figure 8 where 100000 particles are initialized, and the simulation ran for a short period of 10 femtoseconds.

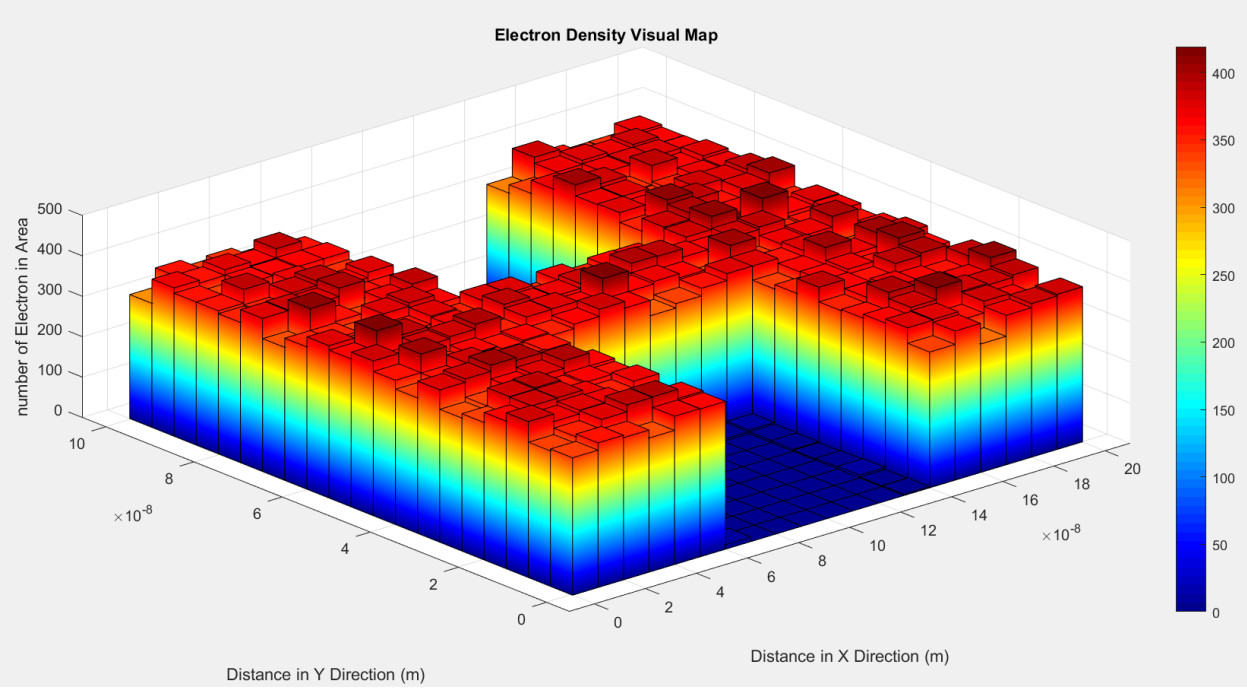

Figure 8: Electron Density Map

A two-dimensional plot can be used to display the region in the x-y plane where the electrons are more populated. Figure 9 shows the birds eye view of the plot where the colour indicates different amounts of electrons.

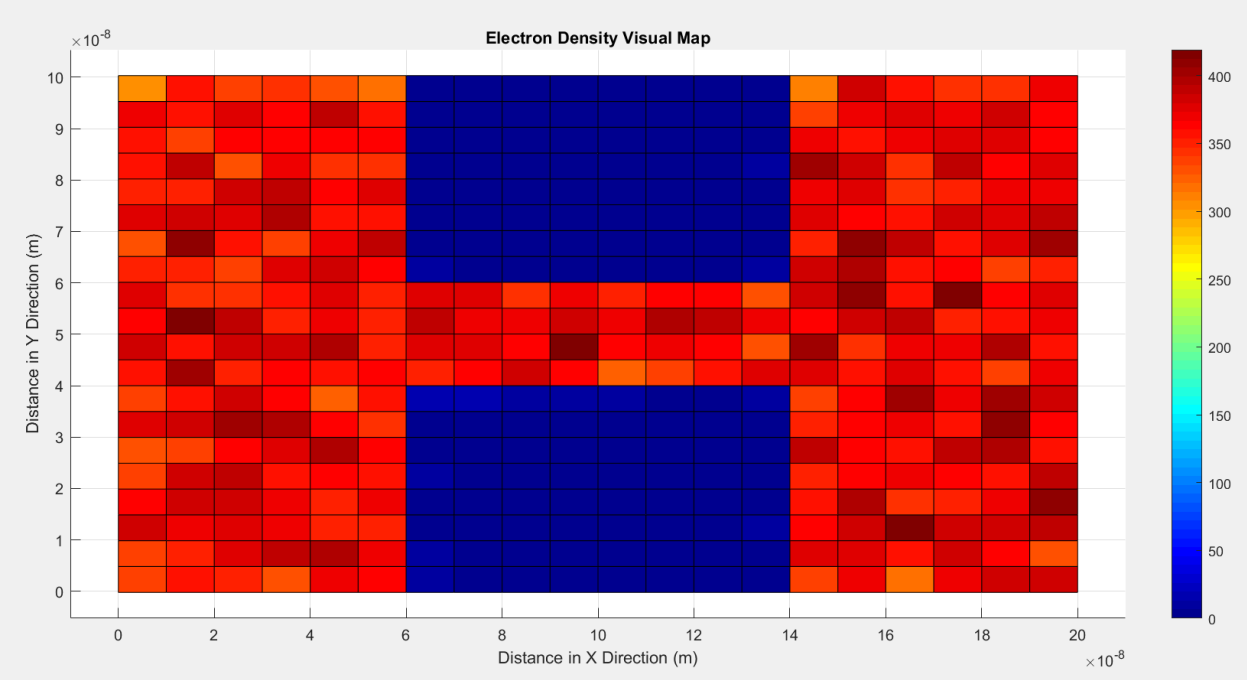

Figure 9: 2D Electron Density Map

Changing the number of particles and bins will affect the way in which the plot is distributed. For example, Figure 10 shows the results from a simulation where the bin size remains the same, but the electron number is changed from 100 000 to 1000 particles. Figure 11 shows the corresponding 2D plot.

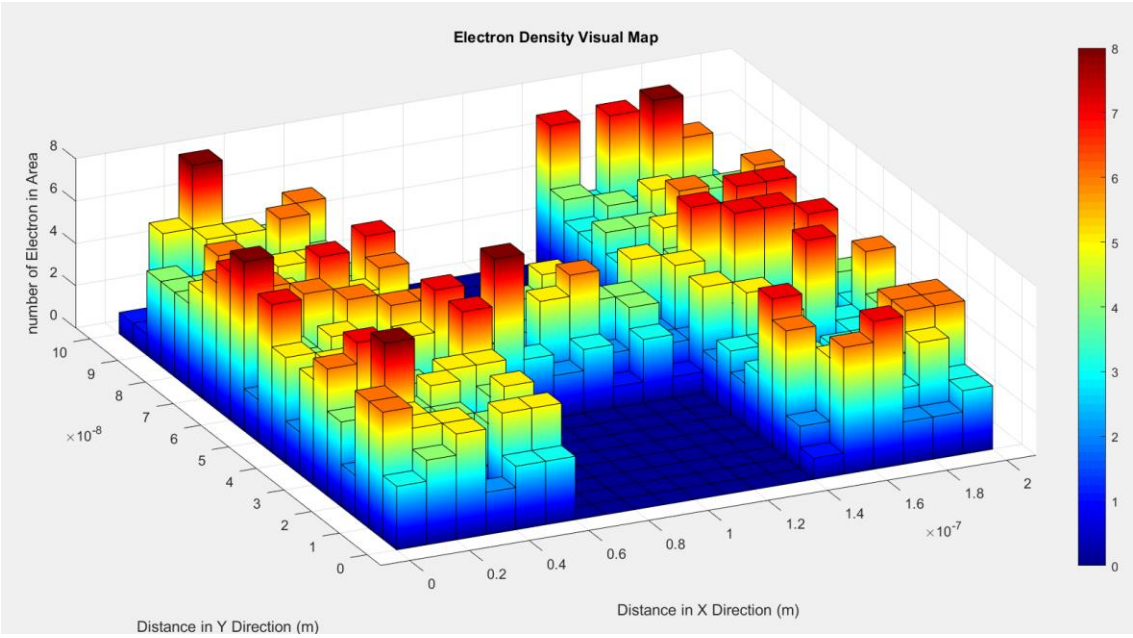

Figure 10: Sparse Electron Density Map

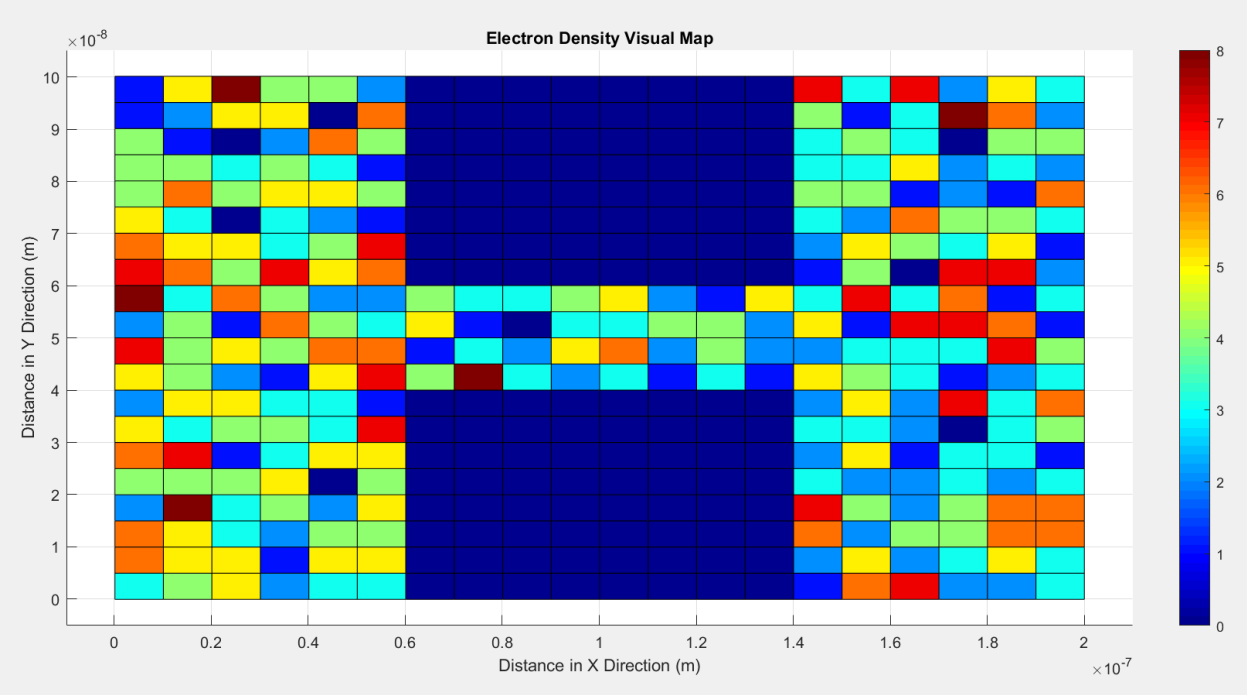

Figure 11: Sparse 2D Electron Density Map

This result shows that different histogram configurations and number of electrons display the random effect of the simulation. When there are many electrons per bin, then the distribution appears quite equal, but when the electrons are small compared to the bin number, then the random variation of the particle velocities is more dominant. In the simulation there is an option to view the electron density map as a movie. This is a computationally expensive plot to generate so it is by default set to only plot the final positions electron density map, however for small number of particles and iterations it can be viewed to see ow the distribution of electrons changes over time. The reason this is computationally expensive for large amounts of particles or time steps is because the simulation is keeping track of all the particle positions as time progresses and stores it in a matrix that has the dimension electron number x time steps. Increasing either the time step or electron number will multiply the amount of calculations that must be done.

One last improvement to the model will be to generate a temperature map of the simulation which will be determined by the velocity of each particle and how they are distributed over the surface. As with the electron density plot, this one can either be viewed as a movie or just the final temperature distribution, with the movie option being very computationally expensive. Figure 12 shows a sample of the final temperature plot for a simulation with 10 000 particles over a 10-femtosecond period.

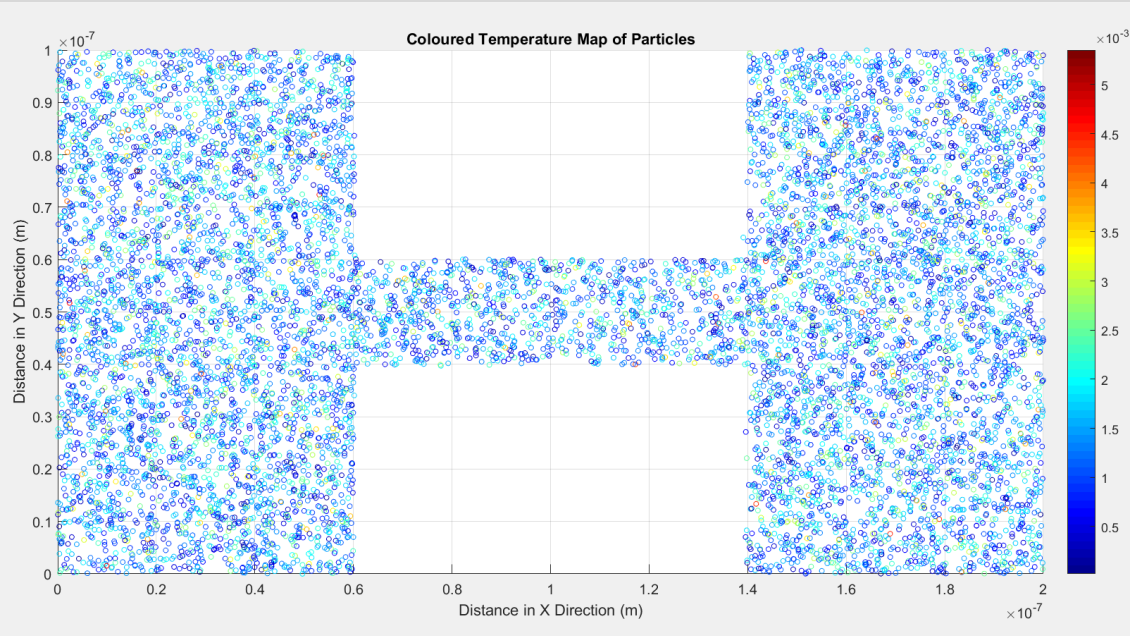

Figure 12: Final Temperature Map of Simulation

Conclusion

The model was able to simulate electrons in a conductor to a reasonable extent and regarding how they act in real life. This led to a better understanding of how electrons possibly move inside of a material. Using the data from the interaction between the random elements in the simulation emergent behavior can be observed such as the calculated time between collisions that simulates a real example where there are fluctuations and not everything is always at the same temperature across a distribution. The effect of using a Monte Carlo simulation is that the imperfectness of reality can be estimated using reasonable distributions interaction with each other based on laws of physics. Macroscopic units like temperature are usually mean values, but the individual particles have their own unique temperature based on their velocity, so creating random distributions of particles allows simulations over short time spans with specific values for each particle. Using MATLAB, these simulations were carried out to show how Monte Carlo simulations can be used.