

ABSTRACT

AGARWAL, ADITI. Simulation of a Thermomagnetic Cooling Device. (Under the direction of Daryoosh Vashae.)

Richard P. Feynman, during one of his famous lectures at Cornell University in 1964 said that-

Nature uses only the longest threads to weave her patterns, so that each small piece of her fabric reveals the organization of the entire tapestry.

Clearly, the deeper meaning behind these poetic words of one of the greatest minds in history is that the study of nanoscale phenomena leads to a greater understanding of how the nature works. Semiconductors play a huge role in human lives today. With miniaturization of devices, new problems emerge such as high electric field, heat management and dissipation, quantum effects et cetera. For microelectronic circuits, heat dissipation is a huge concern. Solid State cooling provides a practical solution to this problem by enabling localized cooling for hot spots inside a semiconductor device (like ICs, lasers, sensors etc.). These microcoolers can be monolithically integrated with the system on chip, greatly reducing the cost of integration while simultaneously improving device performance and efficiency.

The focus of this work is to study the Ettingshausen effect and to use this phenomena to build a solid state refrigerator. In this device, we attain cooling just by applying a magnetic field perpendicular to the direction of current! It has been shown that such a device would be most suitable for cryogenic applications.

Simulation is a powerful tool that makes an engineer's life easier when many correlated factors are at play. Efforts were made to model thermomagnetic refrigerator and show the cooling effect in presence of electric and magnetic fields to better understand carrier transport and optimize the device. We used *Drift-diffusion* model in Sentaurus TCAD and *Monte Carlo* method in MATLAB for modelling. We found that either of these tools are not sufficient by themselves for this problem. The drift-diffusion model lacks the ability to account for carrier temperatures and the Monte Carlo method is unsuitable for large-scale devices. For future work, a coupled approach using these two methods is suggested.

Chapter 1 of this thesis provides an introduction and review of solid state cooling. Explanation of the models used and the results for the simulation, problems encountered using both of these approaches are elucidated in chapters 2 and 3.

© Copyright 2017 by Aditi Agarwal

All Rights Reserved

Simulation of a Thermomagnetic Cooling Device

by
Aditi Agarwal

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Electrical Engineering

Raleigh, North Carolina

2017

APPROVED BY:

B. Baliga

Robert Kolbas

Mehmet Ozturk

Daryoosh Vashaee
Chair of Advisory Committee

DEDICATION

To my mom and dad, Anita and Ajay Agarwal.

BIOGRAPHY

Aditi Agarwal was born in New Delhi, India on April 29, 1992. She got her undergraduate degree in Electrical Engineering at Delhi Technological University, New Delhi.

Aditi is now pursuing graduate career at North Carolina State University. During her MS, she worked on synthesis of thermoelectric materials and theoretical modelling of thermomagnetic refrigerator with Prof. Daryoosh Vashaee. Her current research focusses on fabrication of Silicon Carbide power devices with Prof. Jayant Baliga as her PhD advisor.

Beyond academics, Aditi has a diploma in Indian classical music, plays the guitar and likes to read.

ACKNOWLEDGEMENTS

I would first like to thank my thesis advisor Prof. Daryoosh Vashee of the Department of Electrical Engineering at North Carolina State University for providing me constant guidance when I ran into a difficult problem or question about my research or writing. I would also like to thank him for his valuable inputs while working on this thesis.

I would also like to thank my colleagues and friends- Amin Nozariasbmarz, Abhishek Malhotra and Michael Hall for teaching me how to use the lab equipment. Working with them was a huge learning experience for me and the exchange of ideas on a daily basis truly broadened my horizons.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Aditi Agarwal

TABLE OF CONTENTS

LIST OF TABLES	vii
LIST OF FIGURES	viii
Chapter 1 INTRODUCTION	1
1.1 Nanoscale Solid State Cooling	1
1.1.1 Thermoelectric Coolers (TECs)	2
1.1.2 Thermionic Coolers	3
1.1.3 Thermomagnetic Coolers	4
1.2 Thermogalvanomagnetic Effects	5
Chapter 2 THERMOMAGNETIC COOLER	7
2.1 Effect of magnetic field on a semiconductor	7
2.2 Carrier and heat transport in presence of magnetic field	8
2.2.1 Unipolar or single-carrier transport	8
2.2.2 Bipolar or two-carrier transport	9
2.3 Derivations	10
2.3.1 Heat flow	10
2.3.2 Coefficient of Performance	11
2.3.3 Ettingshausen figure-of-merit	12
2.4 Device Model	12
2.4.1 Synopsys Sentaurus TCAD SDE and SDevice model	12
2.4.2 Results and Discussion	15
Chapter 3 MONTE CARLO SIMULATIONS	17
3.1 Monte Carlo Method	17
3.2 Fundamental Concepts	19
3.2.1 Transport Equation	19
3.2.2 Energy Bands	19
3.3 Explanation of the Monte Carlo Code	20
3.3.1 Parameter File	20
3.3.2 sc_table	21
3.3.3 init	23
3.3.4 free_flight_scatter	25
3.3.5 drift	25
3.3.6 scatter_carrier	29
3.3.7 Poisson solver	32
3.4 Results and problems encountered while modelling a thermomagnetic device using Monte Carlo approach	34
3.4.1 Convergence and noise	34
3.4.2 MC-Poisson coupling	37
3.4.3 Device scaling for thermomagnetic application	37

Chapter 4	CONCLUSION	38
BIBLIOGRAPHY		40
APPENDIX		42
Appendix A	Simulation files and code	43
A.1	Sentaurus TCAD Command File	43
A.1.1	SDE Command File	43
A.1.2	SDevice Command File	45
A.2	Monte Carlo MATLAB code	48
A.2.1	Code	49

LIST OF TABLES

Table A.1	List of variables	48
-----------	-------------------------	----

LIST OF FIGURES

Figure 1.1	TEC module consisting of n-type and p-type legs connected electrically in series and thermally in parallel. Picture taken from [Zia16]	2
Figure 1.2	Principle of operation of a vacuum thermionic cooler. Picture taken from [Lee12]	3
Figure 1.3	Principle of operation of a solid state thermionic cooler. Picture taken from [Zia16]	4
Figure 2.1	Ettingshausen effect in unipolar or single-carrier system.	9
Figure 2.2	Ettingshausen effect in bipolar or two-carrier system.	9
Figure 2.3	Device structure simulated using Sentaurus TCAD.	13
Figure 2.4	When a magnetic field of 1 T is applied in the y-direction, electrons move to one side, leaving behind a bound positive charge which creates a Hall voltage. This colormap is for 1.2 mA applied current.	16
Figure 2.5	Spatial variation of electrostatic potential.	16
Figure 2.6	Electron energy colormap as calculated from spherical harmonic expansion of distribution function (at 1.2mA applied current).	16
Figure 3.1	Flowchart depicting the steps in ensemble Monte Carlo simulation	18
Figure 3.2	Classification of scattering mechanisms. Picture taken from [Vas11]	22
Figure 3.3	Graphical representation of the direct technique for choosing a random energy value from a Maxwellian distribution using a random number generator with uniform probability distribution.	24
Figure 3.4	Initial distributions for momentum vector k_x , k_y and k_z	25
Figure 3.5	Initial distributions for position vector r_x , r_y and r_z	26
Figure 3.6	Flowchart for free_flight_scatter() function	27
Figure 3.7	Pictorial depiction of how a random scattering event is chosen out of a discrete set of mechanisms. Picture taken from [Vas11].	30
Figure 3.8	Electrical conductivity versus temperature curve for Sb_2Te_3	35
Figure 3.9	Seebeck coefficient versus temperature curve for Sb_2Te_3	35
Figure 3.10	Nernst coefficient versus temperature curve for Sb_2Te_3	36
Figure 3.11	Ettingshausen coefficient versus temperature curve for Sb_2Te_3	36

CHAPTER

1

INTRODUCTION

One of the major issues that limits the performance and miniaturization of an Integrated Circuit is heat dissipation. This makes solid state cooling a crucial field of interest these days. Efficient small-scale coolers are needed to cool ICs, sensors etc., to reduce noise and improve device performance. Other devices that need temperature stabilization are semiconductor lasers and optoelectronic devices. There are different types of coolers that operate in different temperature regimes. Some of them are mentioned in this chapter. This work is focused on modeling thermo-magnetic cooling, which is mainly suited for low temperature applications.

1.1 Nanoscale Solid State Cooling

[Zia16] provides a comprehensive review of recent developments in solid state cooling. This section presents a summary of different physical phenomena that can be utilized for refrigeration at nanoscale level. Broadly, these phenomena lie under three categories that are: thermoelectric, thermionic and thermo-magnetic and are described briefly as follows:

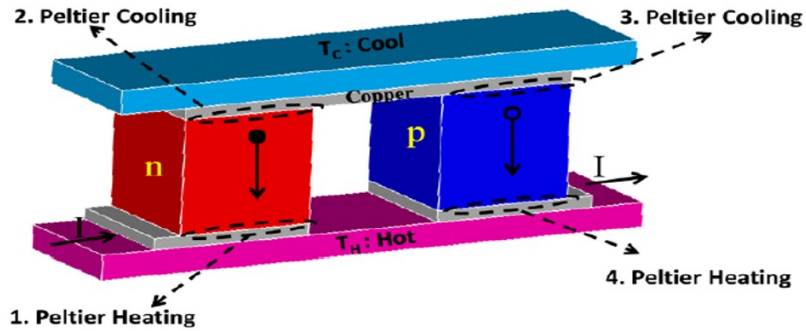


Figure 1.1 TEC module consisting of n-type and p-type legs connected electrically in series and thermally in parallel. Picture taken from [Zia16]

1.1.1 Thermoelectric Coolers (TECs)

A TEC module comprises of an n-type and p-type leg connected electrically in series and thermally in parallel between the hot side and the cold side. For obtaining cooling power from the device, an electric current is passed through the n-type and p-type legs. As the carriers move from one side to the other, heat transfer takes place at the junction. This phenomenon is called the *Peltier effect* according to which heat is absorbed or produced at the junction of two dissimilar materials when electric current is passed across the junction as illustrated in figure 1.1. The Peltier heat, Q is proportional to the electric current, I i.e. $Q = \pi I$, where π is the Peltier coefficient. Conversely, if a temperature gradient exists across such a device, a voltage is obtained across the junctions. This is called the *Seebeck effect* and the device operates in the generating mode. To obtain a high efficiency, the n-type and the p-type materials need to have a high figure-of-merit, ZT which is defined as:

$$ZT = \frac{S^2 \sigma}{\kappa} T \quad (1.1)$$

Here, S is the Seebeck coefficient or the thermoelectric power, σ is the electrical conductivity and κ is the total thermal conductivity, including the electronic thermal conductivity (κ_e) and lattice thermal conductivity (κ_l).

Some of the best materials which exhibit a high ZT in the low temperature regime are Bismuth Telluride and some nanostructured materials, as reviewed by [PB10].

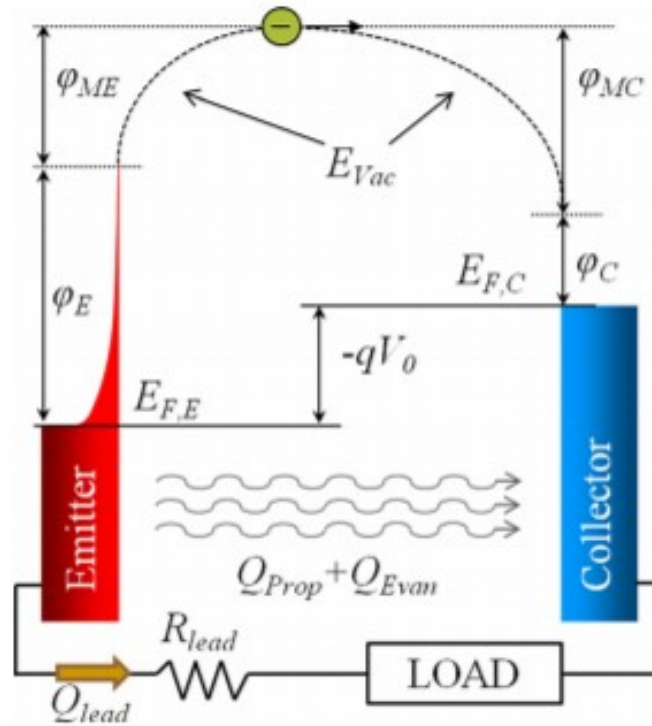


Figure 1.2 Principle of operation of a vacuum thermionic cooler. Picture taken from [Lee12]

1.1.2 Thermionic Coolers

1.1.2.1 Vacuum Thermionic Devices

A thermionic device converts heat into electrical energy by utilizing thermionic emission as illustrated in figure 1.2. Electrons are emitted by a low work function cathode in contact with a heat source. The electrons flow towards a cold lower work function anode against a potential barrier. The cathode and anode are separated by a vacuum gap, typically a few microns in size. Cooling occurs at cathode as high energy electrons are separated by the potential barrier. As mentioned in [Lee12], main challenges faces by vacuum thermionic coolers are:

1. They operate at very high temperatures as electrons have to gain sufficient thermal energy to overcome the potential barrier.
2. Lack of availability of thermally stable low work function materials.

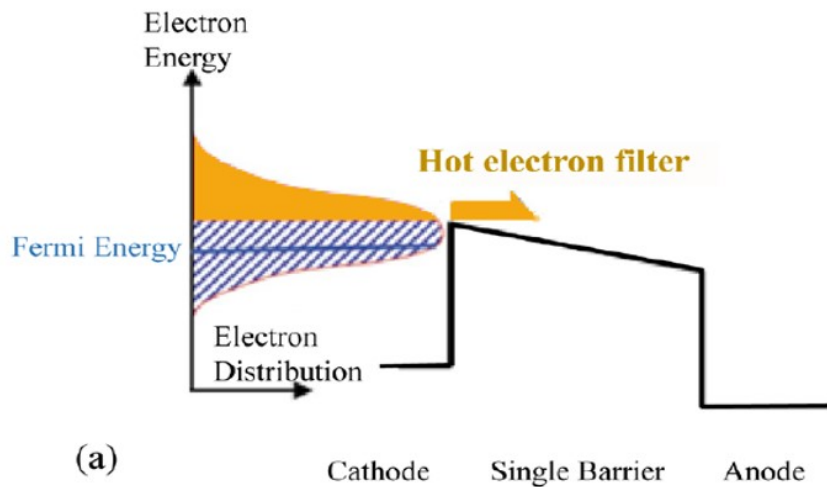


Figure 1.3 Principle of operation of a solid state thermionic cooler. Picture taken from [Zia16]

3. Space charge effects and radiation losses limit the efficiency of a vacuum thermionic cooler.

However, one of the advantages is that hot and cold sides are thermally insulated i.e. no heat transfer can occur between the cathode and anode because of the presence of a vacuum gap.

1.1.2.2 Solid state thermionic devices

Solid state thermionic refrigerators use the principle of thermionic emission but replace the vacuum gap with a semiconductor layer, effectively lowering the barrier potential and the operating temperature range. A single-barrier or a multi-barrier semiconductor heterostructure like InGaAs/-GaAs/InGaAs can be used to separate high energy electrons and bring about evaporative cooling of the electron gas. The principle of operation of a single barrier heterostructure type thermionic cooler is shown in figure 1.3.

1.1.3 Thermomagnetic Coolers

Two types of effects cause cooling when an external magnetic field is present. These are:

1. **Magneto- Peltier or magneto- Seebeck effect**

Generally, the presence of a magnetic field reduces the mobility of the carriers, thereby reducing the electrical and thermal conductivities of a thermoelectric device. However, it has been shown in [WS62] that some Bi-Sb alloys show Seebeck coefficient enhancement in the presence

of a transverse magnetic field. However, the realization of such a device is not possible due to lack of a corresponding p-type material and secondly because the Seebeck enhancement is not large enough to justify using magnets.

2. Nernst-Ettingshausen effect

Nernst-Ettingshausen effect comes into play in presence of transverse electric and magnetic fields inside a bulk material. The following section elaborates different phenomena arising out of interactions between electric field, magnetic field and heat flow.

1.2 Thermogalvanomagnetic Effects

A new means of solid state cooling in 77-200K temperature range is based upon an effect first observed by the German physicist, Ettingshausen in 1887. It states that when a magnetic field exists perpendicular to an electrical current flowing in a conductor, it results in a transverse temperature gradient which is perpendicular to both the current flow and the magnetic field. The Ettingshausen coefficient is defined as:

$$P_{xy} = \frac{\nabla_z T}{B_y J_x} \quad (1.2)$$

Where $\nabla_z T$ is the gradient of temperature in z-direction, B_y is the magnetic field in y-direction and J_x is the current density.

Conversely, if a heat current is made to flow in a conductor perpendicular to the magnetic field, an electric field is observed perpendicular to both the heat current and the magnetic field. This is the *Nernst effect* and the Nernst coefficient is defined by:

$$N_{xy} = -\frac{E_x}{B_y \nabla_z T} \quad (1.3)$$

Where E_x is the electric field.

The *Hall effect*, first reported in 1879 states that in addition to the transverse temperature gradient due to Ettingshausen Effect, a transverse electric field is also observed which is perpendicular to both electric current and the magnetic field. The Hall coefficient is defined as:

$$R_{xy} = \frac{E_x}{B_y J_x} \quad (1.4)$$

In addition, the *Righi-Leduc effect* observed in 1888 states that when a heat current is made to flow in a conductor perpendicular to the magnetic field, in addition to the electric field produced

by Nernst effect, a transverse temperature gradient also exists perpendicular to both heat flow and magnetic field. The Righi-Leduc coefficient is given as:

$$L_{xy} = \frac{\nabla_z T}{B_y \nabla_x T} \quad (1.5)$$

To illustrate the principle of thermomagnetic cooling, let us assume that a stream of free electrons having an initial velocity v_x moving along the x-direction is placed inside a magnetic field B_y in the y-direction. The electrons undergo a deflection in the -z direction, which creates a charge imbalance and results in an electric field E_z . This field, which is the Hall field builds up until it balances the Lorentz force, the particles are no longer deflected and a steady state exists. In a real solid, a velocity distribution exists instead of a constant velocity and thus an average velocity replaces v_x . All the electrons with the velocity greater than this average velocity will be deflected towards negative z direction and those with velocity lesser than the average velocity will be deflected towards positive z direction. This is further explained with illustrations in the next chapter.

Some of the advantages of Ettingshausen coolers over conventional thermoelectric coolers are:

1. Unlike the TEC, in an Ettingshausen cooler, only one material needs to be optimized instead of two material systems (for the n-type and p-type legs).
2. Required cooling power can be attained because the heat flow is normal to the direction of current flow. This makes it possible to have a higher cross-sectional area in the direction of heat flow and lower cross-sectional area in the direction of current flow.
3. It is possible to build an infinite staged cascade Ettingshausen cooler by exponential shaping of the device as shown in [Har63].

Because of the above mentioned advantages of thermomagnetic coolers over conventional thermoelectric ones, especially at lower temperatures, we need to study this phenomena in greater depth in order to fully understand and utilize it for practical devices.

CHAPTER

2

THERMOMAGNETIC COOLER

In Chapter 1, a review of solid state cooling was presented. This chapter presents a detailed study of a specific type of cooling device, namely the Ettingshausen cooler. Section 2.1 describes the effect of magnetic field on a semiconductor. In section 2.2, carrier and heat transport in presence of magnetic field is elucidated. In section 2.3, the relevant mathematical derivations are presented and section 2.4 provides a description of the device model used to perform the simulations in Sentaurus TCAD and the results for the same.

2.1 Effect of magnetic field on a semiconductor

When a magnetic field is applied transverse to the direction of flow of electrical current, Hall effect produces an e.m.f. in the direction perpendicular to both electrical current and magnetic field. The material changes its electrical resistance in the presence of a magnetic field and this phenomenon is called *Magneto-resistance*. These two effects can be used to quantify doping concentration and mobility of charge carriers in a sample.

Magnetic field also affects the electronic part of thermal conductivity, a phenomenon called *magneto-thermal resistance*. Essentially, the product $\kappa\rho$ increases because of magnetic field and thus in most materials, the thermoelectric figure-of-merit decreases. However, some bismuth-antimony

alloys exhibit Seebeck coefficient enhancement in presence of magnetic field [WS62].

Apart from Hall effect, other transverse thermogalvanomagnetic effects are the Nernst effect (transverse electric field produced by a longitudinal heat flow), the Ettingshausen effect (transverse temperature gradient produced by a longitudinal electric current) and the Righi-Leduc effect (transverse temperature gradient produced by longitudinal heat flow). Among these, Nernst and Ettingshausen effects are the most useful for practical applications. For instance, Nernst effect can be used to make an efficient thermal radiation detector and Ettingshausen effect can be utilized in low-temperature solid state cooling applications.

A major challenge faced during realization of a thermomagnetic refrigerator is thermal shorting due to electrical contacts. However, due to separation of directions of electrical and thermal currents, the cooling power of this device can be enhanced by changing device geometry. This presents a significant advantage over thermoelectric coolers. Also, unlike thermoelectric coolers where both p-type and n-type materials are needed, only one type of material system suffices for realization of a refrigerator based on Nernst-Ettingshausen effect. Finding a material with high figure-of-merit is again a big challenge for this field.

2.2 Carrier and heat transport in presence of magnetic field

The mechanisms of heat transfer are different, depending on whether the material system is extrinsic or intrinsic. Carrier and heat transport in both types of semiconductor are explained in this section.

2.2.1 Unipolar or single-carrier transport

This type of transport occurs in extrinsic semiconductors, either p-type or n-type where only one species of carrier exists. As shown in the figure 2.1, when a current flows in the x-direction in the presence of a magnetic field in the y-direction, the electrons experience Lorentz force, which deflects them towards one side, creating a net negative charge on one side and a net positive charge on the other side. This results in a Hall field in the z-direction, which exerts a force on the carriers in the direction opposite to that of the Lorentz force.

The Lorentz force is velocity dependent (and hence energy dependent) while the Hall force is not. Thus, Lorentz force dominates if the carrier velocity is higher than a certain equilibrium value and for low energy carriers, the Hall force dominates. Since these two forces act in opposite directions, the high energy charge carriers go to one side and the low energy carriers go to the other side. This mechanism causes energy separation in a direction perpendicular to both electric current and magnetic field, which leads to the cooling effect on one side of the sample.

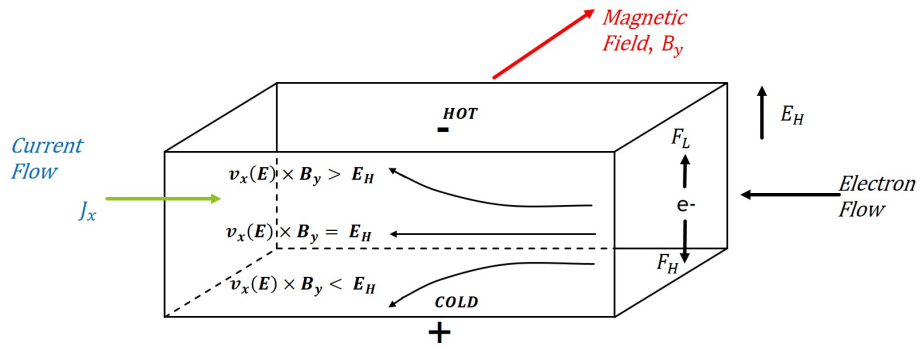


Figure 2.1 Ettingshausen effect in unipolar or single-carrier system.

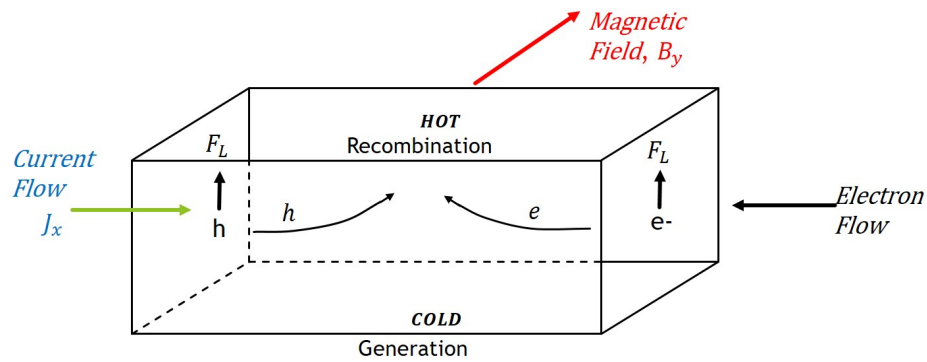


Figure 2.2 Ettingshausen effect in bipolar or two-carrier system.

2.2.2 Bipolar or two-carrier transport

Ettingshausen effect increases considerably in an intrinsic semiconductor or a semimetal, in which the number of electrons and holes are comparable, provided the band gap is comparable to the thermal energy kT as explained in [Del62].

In this case, there is no Hall field since both electrons and holes are deflected towards the same side. However, accumulation of electrons and holes on one side causes increased recombination on one side (heat generation). This is compensated by electron-hole pair generation on the other side (heat absorption). This mechanism is depicted pictorially in figure 2.2. Since heat transfer takes place by recombination and generation processes, a more efficient cooling is obtained in comparison with an extrinsic semiconductor.

2.3 Derivations

This section presents derivations for heat flow, COP and Ettinghausen figure-of-merit.

2.3.1 Heat flow

There are majorly three heat flow mechanisms which affect the cooling power of this device. First, the Nernst-Ettinghausen effect is responsible for creating a thermal current flow from cold side to the hot side of the sample. However, thermal conduction from hot side to cold side as well as Joule heating in the sample are responsible for reducing the net heat flow from cold side to the hot side.

Thus, assuming an electrical current I_x in the x-direction, magnetic field B_y in the y-direction and heat flow q_z in the z-direction, we get the following expression for the net thermal current as explained in [Gol95]-

$$q_z = \frac{NB_y I_x T_1 L_x}{L_z} - \frac{\kappa L_x L_y (T_2 - T_1)}{L_z} - \frac{I_x^2 \rho L_x}{2L_y L_z} \quad (2.1)$$

where L_x , L_y and L_z are the sample dimensions in the x, y and z directions respectively, κ is the thermal conductivity, T_1 and T_2 are the temperatures of the cold and the hot side and N is the Nernst coefficient.

In the equation(2.1), the first term represents the heat produced due to Nernst-Ettingshausen effect, the second term corresponds to the heat conduction from hot side to cold side and the third term is Joule heat that flows from hot side to cold side.

It is obvious that the heat flow is dependent on the geometry of the device i.e. on the factor L_x/L_z . Thus, q_z can be increased by increasing the dimension of the device in the direction of current flow, relative to the dimension of the device in which thermal current flows.

Further, the Nernst coefficient and the Ettingshausen coefficient are related by the Bridgman relation as follows:

$$P\kappa = NT \quad (2.2)$$

where P is the Ettingshausen coefficient and T is the temperature.

In steady state or static condition, no heat flows i.e. $q_z = 0$ and the temperature drop is given by:

$$\Delta T_E = PI_x B_y / L_y \quad (2.3)$$

Under dynamic conditions (i.e. thermal current flowing), $q_z \neq 0$ and the temperature drop is

given by:

$$\Delta T_E = \Delta T + L_z q_z / \kappa \quad (2.4)$$

Further, the thermal current interacts with the magnetic field B_y to produce the Nernst-Ettingshausen potential gradient, which the electrical current must overcome. The Nernst- Ettingshausen potential gradient is given by:

$$\Delta V_{NE} = -N q_z B_y L_x / \kappa \quad (2.5)$$

Thus the potential difference required to drive the current, ΔV is:

$$\Delta V = \Delta V_{NE} + \rho I_x L_x / L_y L_z \quad (2.6)$$

The total power delivered by the voltage source to the system is, $P_t = I_x \Delta V$. Power delivered to the thermal current is:

$$P_{thermal} = I_x \Delta V - \rho I_x^2 L_x / L_y L_z$$

According to Carnot principle, the heat Q_c absorbed at cooler side is given by:

$$Q_c = [I_x \Delta V - \rho I_x^2 L_x / L_y L_z] \frac{T_c}{\Delta T}$$

There will also be Joule heat inside the sample, half of which flows to the cooler side and the other half flows to the hotter side. Thus the net heat absorbed at the cold side is:

$$Q'_c = Q_c - \rho I_x^2 L_x / 2 L_y L_z$$

2.3.2 Coefficient of Performance

The coefficient of performance, COP is defined as:

$$\text{COP} = \frac{\text{Heat pumped at cooling surface}}{\text{Power Input}} = \frac{Q'_c}{P_t}$$

$$\text{COP} = \frac{T_c}{\Delta T} \frac{I \Delta V - \rho I_x^2 L_x / L_y L_z - \rho I_x^2 L_x \Delta T / L_y L_z T_c}{I_x \Delta V} \quad (2.7)$$

From equations (2.3,2.4, 2.5, 2.6), the relation between ΔV , I_x and ΔT can be found and substituted in equation (2.7).

If we define a dimensionless number, X as $X = NP B_y^2 / \rho$, equation (2.7) can be written as:

$$\text{COP} = \frac{T_c}{\Delta T} \frac{X - L_y N B_y I \Delta T / \rho I_x - \Delta T / 2 T_c}{X - L_y N B_y I \Delta T / \rho I_x + 1} \quad (2.8)$$

For a detailed derivation of the COP, see [ES62].

2.3.3 Ettingshausen figure-of-merit

We can define the thermomagnetic power analogous to thermoelectric power (Seebeck coefficient) as:

$$\alpha_{xz} = B_y N_{xz} = -E_x / \nabla_z T \quad (2.9)$$

The thermomagnetic figure-of-merit can similarly be defined as:

$$Z_{xz} = \alpha_{xz}^2 / \rho_{xx} \kappa_{zz} \quad (2.10)$$

where ρ_{xx} and κ_{zz} are the electrical resistivities and thermal conductivities in the respective directions. Maximum ΔT is obtained when $Q'_c = 0$ and is given by:

$$\Delta T_{max} = \frac{1}{2} Z_{xz} T_h^2 \quad (2.11)$$

Maximum COP that can be obtained from the device is given by:

$$\text{COP}_{max} = \frac{T_c}{\Delta T} \frac{1 - \delta T_h / T_c}{1 + \delta}$$

where $\delta = (1 - Z_{xz} \bar{T})^{1/2}$ and $\bar{T} = (T_h + T_c) / 2$. The derivations for these quantities are explained in entirety in [Haw64].

2.4 Device Model

2.4.1 Synopsys Sentaurus TCAD SDE and SDevice model

The command files for the SDE and SDevice modules can be found in the appendix. The steps for defining a structure through SDE are as follows:

1. A cuboidal 3D structure made of silicon is defined with variable n-type doping, which can be set from the workbench.

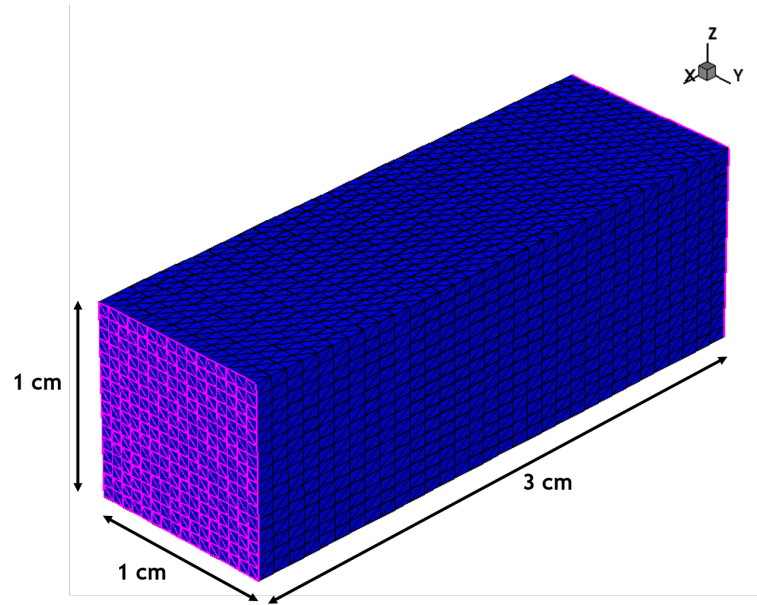


Figure 2.3 Device structure simulated using Sentaurus TCAD.

2. Positive and negative contacts are defined at appropriate faces.
3. Mesh refinements are specified for the entire region.
4. Additionally, thermal contacts can also be defined if heat flow simulations are desired.

The resulting structure is shown in figure 2.3.

The salient features of the Sdevice simulation are as follows:

1. Drift-diffusion model

The drift-diffusion model is the default model in Sdevice. The current densities for electrons and holes are solved using following equations:

$$J_n = -nq\mu_n \nabla \phi_n \quad (2.12)$$

$$J_p = -pq\mu_p \nabla \phi_p \quad (2.13)$$

where ϕ_n , ϕ_p are electron and hole quasi fermi potentials and μ_n and μ_p are electron and hole mobilities.

Additional models available in Sdevice are the thermodynamic and hydrodynamic. The ther-

hydrodynamic model includes ΔT as a driving force in current continuity equations:

$$J_n = -nq\mu_n(\nabla\phi_n + P_n\Delta T) \quad (2.14)$$

$$J_p = -pq\mu_p(\nabla\phi_p + P_p\Delta T) \quad (2.15)$$

where P_n and P_p are the thermoelectric powers.

The hydrodynamic model takes into account the spatial variations of electrostatic potential, electron affinity, band gap, effective masses, carrier concentration gradient, carrier temperature gradient etc. However, this model can not be coupled with magnetic field.

2. Magnetic field

By activating the magnetic field model in the Physics section of Sdevice command file, we can include magnetic field- dependent terms in the drift-diffusion model. The galvanic transport model however can not be combined with hydrodynamic, impact ionization, aniso, piezo and quantum models in Sentaurus TCAD.

3. eSHEDistribution

To visualize electron energies, we include the eSHEDistribution keyword in the physics section. It uses the spherical harmonic expansion (SHE) method to compute energy distribution function, $f(\mathbf{r}, \epsilon)$ by solving lowest-order SHE of the Boltzmann transport equation (BTE). This enables SDevice to solve for different macroscopic variables that can be obtained from the energy distribution such as:

$$n_{SHE} = 2g_v \int_0^{\infty} g(\epsilon)f(\epsilon)d\epsilon \quad (2.16)$$

$$T_{n,SHE} = \frac{2g_v}{n_{SHE}} \int_0^{\infty} \frac{2\epsilon}{3k} g(\epsilon)f(\epsilon)d\epsilon \quad (2.17)$$

where n_{SHE} and $T_{n,SHE}$ are the electron density and average electron energy respectively. The keywords eSHEDensity and ESHEEnergy need to be specified in the plot section in order to solve for them during the simulation.

4. Mobility

Carrier mobility degradation due to doping as well as high field are accounted for by including the high field saturation and doping dependence mobility models.

5. Generation- Recombination

Doping dependent and temperature dependent Shockley-Read-Hall recombination as well as Auger recombination models are included.

2.4.2 Results and Discussion

Some of the limitations for modelling this device in Sentaurus TCAD are:

1. Inability to enable coupling between electron/hole temperatures with carrier continuity equations in presence of magnetic field. The hydrodynamic model in Sdevice which solves for electron and hole temperatures does not work when magnetic field is enabled. A work around to this problem was found by tracking the eSHEEnergy variable which calculates electron energy from the distribution.
2. In order to solve for lattice temperature, a thermal boundary condition needs to be specified i.e. we can not assume a floating-temperature boundary. Electron and hole temperatures don't require boundary conditions but they are computed using the hydrodynamic model, which can not be combined with magnetic field.
3. The value of magnetic field is a critical parameter for obtaining convergence. If doping-dependent mobility model is used, convergence is reliable up to $B= 10T$. For general mobility, the limit is up to $B= 1T$. Thus, convergence issues limit the applied magnetic field value. For our device, we used a magnetic field of 1T and also included the doping-dependent mobility model to circumvent any convergence issues.

Apart from the above mentioned limitations, following observations were made as per our simulation results:

1. When a magnetic field is applied in the y-direction and a current flows in the x-direction, charge separation in z-direction is clearly visible as shown in the colormap in figure 2.4. This also results in a Hall voltage in the z-direction. The spatial variation of electrostatic potential at different values of applied current is shown in figure 2.5 and it was observed that ΔV increases as applied current is increased.
2. When the variable eSHEEnergy is visualized in Sentaurus Visual, we can clearly see that electron energy at one end of the device is higher than the other end (see figure 2.6).

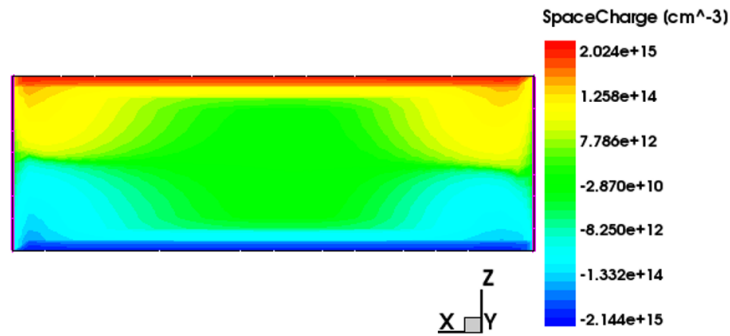


Figure 2.4 When a magnetic field of 1 T is applied in the y-direction, electrons move to one side, leaving behind a bound positive charge which creates a Hall voltage. This colormap is for 1.2 mA applied current.

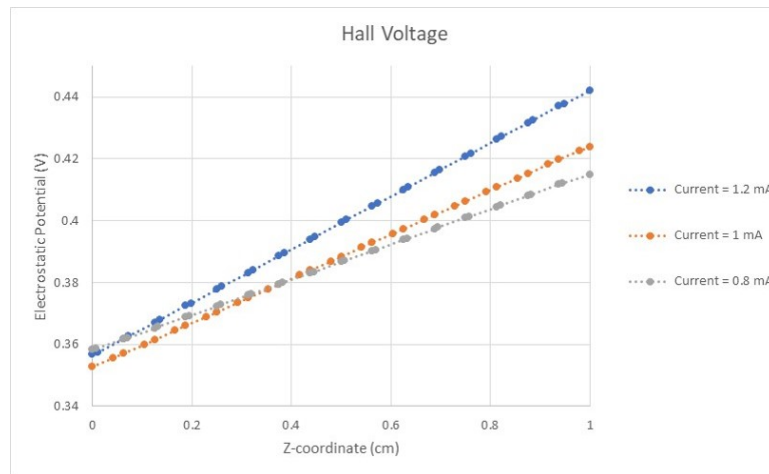


Figure 2.5 Spatial variation of electrostatic potential.

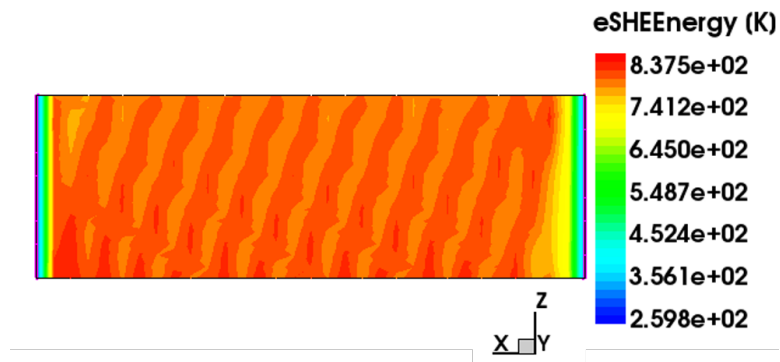


Figure 2.6 Electron energy colormap as calculated from spherical harmonic expansion of distribution function (at 1.2mA applied current).

CHAPTER

3

MONTE CARLO SIMULATIONS

3.1 Monte Carlo Method

The Monte Carlo method is used to simulate nonequilibrium transport in semiconductor materials and devices. The technique is based on generating random lattice walk by simulating free particle motion terminated by instantaneous random scattering events. The algorithm thus consists of injecting carriers through a contact with specific energy and momentum distributions, generating free flight times for each particle, changing energy and momentum during free flight, choosing the type of scattering occurring at the end of each free flight, changing the final energy and momentum of the particle after scattering and then repeating the process for the next free flight. The flowchart is given in figure 3.1.

Using this method, the time-dependent evolution of electron and hole distributions under the influence of a time-dependent driving force can be simulated. By sampling the particle motion at various times, we can estimate physical quantities of interest such as average drift velocity, average energy et cetera statistically.

To introduce carrier-carrier interactions, we need to couple the Monte Carlo program with a Poisson solver. Using spatial coordinates of the carriers, the local carrier density can be calculated at each time step, which can be used to obtain local electric field from the Poisson equation. This

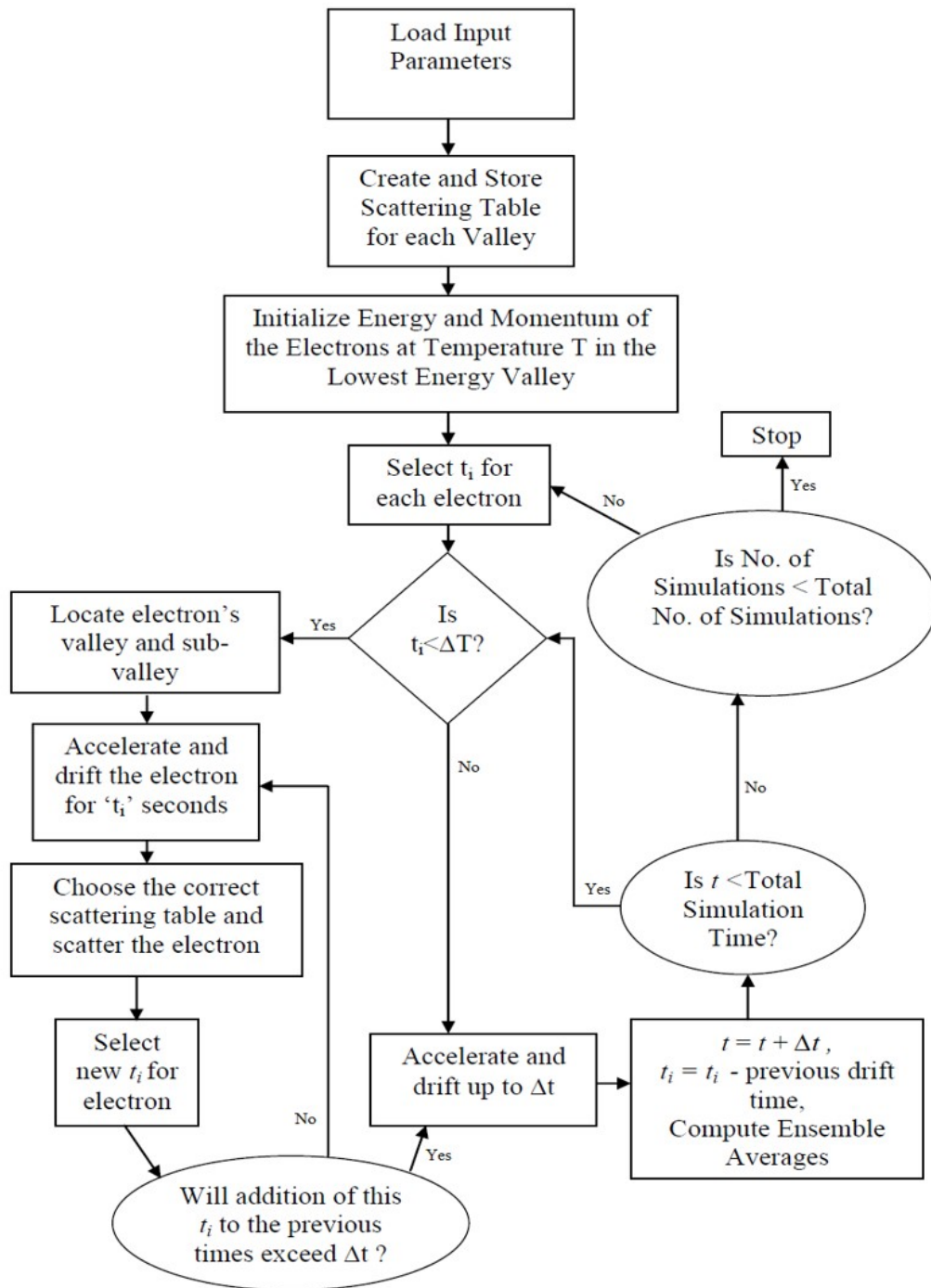


Figure 3.1 Flowchart depicting the steps in ensemble Monte Carlo simulation

approach prevents carrier accumulation however, choosing the correct boundary condition for the given application is very essential.

The complete benefits of using this method can be realized while studying transient transport in sub-micron devices or in high-field regime. Phonon transport can also be simulated using this technique. Other approaches used in semiconductor device modelling are *drift diffusion* and *hydrodynamic* models. These models are useful for large devices in which electric fields are not that high. A direct solution of Boltzmann Transport Equation (BTE) using Monte Carlo is necessary when higher order effects are to be included.

3.2 Fundamental Concepts

3.2.1 Transport Equation

The fundamental quantity being solved through the MC simulation is the distribution function $f(\mathbf{r}, \mathbf{k}, t)$ which is proportional to the density of electrons in 6-dimensional space (\mathbf{r}, \mathbf{k}) . Here, \mathbf{r} is the position vector for the carrier and \mathbf{k} is the momentum vector. The equation that determines the semiclassical transport is given by the Boltzmann equation:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{r}} f + \mathbf{k} \cdot \nabla_{\mathbf{k}} f = \left. \frac{df}{dt} \right|_{collision} \quad (3.1)$$

The first term in the LHS of the equation gives the temporal variation of electron distribution, the second term is for spatial variation of f due to temperature or concentration gradients and the third term is due to applied fields (electric/magnetic). The term on the RHS is the rate of change of f due to collisions.

3.2.2 Energy Bands

The relationship between the electron energy, ϵ and its momentum \mathbf{k} specifies the band structure of the material. The exact band structure of a material is complex and difficult to deal with. However since the free carriers are present usually near the conduction band minima and the valence band maxima, we can simplify the problem immensely by approximating $\epsilon(\mathbf{k})$ to be quadratic in the region of interest. Two types of approximations for the band structure are:

1. **Parabolic bands**

The energy-momentum relation is given by:

$$\epsilon(\mathbf{k}) = \frac{\hbar^2 \mathbf{k}^2}{2m_0^*} \quad (3.2)$$

Here, m_0^* is the effective mass at the band minima/maxima. The particle group velocity is given by:

$$\mathbf{v}_g = \frac{1}{\hbar} \nabla_{\mathbf{k}} \epsilon(\mathbf{k}) = \frac{\hbar \mathbf{k}}{m_0^*} \quad (3.3)$$

2. Non-parabolic bands

For non-parabolic bands, the energy-momentum relation is given by:

$$\epsilon(1 + \alpha\epsilon) = \frac{\hbar^2 \mathbf{k}^2}{2m_0^*} \quad (3.4)$$

Here, α is the non-parabolicity parameter. The group velocity can be calculated from the gradient of energy as:

$$\mathbf{v}_g = \frac{1}{\hbar} \nabla_{\mathbf{k}} \epsilon(\mathbf{k}) = \frac{\hbar \mathbf{k}}{m_0^*(1 + 2\alpha\epsilon)} \quad (3.5)$$

If the effective mass for the carrier is assumed to be the same in all three directions in k-space, we get spherical equienergetic surfaces. However, if due to anisotropy, the effective masses are different in all three directions for the given valley, we get ellipsoidal equienergetic surfaces. For more information about band structures, see ref [Vas].

3.3 Explanation of the Monte Carlo Code

[JL12], [JR83] and [Sno85] provide a comprehensive review of charge transport in semiconductors, Monte Carlo method and semiconductor devices. [VG08] explains the Monte Carlo code for GaAs based bulk material system and the code available on the Nanohub website was used as a starting point for this work. Different parts of the code are explained in this section and the MATLAB code is provided in the appendix section.

3.3.1 Parameter File

The parameter file contains the values of global variables like temperature, universal constants, material specific properties like the band gap, deformation potentials, effective masses, Debye length and device related parameters such as dimensions and mesh size.

3.3.2 sc_table

The function `sc_table()` calculates scattering rates for each type of scattering versus energy and writes them to a file. It also constructs a look-up table stored as the variable `scatt_table` consisting of normalized cumulative scattering rates versus carrier energies. The normalization is done with respect to the highest scattering rate i.e. `tau_max`. This table is used by the function `scatter_carrier()` to select a scattering event based on a random number generator.

Following are the types of scattering mechanisms and the formulae used to calculate them. The derivations for these relations can be found in [JL12].

3.3.2.1 Types of scattering

1. Intravalley Inelastic Acoustic Phonon Scattering

The rates of scattering by acoustic phonons in case of ellipsoidal nonparabolic bands is given by the following equation:

$$P_{e,ac}(\epsilon) = \frac{m_d^{1/2}(K_B T)^3 V_1^2}{2^{5/2}\pi\hbar^4 u^4 \rho} \gamma^{-1/2}(\epsilon) \times \begin{cases} (1 + 2\alpha\epsilon)[F_1(x_{2,a}) - F_1(x_{1,a})] + 2\alpha K_B T [F_2(x_{2,a}) - F_2(x_{1,a})] \\ (1 + 2\alpha\epsilon)[G_1(x_{2,e}) - G_1(x_{1,e})] - 2\alpha K_B T [G_2(x_{2,e}) - G_2(x_{1,e})] \end{cases} \quad (3.6)$$

where m_d is the density of states effective mass, V_1 is the deformation potential, u is the sound velocity in the material, ρ is the mass density, α is the nonparabolicity and the definitions of the functions F_1, F_2, G_1, G_2 can be found in the appendix.

The upper part of the equation is for acoustic phonon absorption and the lower one is for emission and $x_{1,a}, x_{2,a}, x_{1,e}, x_{2,e}$ are the limits of integration for absorption and emission respectively and are also given in the appendix.

2. Intravalley Inelastic Optical Phonon Scattering

Similarly, the scattering rate for optical phonons for ellipsoidal nonparabolic bands is given by the following equation:

$$P_{e,op}(\epsilon) = \frac{(D_t K)^2 m_d^{3/2}}{2^{1/2}\pi\hbar^3 \rho \omega_{op}} \gamma^{1/2}(\epsilon\pi\hbar\omega_{op}) [1 + 2\alpha(\epsilon \pm \hbar\omega_{op})] \begin{cases} N_{op} \\ N_{op} + 1 \end{cases} \quad (3.7)$$

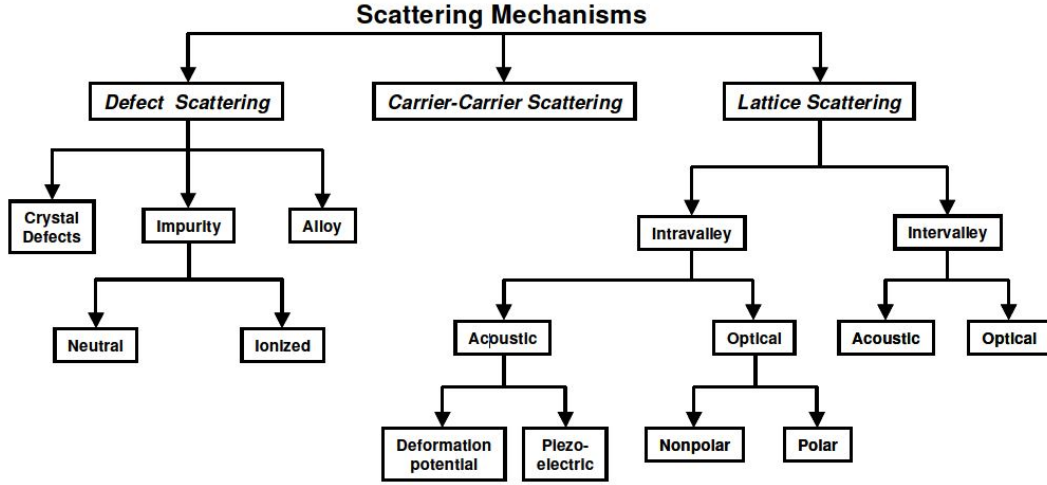


Figure 3.2 Classification of scattering mechanisms. Picture taken from [Vas11]

The factor $(D_t K)$ is the optical coupling constant and N_{op} is the phonon occupation number given by $1/\exp(\hbar\omega_{op}/K_B T)$. The upper part of the equation is optical phonon absorption rate and the lower part is optical phonon emission.

3. Ionized Impurity scattering

The ionized impurity scattering is elastic in nature and unlike phonon scattering, it is not accompanied by energy loss. Two formulations are known for calculating this type of scattering rate: the Conwell- Weisskopf (CW) approach and the Brooks- Herring (BH) approach. We use the BH formulation which is given by:

$$P_{e,I}^{BH}(\epsilon) = \frac{2^{5/2} \pi n_I Z^2 e^4}{\kappa^2 \epsilon_\beta^{1/2} m_d^{1/2}} \gamma^{1/2}(\epsilon) \frac{1 + 2\alpha\epsilon}{1 + 4\frac{\gamma(\epsilon)}{\epsilon_\beta}} \quad (3.8)$$

where κ is the dielectric constant of the material, n_I is the impurity concentration, Z is the charge on each impurity and $\epsilon_\beta = \frac{\hbar^2 \beta^2}{2m_d}$.

Apart from the types of scattering mentioned above, there are numerous other mechanisms that can be included based on the material system used. Figure 3.7 presents a classification of different types of scattering mechanisms.

3.3.3 init

The `init()` function is responsible for initializing the momenta, energies and positions of the particles such that they satisfy the required distributions. For energy and momentum, a Maxwell-Boltzmann distribution is used. For initial positions, a uniform distribution is assumed within the device. Since we are injecting electrons in +x direction, the initial momentum values for k_x are assumed to be all positive as shown in the figure.

While generally the steady state distributions are not dependent upon the initial conditions under which an electron is injected into the semiconductor, these initial conditions are very important with respect to transient transport studies.

Using a random number generator, we can generate a random number between 0 and 1 and then use the direct technique to pick random values for energy from a Maxwellian distribution. The direct technique is described below.

3.3.3.1 Generating random free flight times

Let the probability that a particle with wave vector $k(t)$ suffers a collision during a time interval dt be $P[k(t)] dt$. Probability that the particle which underwent a collision at $t=0$ has not yet suffered a collision after a time t is given by $\exp(-\int_0^t P[k(t')] dt')$. Probability that the particle undergoes a collision during dt around t is $P[k(t)] \exp(-\int_0^t P[k(t')] dt')$. To simplify the problem of generating stochastic free flight times, we assume that $P[k(t)]$ is constant in the region of k -space of interest and is equal to $1/t_{max}$. This modifies the above scattering rate equation to $\frac{1}{t_{max}} e^{(-t/t_{max})}$.

Applying the direct technique to generate times t satisfying the above relation from a uniformly distributed random number r . Equating areas under the curves from figures (as per Direct Technique), we get:

$$\int_0^R dr = \int_0^{t_r} \frac{1}{t_{max}} e^{-t/t_{max}} dt \quad (3.9)$$

$$R = 1 - e^{-t_r/t_{max}} \quad (3.10)$$

Re-arranging the above equation, we get: free flight time, $t_r = -t_{max} \ln R$, where t_{max} is the total rate (self-scattering plus real scattering). Self-scattering is a fictitious scattering event, introduced by Rees in 1968 [1]. The self-scattering rate at each energy adjusts itself in a way such that the total rate is constant. The total constant rate is chosen in the beginning of the program, while populating the scattering rate table as the largest value in the energy region of interest.

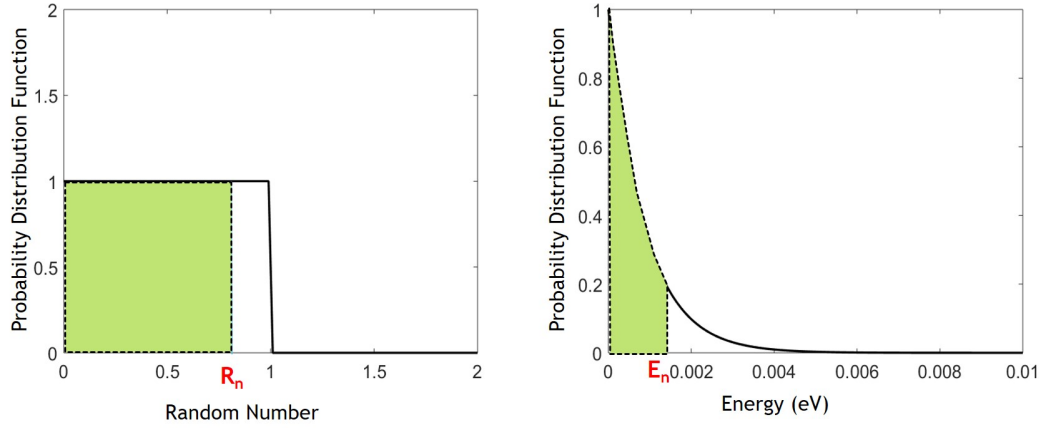


Figure 3.3 Graphical representation of the direct technique for choosing a random energy value from a Maxwellian distribution using a random number generator with uniform probability distribution.

3.3.3.2 Generating energy and momentum distributions

Energy is chosen using a random number generator using the following formula:

$$\epsilon = \frac{3}{2} K_B T \ln\left(\frac{1}{r}\right) \quad 0 < r \leq 1 \quad (3.11)$$

This yields a Maxwellian distribution for energy. The direct technique for choosing a random energy value from a given distribution using a random number generator is shown in figure 3.3. From the chosen value of energy, three random numbers, k_x , k_y and k_z are selected satisfying the dispersion relation:

$$k = \sqrt{\frac{2m\epsilon(1 + \alpha\epsilon)}{\hbar^2}} \quad (3.12)$$

The dispersion relation accounts for the non-parabolicity of the conduction band, given by α . A random number is used to generate a random value for k_x :

$$k_x = k(1 - 2r_1) \quad 0 \leq r_1 \leq 1$$

Then k_y and k_z are selected using another random number, r_2 where $0 \leq r_2 \leq 1$:

$$k_y = k \sqrt{1 - (1 - 2r_1)^2} \cos(2\pi r_2)$$

$$k_z = k \sqrt{1 - (1 - 2r_1)^2} \sin(2\pi r_2)$$

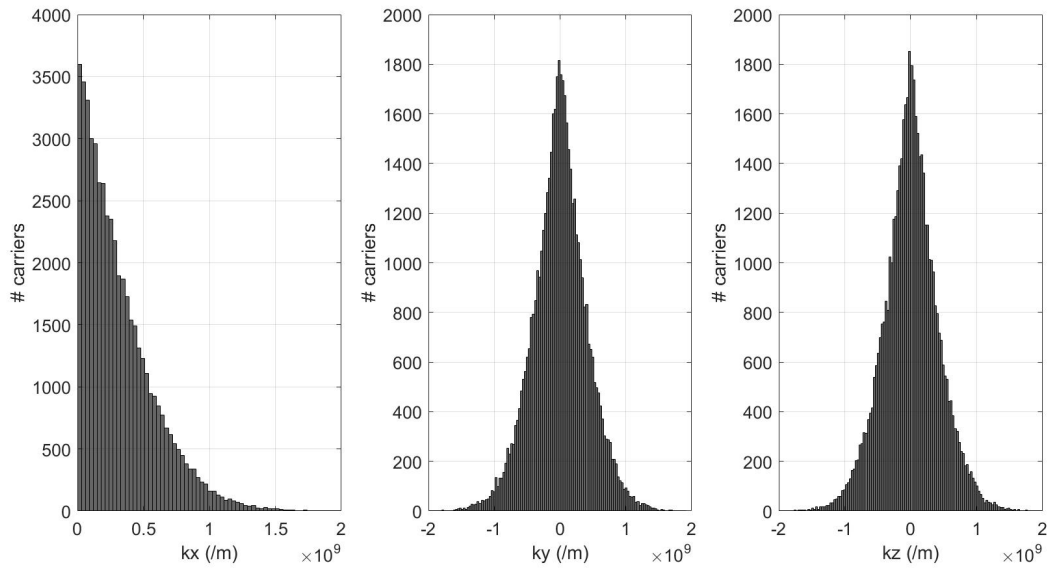


Figure 3.4 Initial distributions for momentum vector k_x , k_y and k_z .

The k_x , k_y and k_z satisfy the condition $k_x^2 + k_y^2 + k_z^2 = k^2$. We keep only positive values of k_x since we are injecting electrons at $t=0$. Spatially, all the electrons are assumed to be randomly scattered throughout the region of interest. The initial distributions generated from the code are shown in figures 3.4 and 3.5.

3.3.4 free_flight_scatter

This function is the heart of the Monte Carlo code. This is where carriers undergo changes in position, momentum and energy after being subjected to either a free flight or a scattering event. The flowchart for this function is shown in figure 3.6. The calls to the `drift()` and `scatter_carrier()` functions are done through `free_flight_scatter()` function.

3.3.5 drift

The `drift()` function is the part where the particle undergoes free flight subject to external forces. The derivations for the formulae used in this function are presented below.

Instead of having a spherical symmetry to the E-k relationship, we assume different effective

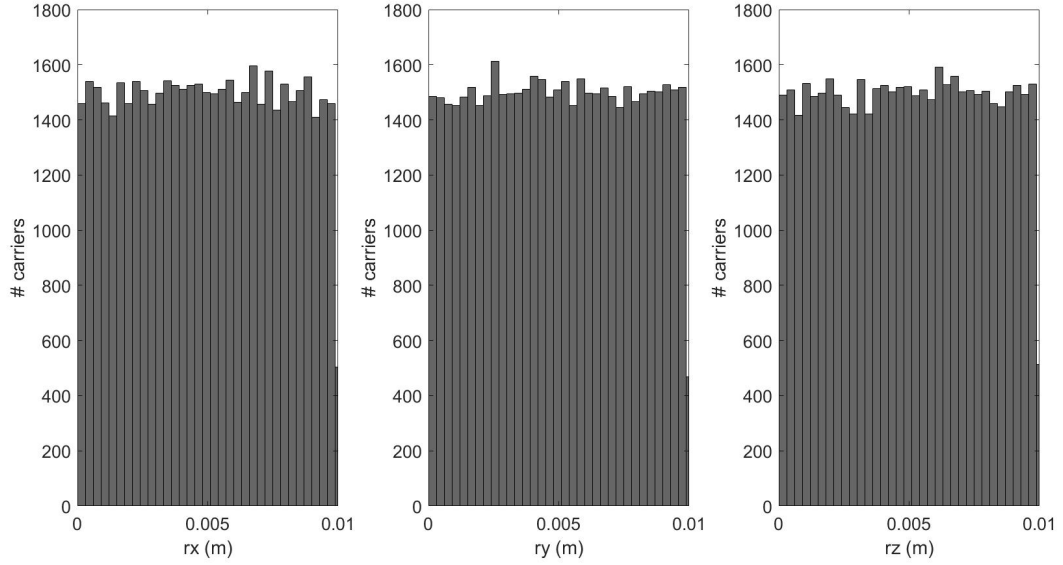


Figure 3.5 Initial distributions for position vector r_x , r_y and r_z .

masses in all three directions i.e.

$$\epsilon(\mathbf{k}) = \frac{\hbar^2}{2} \left(\frac{k_x^2}{m_{t1}} + \frac{k_y^2}{m_{t2}} + \frac{k_z^2}{m_l} \right) \quad (3.13)$$

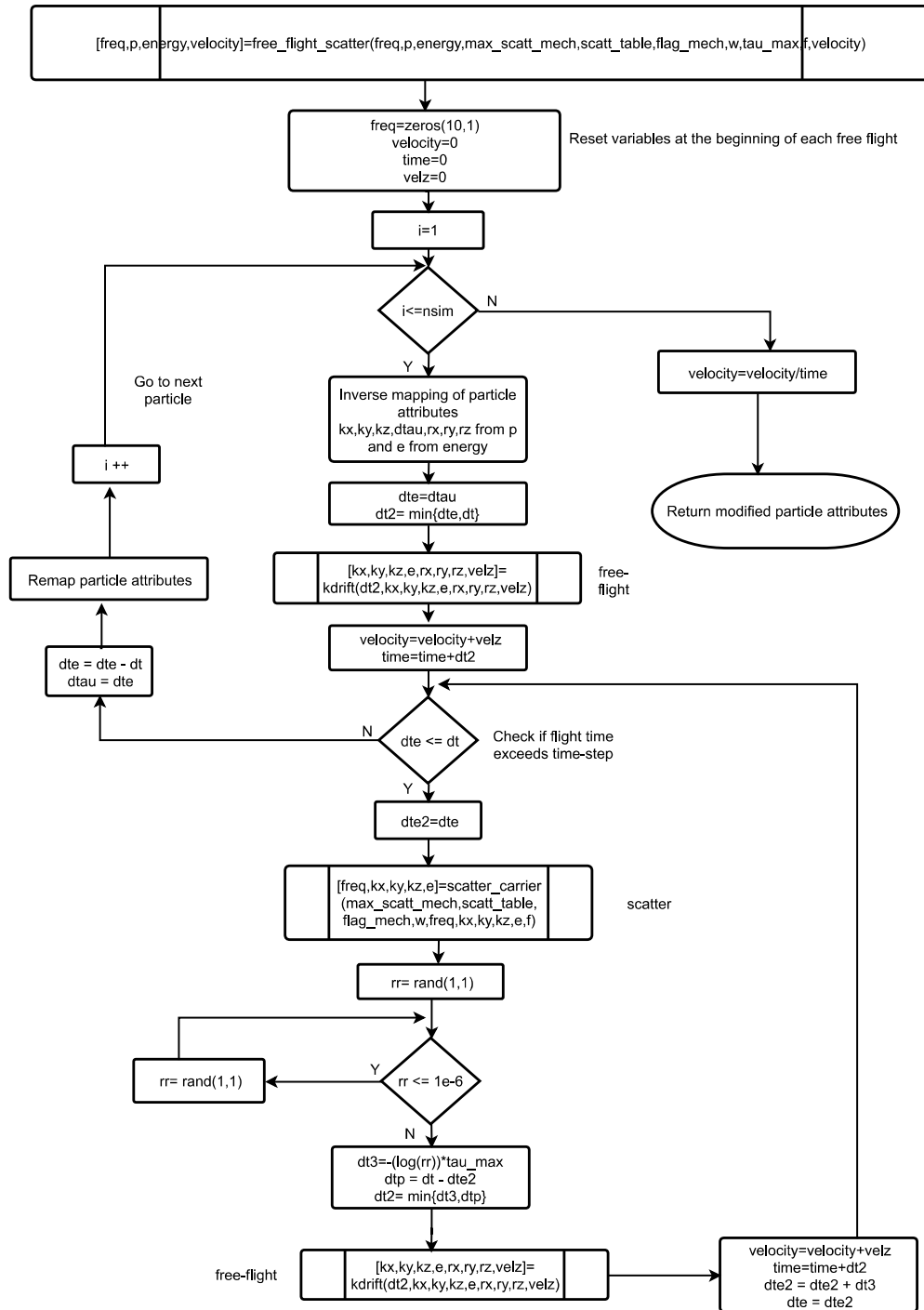
where m_{t1} , m_{t2} and m_l are the transverse and longitudinal masses respectively. Thus in order to simplify the analytical calculations, we use the *Herring-Vogt transformation* to reduce the ellipsoidal equienergetic surfaces to spheres, which gives us the wave vector \mathbf{k}^* in the starred k-space:

$$k_i^* = T_{ij} k_j \quad (3.14)$$

The transformation matrix is of the following form:

$$T = \begin{vmatrix} \sqrt{\frac{m_0}{m_{t1}}} & 0 & 0 \\ 0 & \sqrt{\frac{m_0}{m_{t2}}} & 0 \\ 0 & 0 & \sqrt{\frac{m_0}{m_l}} \end{vmatrix} \quad (3.15)$$

Figure 3.6 Flowchart for free_flight_scatter() function



Thus we get the following components of momentum in the starred space:

$$k_x^* = \sqrt{\frac{m_0}{m_{t1}}} k_x \quad (3.16)$$

$$k_y^* = \sqrt{\frac{m_0}{m_{t2}}} k_y \quad (3.17)$$

$$k_z^* = \sqrt{\frac{m_0}{m_l}} k_z \quad (3.18)$$

Extracting k_x , k_y and k_z from the above equations and substituting in the equation (3.13), we get a reduced expression for energy in terms of \mathbf{k}^* :

$$\epsilon(\mathbf{k}^*) = \frac{\hbar^2}{2m_0} \mathbf{k}^{*2} \quad (3.19)$$

Accounting for non-parabolicity of conduction band, we get:

$$\gamma(\mathbf{k}^*) = \epsilon(1 + \alpha\epsilon) = \frac{\hbar^2}{2m_0} \mathbf{k}^{*2} \quad (3.20)$$

Now, we know that the carrier group velocity in the normal k-space is given by:

$$\mathbf{v}(\mathbf{k}) = \frac{1}{\hbar} \frac{\partial \epsilon}{\partial \mathbf{k}} = \frac{\hbar \mathbf{k}}{m(1 + 2\alpha\epsilon)} \quad (3.21)$$

The carrier velocity components in the real space can then be expressed in terms of the starred momentum by the following equations:

$$\mathbf{v}_x = \frac{\hbar \mathbf{k}_x^*}{m_{t1}(1 + 2\alpha\epsilon)} \sqrt{\frac{m_{t1}}{m_0}} \quad (3.22)$$

$$\mathbf{v}_y = \frac{\hbar \mathbf{k}_y^*}{m_{t2}(1 + 2\alpha\epsilon)} \sqrt{\frac{m_{t2}}{m_0}} \quad (3.23)$$

$$\mathbf{v}_z = \frac{\hbar \mathbf{k}_z^*}{m_l(1 + 2\alpha\epsilon)} \sqrt{\frac{m_l}{m_0}} \quad (3.24)$$

The applied external forces such as the electric and magnetic fields will change the momentum according to the relation:

$$\mathbf{F}_x = \hbar \frac{dk_x}{dt} = \hbar \sqrt{\frac{m_{t1}}{m_0}} \frac{dk_x^*}{dt} \quad (3.25)$$

Hence we get the following incremental values for change in momentum after every free flight:

$$dk_x^* = \frac{1}{\hbar} \sqrt{\frac{m_0}{m_{t1}}} F_x dt \quad (3.26)$$

$$dk_y^* = \frac{1}{\hbar} \sqrt{\frac{m_0}{m_{t2}}} F_y dt \quad (3.27)$$

$$dk_z^* = \frac{1}{\hbar} \sqrt{\frac{m_0}{m_l}} F_z dt \quad (3.28)$$

Adding these changes to the original momentum, we get the final momentum in the starred space which can be used to calculate $\gamma(k^*)$ in equation (3.20). The carrier energy can be extracted from $\gamma(k^*)$ using the formula:

$$\epsilon = \frac{2\gamma}{1 + \sqrt{1 + 4\alpha\gamma}} \quad (3.29)$$

The changes in position of the carriers can be found by using equations from Newtonian classical mechanics.

$$\Delta x = \mathbf{v}_x \Delta t + \frac{1}{2} \frac{F_x}{m_{t1}} \Delta t^2 \quad (3.30)$$

$$\Delta y = \mathbf{v}_y \Delta t + \frac{1}{2} \frac{F_y}{m_{t2}} \Delta t^2 \quad (3.31)$$

$$\Delta z = \mathbf{v}_z \Delta t + \frac{1}{2} \frac{F_z}{m_l} \Delta t^2 \quad (3.32)$$

3.3.6 scatter_carrier

This function selects a random scattering event based on the look-up table `scatt_table`. The `scatt_table` consists of cumulative normalized probabilities of each scattering event. For instance, if the probability of occurrence of i^{th} event is P_i then the i^{th} column of `scatt_table` corresponds to the sum $P_1 + P_2 + P_3 + \dots + P_i$. Then a random number is chosen and if j is the event such that $P_1 + P_2 + P_3 + \dots + P_j$ is the first partial sum greater than the random number, then j^{th} event is picked. This is pictorially represented in figure (). The flowchart for the actual program is also shown in figure 3.7.

After picking an event, the next step is to choose the final state of the carrier, the methodology for which differs for different types of scattering.

1. Optical Phonon scattering

The optical phonon scattering is assumed to be isotropic i.e. the final momentum does not

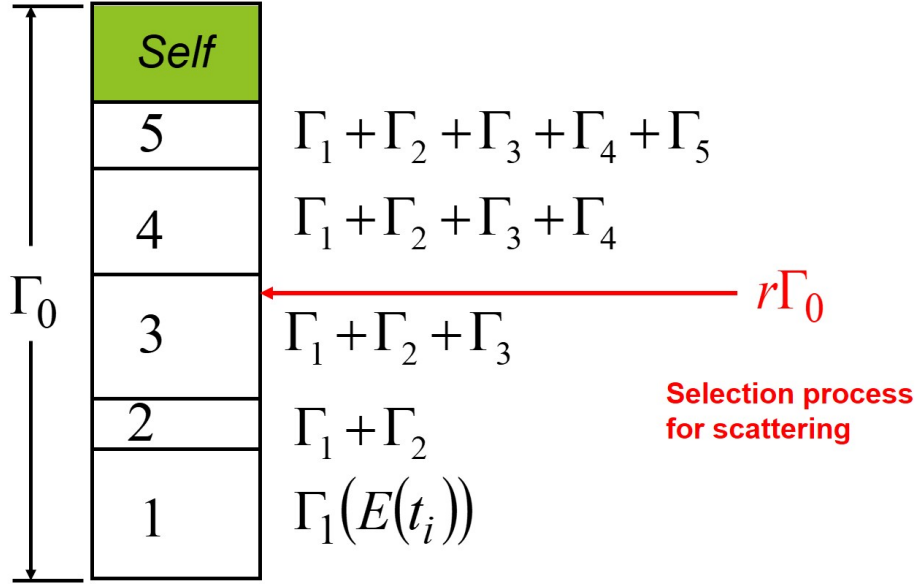


Figure 3.7 Pictorial depiction of how a random scattering event is chosen out of a discrete set of mechanisms. Picture taken from [Vas11].

have an angle dependence. Since energy of an optical phonon is known, we can easily determine the final energy of the carrier based on whether a phonon was absorbed or emitted. After calculating final energy, ϵ , the wave vector \mathbf{k}^* can be determined from ϵ by using the nonparabolic dispersion relation. The x,y and z components of the wave vector can then be randomly chosen from \mathbf{k}^* . The derivation is as follows:

$$\begin{aligned} \epsilon_f &= \epsilon_i \pm \hbar\omega_{op} \\ k_f &= \frac{\sqrt{2m_0\epsilon_f(1+\alpha\epsilon_f)}}{\hbar} \\ \phi &= 2\pi R_1 \quad R_1 \in (0,1) \\ \cos\theta &= 1 - 2R_2 \quad R_2 \in (0,1) \\ \sin\theta &= \sqrt{1 - \cos^2\theta} \\ k_x &= k_f \sin\theta \cos\phi \\ k_y &= k_f \sin\theta \sin\phi \\ k_z &= k_f \cos\theta \end{aligned}$$

2. Acoustic Phonon Scattering

Because of phonon dispersion, we don't know the exact energy associated with acoustic phonons, unlike the optical phonon. So the phonon wave vector is sampled from a distribution and the final momentum is calculated by rotating the coordinate system. The derivation is as follows:

$$\begin{aligned} \cos\theta_0 &= \frac{k_{z,i}}{k_i} \\ \sin\theta_0 &= \frac{\sqrt{k_{x,i}^2 + k_{y,i}^2}}{k_i} \\ \cos\phi_0 &= \frac{k_{x,i}}{\sqrt{k_{x,i}^2 + k_{y,i}^2}} \\ \sin\phi_0 &= \frac{k_{y,i}}{\sqrt{k_{x,i}^2 + k_{y,i}^2}} \\ k_f &= \frac{\sqrt{2m\epsilon_f(1+\alpha\epsilon_f)}}{\hbar} \end{aligned}$$

The angle θ for scattering can be found using the rule of cosines:

$$\begin{aligned} \cos\theta &= \frac{k_f^2 + k_i^2 - q_{ph}^2}{2k_f k_i} \\ \sin\theta &= \sqrt{1 - \cos^2\theta} \\ \phi &= 2\pi R_1 \quad R_1 \in (0, 1) \end{aligned}$$

Then the x, y and z coordinates of momentum in rotated space are given by:

$$\begin{aligned} k_{xp} &= k_f \sin\theta \cos\phi \\ k_{yp} &= k_f \sin\theta \sin\phi \\ k_{zp} &= k_f \cos\theta \end{aligned}$$

Switching to original coordinate system:

$$\begin{aligned}k_x &= k_{xp} \cos \phi_0 \cos \theta_0 - k_{yp} \sin \phi_0 + k_{zp} \cos \phi_0 \sin \theta_0 \\k_y &= k_{xp} \sin \phi_0 \cos \theta_0 + k_{yp} \cos \phi_0 + k_{zp} \sin \phi_0 \sin \theta_0 \\k_z &= -k_{xp} \sin \theta_0 + k_{zp} \cos \theta_0\end{aligned}$$

3. Ionized Impurity scattering

In case of ionized impurity scattering, there is no energy loss and the procedure is same as that for acoustic phonon scattering as mentioned in previous section. The only difference is the calculation of scattering angle, which is given by:

$$\cos \theta = 1 - \frac{2r}{1 + 4k^2 L_D^2 (1 - r)}$$

3.3.7 Poisson solver

For accurate treatment of particle dynamics, we need to couple MC with a Poisson solver. For our application, a 1D Poisson solver in z-direction should suffice since the Hall field and Lorentz force are in the z-direction. Without solving the Poisson equation in the z-direction for electric field and potential, a much larger accumulation of electrons on one side will be observed, since there is no other mechanism accounting for carrier-carrier scattering.

The Poisson solver uses finite difference scheme to solve for potential and electric field across the z-direction. The first step is to define a 1D spatial grid in the z-direction. Debye length should be a good starting point for the mesh size. Next, charge density in each slice needs to be determined from the spatial coordinates of the particles. This gives us $\rho(z)$ and we know that the Poisson equation is:

$$\frac{d^2 V}{dz^2} = \frac{\rho(z)}{\epsilon_s} \quad (3.33)$$

Where V is the potential and ϵ_s is the dielectric constant for the material. Using finite difference scheme, the second derivative term can be replaced as follows:

$$\frac{V_{i+1} - 2V_i + V_{i-1}}{\Delta z^2} = \frac{\rho(z)}{\epsilon_s} \quad (3.34)$$

With $1 \leq i \leq I$ where L is the total length of the region of interest and $L = I \Delta z$. Thus, we obtain a set

of equations from every point on the grid:

$$\begin{aligned}
 V_1 &= w_1 \\
 V_1 - 2V_2 + V_3 &= \Delta z^2 \rho_2 / \epsilon_s \\
 V_2 - 2V_3 + V_4 &= \Delta z^2 \rho_3 / \epsilon_s \\
 &\dots \\
 V_{I-2} - 2V_{I-1} + V_I &= \Delta z^2 \rho_{I-1} / \epsilon_s \\
 V_I &= w_I
 \end{aligned}$$

In the matrix notation, we can write:

$$\mathbf{AV} = \mathbf{w} \quad (3.35)$$

And the matrix \mathbf{A} has the form:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & -2 & 1 & 0 & \dots \\ 0 & 1 & -2 & 1 & \dots \\ & & \dots & & \\ 0 & \dots & 1 & -2 & 1 \\ 0 & \dots & 0 & 0 & 1 \end{bmatrix} \quad (3.36)$$

\mathbf{A} is a sparse matrix since it only has elements along the central three diagonals of the matrix (tridiagonal matrix). Inverting a matrix is computationally expensive and it is advisable to invert the matrix at the beginning of the simulation to save time. Electric field as a function of z i.e. $E_z(z)$ can be calculated from the potential by using the gradient formula:

$$E_z = -\nabla V(z) = \frac{V_{i+1} - V_{i-1}}{2\Delta z} \quad (3.37)$$

Another important consideration while solving the Poisson equation is setting the boundary condition i.e. choosing w_1 and w_I . If *Dirichlet* boundary condition is used on both sides, then w_1 and w_I are set to a constant value. Another possibility is using the *Neumann* boundary condition in which the derivative of potential (and thus electric field) at the surface is set to zero i.e. $\frac{V_2 - V_1}{2\Delta z} = 0$ or $V_1 = V_2$ and $V_I = V_{I-1}$. Using Neumann BC on both sides results in a singular matrix which can't be inverted. So to avoid this problem, we use Dirichlet BC at one end of the device and Neumann BC

on the other. So the matrix form of our Poisson equation is:

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & -2 & 1 & 0 & \dots \\ 0 & 1 & -2 & 1 & \dots \\ & & & \dots & \\ 0 & \dots & 1 & -2 & 1 \\ 0 & \dots & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ \dots \\ V_{I-1} \\ V_I \end{bmatrix} = \begin{bmatrix} 0 \\ \Delta z^2 \rho_2 / \epsilon_s \\ \Delta z^2 \rho_3 / \epsilon_s \\ \dots \\ \Delta z^2 \rho_{I-1} / \epsilon_s \\ 0 \end{bmatrix} \quad (3.38)$$

3.4 Results and problems encountered while modelling a thermomagnetic device using Monte Carlo approach

The material used for this code was Antimony Telluride and the results of analytical calculations for electrical conductivity, Seebeck coefficient, Nernst coefficient and Ettingshausen coefficient are shown in figures 3.8, 3.9, 3.10 and 3.11 respectively. These calculations were done using a pre-existing MATLAB tool developed by Dr. Vashae's group to serve as a reference and give us an idea about optimum conditions to obtain maximum ΔT .

3.4.1 Convergence and noise

Before we start averaging the desired quantities, it is of utmost importance to ensure that the average energy and velocity of particles converge. To do this, we have two for-loops in our code. During the first for-loop, we monitor how average energy and drift velocity reach a steady state and in the next for-loop, we start averaging the desired properties.

Convergence depends of a lot of different factors like the applied electric, magnetic fields and the temperature. An analysis was done to show how average energy and velocity change with time for different external fields and temperatures. Following conclusions can be made on the basis of the study:

1. At higher electric fields, average energy for the carriers takes less number of iterations to converge.
2. The final value at which average energy settles is lower at higher temperature. This might be because of higher scattering rates at elevated temperatures, which does not let the electrons accelerate and gain kinetic energy.

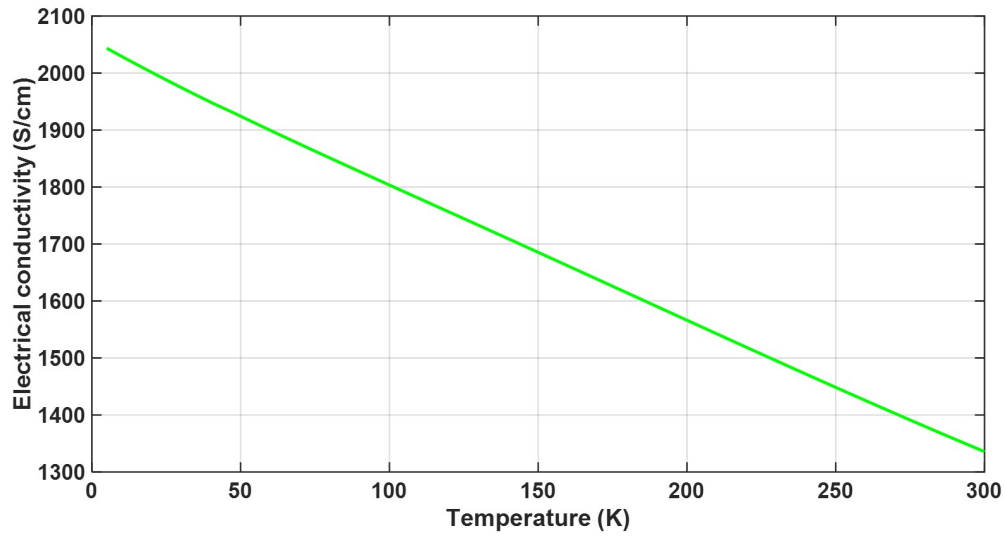


Figure 3.8 Electrical conductivity versus temperature curve for Sb_2Te_3

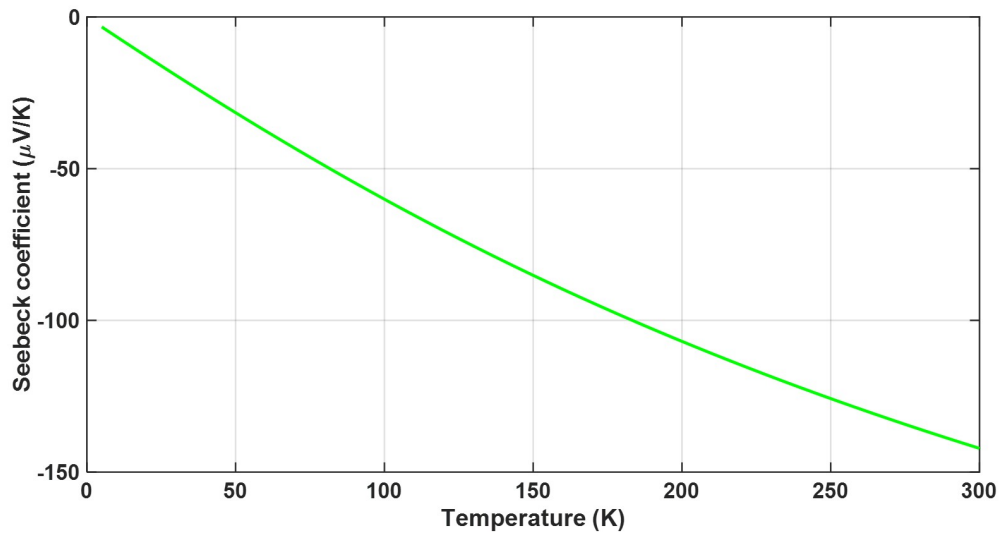


Figure 3.9 Seebeck coefficient versus temperature curve for Sb_2Te_3

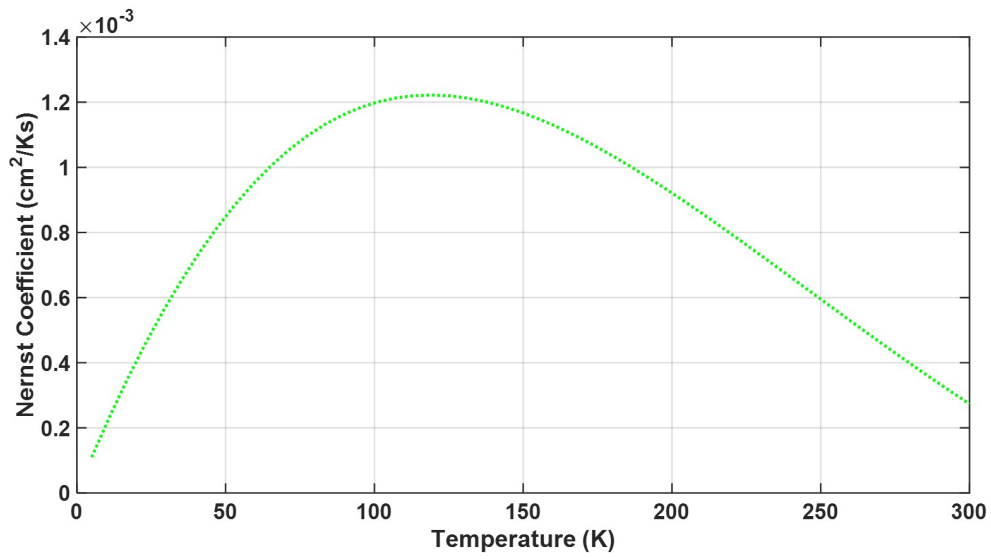


Figure 3.10 Nernst coefficient versus temperature curve for Sb_2Te_3

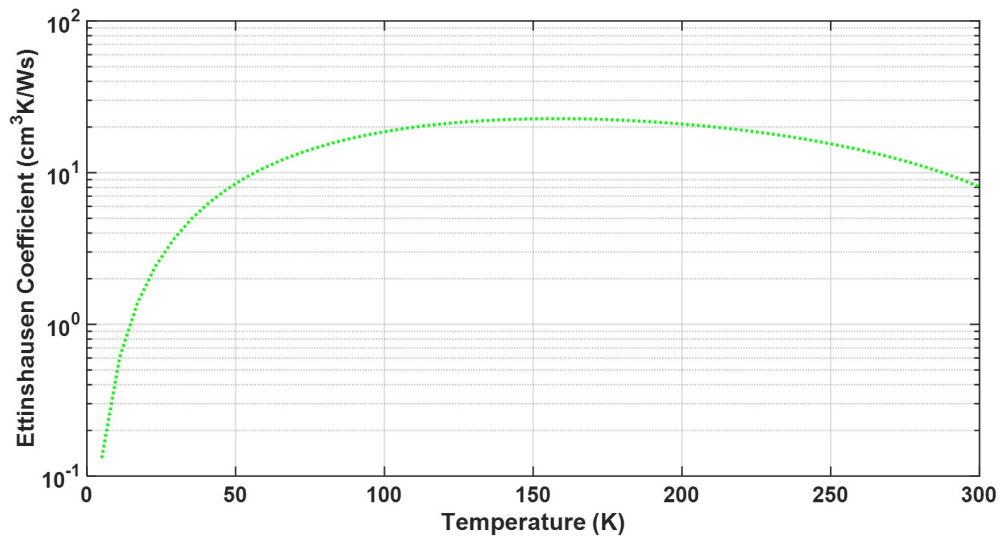


Figure 3.11 Ettingshausen coefficient versus temperature curve for Sb_2Te_3

3. The number of particles simulated affects the noise level significantly. Since the Monte Carlo method relies heavily on statistical averaging, we need to make sure that a good number of particles are simulated within a given energy range and space grid. The results show that without any spatial grid, we need atleast 20,000 particles to overcome the large noise level. A similar analysis needs to be done within a differential spatial element to find out the optimum number of particles to eliminate noise.

3.4.2 MC-Poisson coupling

The issues with Monte Carlo- Poisson solver coupling are mentioned in [Ven89] for sub-micron devices. We observed that small variations in spatial distribution of charge result in large changes in the electrostatic potential obtained from Poisson equation. As mentioned in the paper, one way to stabilize the potential is to use quasi Fermi levels instead of actual carrier concentration and a nonlinear Poisson solver.

3.4.3 Device scaling for thermomagnetic application

The analytical calculations show that the device dimensions to obtain an appreciable ΔT across the thermomagnetic cooler are well out of micron-range. In fact, the sizes in different dimensions need to be in the range of a few millimeters. From our findings, we conclude that the simulation time is not enough to enable the carriers to travel such a large distance. To counter this issue, the changes in displacement were scaled up by a very large factor. This approach will however amplify the noise by a large factor as well.

These problems suggest that Monte Carlo method alone might not be the most suitable approach for this particular problem. It is more suited to applications involving sub-micron devices, heterostructures and more complex material systems. In order to simulate a thermomagnetic cooler with higher accuracy, it would be a better idea to couple the Monte Carlo with other tools like Sentaurus TCAD or COMSOL.

CHAPTER

4

CONCLUSION

In summary, during the course of this work, different types of solid state cooling mechanisms were studied. Inspiration from the success of thermoelectric cooling led to the study of thermomagnetic cooling. Different approaches were used in order to carry out that study. The Monte Carlo method provides a more rigorous way of studying the physical phenomena and we get to control how each particle behaves. Another way is to use an existing simulator, Synopsys Sentaurus TCAD in this case. While we may not be able to observe the behavior of each particle, we can certainly choose physical models that fit our description. Although there were limitations to using both these approaches, this shouldn't discourage the readers from studying the thermomagnetic effect.

The limitations for using the Sentaurus simulator are:

1. Lack of compatibility between magnetic field and hydrodynamic models.
2. Inability to define floating temperature boundaries.
3. There is no reliable way to track electron/hole temperatures without including the hydrodynamic model.

The limitations for using the Monte Carlo approach was:

1. Convergence of average energy and velocities is the most important criteria to get reliable results. There needs to be sufficient number of particles at each energy level and with in each spatial element in order for the ensemble to give an accurate prediction. The number of iterations required for convergence changes with applied external field, so it needs to be monitored carefully as well.
2. It is difficult to obtain coupling between the Monte Carlo and Poisson solver because of the random statistical noise in carrier density. This issue might be solved by using higher number of particles or by using quasi Fermi levels instead of carrier densities (and a non linear Poisson solver).
3. In order to get appreciable ΔT , we need to optimize the structure we're simulating. If we use a larger structure, we will encounter more noise and the simulations will require longer times.

Due to above mentioned limitations, it is strongly suggested to find alternative methods to study thermomagnetic effects theoretically. It would also be helpful to consider experiments done in this area and design the simulation as close to the reality as possible. For instance, ambipolar materials yield higher figures-of-merit and these models could be extended to study transport in presence of both electrons and holes.

BIBLIOGRAPHY

- [Del62] Delves, R. "The prospects for Ettingshausen and Peltier cooling at low temperatures". *British Journal of Applied Physics* **13.9** (1962), p. 440.
- [ES62] El-Saden, M. R. "Theory of the Ettingshausen Cooler". *Journal of Applied Physics* **33.5** (1962), pp. 1800–1803.
- [Gol95] Goldsmid, H. "Thermomagnetic phenomena". *CRC Handbook of Thermoelectrics* edited by DM Rowe (CRC Press, Boca Raton, 1994) (1995), p. 75.
- [Har63] Harman, T. "Theory of the infinite stage Nernst-Ettingshausen refrigerator". *Advanced energy conversion* **3.4** (1963), pp. 667–676.
- [Haw64] Hawkins, S. et al. "Low-Temperature Ettingshausen Coolers". *Advances in Cryogenic Engineering*. Springer, 1964, pp. 367–378.
- [JL12] Jacoboni, C. & Lugli, P. *The Monte Carlo method for semiconductor device simulation*. Springer Science & Business Media, 2012.
- [JR83] Jacoboni, C. & Reggiani, L. "The Monte Carlo method for the solution of charge transport in semiconductors with applications to covalent materials". *Reviews of Modern Physics* **55.3** (1983), p. 645.
- [Lee12] Lee, J.-H. et al. "Optimal emitter-collector gap for thermionic energy converters". *Applied Physics Letters* **100.17** (2012), p. 173904.
- [PB10] Pichanusakorn, P. & Bandaru, P. "Nanostructured thermoelectrics". *Materials Science and Engineering: R: Reports* **67.2** (2010), pp. 19–63.
- [Sno85] Snowden, C. M. "Semiconductor device modelling". *Reports on Progress in Physics* **48.2** (1985), p. 223.
- [Vas] Vasileska, D. "Simplified Band-Structure Models and Carrier Dynamics" ().
- [VG08] Vasileska, D. & Goodnick, S. M. "Bulk Monte Carlo Code Described" (2008).
- [Vas11] Vasileska, D. et al. "Monte Carlo Device Simulations". *Applications of Monte Carlo Method in Science and Engineering*. InTech, 2011.
- [Ven89] Venturi, F. et al. "A general purpose device simulator coupling Poisson and Monte Carlo transport with applications to deep submicron MOSFETs". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **8.4** (1989), pp. 360–369.

- [WS62] Wolfe, R & Smith, G. "EFFECTS OF A MAGNETIC FIELD ON THE THERMOELECTRIC PROPERTIES OF A BISMUTH-ANTIMONY ALLOY". *Applied Physics Letters* **1.1** (1962), pp. 5–7.
- [Zia16] Ziabari, A. et al. "Nanoscale solid-state cooling: a review". *Reports on Progress in Physics* **79.9** (2016), p. 095901.

APPENDIX

APPENDIX

A

SIMULATION FILES AND CODE

A.1 Sentaurus TCAD Command File

A.1.1 SDE Command File

```
#set dop @dop@
(sdegeo:set-default-boolean "ABA")

;Define region
(sdegeo:create-cuboid (position 0.0 0.0 0.0) (position 3 1 1) "Silicon" "region_1")

;Define contacts
(sdegeo:define-contact-set "Positive_Contact" 4.0 (color:rgb 1.0 0.0 0.0) "##")
(sdegeo:define-contact-set "Negative_Contact" 4.0 (color:rgb 0.0 1.0 0.0) "||")

(sdegeo:define-3d-contact (find-face-id (position 3.0 0.5 0.5)) "Positive_Contact")
(sdegeo:define-3d-contact (find-face-id (position 0.0 0.5 0.5)) "Negative_Contact")
```

```

;For defining thermal contacts
;(sdegeo:create-cuboid (position 1 0 0) (position 2 1 -0.5) "Metal" "Face")
;(sdegeo:create-cuboid (position 1 0 1) (position 2 1 1.5) "Metal" "Face")

;(sdegeo:delete-region (find-body-id (position 1.5 0.1 -0.025)))
;(sdegeo:delete-region (find-body-id (position 1.5 0.025 1.25)))

;(sdegeo:define-contact-set "top" 4.0 (color:rgb 1.0 0.0 0.0) "##")
;(sdegeo:define-contact-set "bottom" 4.0 (color:rgb 0.0 1.0 0.0) "||")

;(sdegeo:define-3d-contact (find-face-id (position 1.5 0.5 1.0)) "top")
;(sdegeo:define-3d-contact (find-face-id (position 1.5 0.5 0)) "bottom")

;Define doping
(sdedr:define-constant-profile "NDoping" "PhosphorusActiveConcentration" @dop@)
(sdedr:define-constant-profile-region "Placeconst" "NDoping" "region_1")

;Define mesh
(sdedr:define-refeval-window "RefEvalWin_1" "Cuboid" (position 0 0 0) (position 3 1 1))
(sdedr:define-refinement-size "RefinementDefinition_0" 0.1 0.1 0.1 0.1 0.1 0.1 )

(sdedr:define-refinement-placement "RefinementPlacement_0" "RefinementDefinition_0" "RefEvalWin_1" )
(sdedr:define-refinement-function "RefinementDefinition_0" "DopingConcentration" "MaxTrans-Diff" 1)

; Saving BND file
(sdeio:save-tdr-bnd (get-body-list) "@tdrboundary/o@")

; Save CMD file
(sdedr:write-cmd-file "@commands/o@")
(system:command "snmesh -offset n@node@_msh")

```

A.1.2 SDevice Command File

```
Device TMCooler{
```

```
File {
```

```
Grid= "@tdr@"
```

```
Parameters= "@parameter@"
```

```
Plot= "@tdrdat@"
```

```
Current= "@plot@"
```

```
*Output= "@log@"
```

```
*eSHEDistribution = "edist"
```

```
}
```

```
Electrode {
```

```
{ Name="Positive_Contact" Voltage=0 Current=1e-3}
```

```
{ Name="Negative_Contact" Voltage= 0 Current=-1e-3}
```

```
}
```

```
*Thermode {
```

```
*{Name="top" Temperature=300}
```

```
*{Name="bottom" Temperature=300}
```

```
*}
```

```
Physics {
```

```
MagneticField=(0.0 1 0.0)
```

```
*Hydrodynamic(eTemperature)
```

```
*Thermodynamic
```

```
eSHEDistribution
```

```
*IncompleteIonization
```

```
Recombination (
```

```
SRH(DopingDependence TempDep)
```

```
Auger
```

```
)
```

```
Mobility (
```

```
DopingDependence
```

```

HighFieldSaturation
Enormal
)
EffectiveIntrinsicDensity (OldSlotboom NoFermi)
Temperature= 300
}

} * End of Device
File {
Output= "@log@"
}

Plot {
IntrinsicDensity
eDensity hDensity
eCurrent/Vector hCurrent/Vector
eCurrent hCurrent
TotalCurrent/Vector
TotalCurrent
ElectricField
eQuasiFermi hQuasiFermi
egradQuasiFermi hgradQuasiFermi
Potential Doping SpaceCharge
SRH Auger
AvalancheGeneration
eAvalanche hAvalanche
eMobility hMobility
DonorConcentration AcceptorConcentration
Doping
eVelocity hVelocity
eQuasiFermi hQuasiFermi
eLifetime
hLifetime
eTrappedCharge hTrappedCharge
eInterfaceTrappedCharge hInterfaceTrappedCharge

```

```

TotalInterfaceTrapConcentration
ConductionBandEnergy
ValenceBandEnergy
eTemperature hTemperature
Temperature
*eTemperatureRelaxationTime
eSHEDensity # electron density [/cm3]
eSHEEnergy # average electron energy [K]
*eSHEAvalancheGeneration # electron avalanche generation rate [/cm3s]
*eSHECurrentDensity/Vector # electron current density [A/cm2]
*eSHEVelocity/Vector # electron average velocity [cm/s]
eSHEDistribution/SpecialVector
}

```

```

Math {
ExtendedPrecision(256)
Extrapolate
Digits=7
ErrEff(electron)= 1e-12
ErrEff(hole)= 1e-12
RHSmax= 1e30
RHSmin= 1e-30
CdensityMin= 1.e-30
Notdamped= 20
Iterations= 50
RecBoxIntegr
Number_Of_Threads=3
DirectCurrent
SHERefinement=1
}

```

```

Solve {
Poisson
Coupled (IncompleteNewton) { Poisson Electron Hole }
*Quasistationary (

```

```

* Goal { Name= Positive_Contact Value=0.01 }
*){ Coupled { Poisson Electron Hole Temperature}}
}

```

A.2 Monte Carlo MATLAB code

Table A.1 List of variables

Variable name	Definition	Units
af	Nonparabolicity of conduction band	[1/eV]
am	Density of states effective mass	[kg]
am0	Mass of an electron	9.1×10^{-31} [kg]
amc	Conductivity effective mass	[kg]
amd	Density of states effective mass	[kg]
aml	Longitudinal effective mass	[kg]
amt1, amt2	Transverse effective mass	[kg]
by	Magnetic field in y-direction	[T]
de	Minimum energy step for scatt_ table	[eV]
Debye_length	Debye length	[m]
DefPot_gamma_gamma	Deformation potential	[eV]
density	Mass density	$[kg/m^3]$
doping_density	Doping density	$[m^{-3}]$
dt	Iteration time step	[s]
Efermi	Fermi energy	[eV]
Eg	Energy gap	[eV]
emax	Maximum energy in scatt_ table	[eV]
energy	Carrier energy	[eV]
energy0	Average carrier energy without external forces	[eV]
eps_0	Permittivity of free space	[F/m]
eps_high	Permittivity of material at high frequencies	[F/m]
eps_low	Permittivity of material at low frequencies	[F/m]
f	Distribution function	[No units]
freq	Frequency of each scattering	[#]
fx	External electric field in x-direction	[V/m]

fy	External electric field in y-direction	[V/m]
fz	External electric field in z-direction	[V/m]
h	Planck's constant	1.05459×10^{-34} [Js]
kb	Boltzmann's constant	1.38066×10^{-23} [$m^2 kg s^{-2} K^{-1}$]
ke	Electronic thermal conductivity	[W/mK]
L	Dimensions of the sample	[m]
lat_const	Lattice constant	[m]
max_scatt_mech	Number or scattering mechanisms	[#]
n_lev	Number of levels in the scatt_ table	[No units]
nsim	Number of particles simulated	[#]
nslice	Number of spatial elements in the z-direction	[#]
p	nsim \times 7 array containing $k_x, k_y, k_z, \tau, r_x, r_y, r_z$	[Array]
phonon_gamma_gamma	Optical phonon coupling constant	[eV]
q	Electronic charge	[C]
scatt_table	Scattering probabilities at different energies	[No units]
seeb	Seebeck coefficient	[m]
sigma_gamma	Deformation potential for acoustic phonons	[eV]
sound_velocity	Velocity of sound in the material	[m/s]
sume	Total ensemble energy	[eV]
sumekd	Average incremental change in energy	[eV]
tau_max	Maximum scattering rate	[1/s]
Tel	Electron temperature	[K]
tem	Temperature	[K]
tot_time	Total simulation time	[s]
Vt	Thermal Voltage, $k_b T/q$	[V]

A.2.1 Code

1. Main function

```

1 clear all; clc;
2 readin; %parameter file
3 nslice=100; %grid size in z-direction
4 %
5 %%%Start the Monte Carlo Simulation%%%
```

```

6 %
7 %%%%%%%%%% Calculate the scattering table
  %%%%%%%%%%
8
9 [scatt_table, flag_mech, w, tau_max, max_scatt_mech, tot_scatt]=sc_table(
  Debye_length, phonon_gamma_gamma, DefPot_gamma_gamma, am0, Vt,
  doping_density, eps_low, h, q, kb, tem, density, sound_velocity,
  sigma_gamma, am, af, n_lev, de, acoustic_gamma, Coulomb_scattering,
  intervalley_gamma_gamma);
10
11 %%%%%%%%%% Initialize carriers
  %%%%%%%%%%
12 p=zeros(nsim,7);%this array contains kx,ky,kz,tau,rx,ry,rz for nsim
  carriers
13 energy=0;%carrier energy
14 f=0;%distribution function
15 %assign initial distributions
16 [p, energy, n, f]=init(L, Efermi, Eg, n_lev, nsim, h, q, am0, Vt, doping_density
  , de, amd, amd1, amd2, af, tau_max, p, energy, f);
17 time = 0;
18 j = 0;
19 freq=zeros(10,1);%frequency of each type of scattering
20 velocity=0;
21 Ez=zeros(nslice+1,1);%Hall field
22 phonon_energy=0; % energy exchanged with phonons in each iteration
23 %%%%%%%%%% Initial loop until particles attain equilibrium
  %%%%%%%%%%
24 j=0;
25 %%%%%%%%%% Initial loop
  %%%%%%%%%%
26 while(j <=150000)
27     j = j + 1;
28     time=dt j;
29     phonon_energy=zeros(nslice,1);
30     fprintf('iteration= %d \n',j);

```

```

31 [n, f]=fermi(n_lev, nsim, h, q, doping_density, de, amd, af, energy, f);
32 %[v, Ez]=poissonsolver(q, p(:,7), L, nslice, eps_low, doping_density,
    nsim);
33 [freq, p, energy, velocity, phonon_energy]=free_flight_scatter(
    phonon_energy, Ez, eps_low, nslice, by, L, Energy_debye, kb, tem,
    density, sound_velocity, sigma_gamma, af, n_lev, de, fx, fy, fz, aml,
    amt1, amt2, nsim, am0, h, q, dt, am, freq, p, energy, max_scatt_mech,
    scatt_table, flag_mech, w, tau_max, f, velocity);
34 p_e(j,:) = phonon_energy;
35 vx_t(j) = mean((h/amt1) * p(:,1) * sqrt(amt1/am0) ./ transpose(1+2 * af
    energy));
36 vy_t(j) = mean((h/amt2) * p(:,2) * sqrt(amt2/am0) ./ transpose(1+2 * af
    energy));
37 vz_t(j) = mean((h/aml) * p(:,3) * sqrt(aml/am0) ./ transpose(1+2 * af
    energy));
38 energy_t(j) = mean(energy);
39 if (mod(j, 100) == 0)
40     figure(1);
41     subplot(2,3,1);
42     histogram(energy);
43     title('Energy');
44     subplot(2,3,2);
45     hold on;
46     title('vx');
47     scatter(j, vx_t(j), 'r', 'filled');
48     subplot(2,3,3);
49     hold on;
50     title('Energy vs time');
51     scatter(j, mean(energy), 'g', 'filled');
52     hold off;
53     pause(0.000001);
54     subplot(2,3,4);
55     histogram(p(:,5));
56     title('x-position');
57     subplot(2,3,5);

```

```

58         histogram(p(:,6));
59         title('y-position');
60         subplot(2,3,6);
61         histogram(p(:,7));
62         title('z-position');
63     end
64 end
65
66 time = 0;
67 j = 0;
68 by=2;%Set magnetic field
69 [velocity_x, velocity_y, velocity_z, vel_squared, dist, sume, sumekd]=
        init_final(n_lev);
70 count=0;
71 count1=zeros(n_lev,1);
72 count2=zeros(n_lev,1);
73 n=zeros(n_lev,1);
74 phonon_energy=zeros(nslice,1);
75 fz=-mean(vx_t(140000:150000)) by;
76 %
77 % %! Start averaging quantities for converged simulation
78 vx_t_2=zeros(150001,1);
79 vy_t_2=zeros(150001,1);
80 vz_t_2=zeros(150001,1);
81 energy_t_2=zeros(150001,1);
82 d=0;
83 Ez=zeros(nslice+1,1);
84 v=zeros(nslice+1,1);
85 j=0;
86 %[rho, v, Ez]=poissonsolver(q,p(:,7),L,nslice,eps_low,doping_density,
        nsim);
87 while(j <=150000)
88     j=j+1;
89     fprintf('%d \n',j);
90     time=dt j;

```

```

91     [no, f]=fermi(n_lev, nsim, h, q, doping_density, de, amd, af, energy, f);
92     phonon_energy=zeros(nslice, 1);
93     if (mod(j, 1000)==0)
94         fz=-mean(vx_t_2(j-999:j)) by;
95 %         [rho, v, Ez]=poissonsolver(q, p(:, 7), L, nslice, eps_low,
doping_density, nsim);
96     end
97     [freq, p, energy, velocity, phonon_energy]=free_flight_scatter(
        phonon_energy, Ez, eps_low, nslice, by, L, Energy_debye, kb, tem,
        density, sound_velocity, sigma_gamma, af, n_lev, de, fx, fy, fz, aml,
        amt1, amt2, nsim, am0, h, q, dt, am, freq, p, energy, max_scatt_mech,
        scatt_table, flag_mech, w, tau_max, f, velocity);
98     [velocity_x, velocity_y, velocity_z, vel_squared, dist, sume, sumekd,
        Tel, n, count, count1, count2, kxavg, kyavg, kzavg]=final_avg(amt1,
        amt2, aml, hmt1, hmt2, hml, n_lev, nsim, am0, h, q, kb, energy0, tem,
        doping_density, de, amd, af, hhm, energy, p, velocity_x, velocity_y,
        velocity_z, vel_squared, dist, sume, sumekd, n, velocity, count,
        count1, count2, f);
99     p_e_2(j, :)=phonon_energy;
100    vx_t_2(j)=mean((h/amt1) p(:, 1) sqrt(amt1/am0) ./ transpose(1+2 af
        energy));
101    vy_t_2(j)=mean((h/amt2) p(:, 2) sqrt(amt2/am0) ./ transpose(1+2 af
        energy));
102    vz_t_2(j)=mean((h/aml) p(:, 3) sqrt(aml/am0) ./ transpose(1+2 af
        energy));
103    energy_t_2(j)=mean(energy);
104    if (mod(j, 10)==0)
105        figure(3);
106        subplot(2, 3, 1);
107        hold on;
108        title('vx');
109        scatter(j, vx_t_2(j), 'r', 'filled');
110        subplot(2, 3, 2);
111        hold on;
112        title('vz');

```

```

113         scatter(j,vz_t_2(j),'b','filled');
114         subplot(3,3,3);
115         hold on;
116         title('energy');
117         scatter(j,energy_t_2(j),'g','filled');
118         subplot(2,3,4);
119         title('rz');
120         histogram(p(:,7));
121         subplot(2,3,5);
122         title('volt');
123         plot(v);
124         subplot(2,3,6);
125         title('Ez');
126         plot(Ez);
127         pause(0.000001);
128     end
129 end
130 velocity_x=velocity_x/count;
131 for i=1:n_lev
132     if (count1(i)==0)
133         vel_squared(i)=0;
134     else
135         vel_squared(i)=vel_squared(i)/count1(i);
136     end
137     if (count2(i)==0)
138         dist(i)=0;
139     else
140         dist(i)=dist(i)/count2(i);
141     end
142 end
143 fprintf('%d %d %d \n',count,count1(2),count2(2));
144 [s,seeb,ke]=transport(n_lev,h,q,kb,Efermi,tem,de,amd,af,tot_scat,
    velocity_x,velocity_y,velocity_z,vel_squared,dist,Tel);
145 write_final(Efermi,n_lev,tem,de,velocity_x,velocity_y,velocity_z,
    sume,sumekd,Tel,dist,n,s,seeb,freq,ke,vel_squared);

```

2. Function to compute the scattering table

```
1 %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %
3 % SUBROUTINE THAT CREATES THE SCATTERING TABLE
4 % flag_mech = 1 ==> Optical phonon
5 % flag_mech = 2 ==> acoustic phonons
6 % flag_mech = 3 ==> Coulomb scattering
7 %
8 %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9 function [ scatt_table , flag_mech , w, tau_max , max_scatt_mech , tot_scatt ] =
    sc_table ( Debye_length , phonon_gamma_gamma , DefPot_gamma_gamma , am0 ,
    Vt , doping_density , eps_low , h , q , kb , tem , density , sound_velocity ,
    sigma_gamma , am , af , n_lev , de , acoustic_gamma , Coulomb_scattering ,
    intervalley_gamma_gamma )
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% CREATE TABLE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11 scatt_table = zeros ( n_lev , 6 ) ;
12 i_count = 0 ;
13
14 % Acoustic phonons scattering rate
15 if ( acoustic_gamma == 1 )
16     out_file_1 = 'acoustic_ab.txt' ;
17     out_file_2 = 'acoustic_em.txt' ;
18     [ i_count , scatt_table , flag_mech , w ] = acoustic_rate ( n_lev , h , q , kb ,
        tem , density , sound_velocity , de , sigma_gamma , am , af , i_count ,
        out_file_1 , out_file_2 , scatt_table ) ;
19 end
20
21 % Coulomb scattering rate – Brooks–Herring approach
22 if ( Coulomb_scattering == 1 )
```

```

23     out_file_1 = 'Coulomb.txt';
24     [i_count, scatt_table, flag_mech, w] = Coulomb_BH(n_lev, am0, h, q,
        doping_density, eps_low, de, am, af, Debye_length, i_count,
        out_file_1, scatt_table, flag_mech, w);
25 end
26
27
28 % optical phonons scattering rate
29 if(intervalley_gamma_gamma == 1)
30     out_file_1 = 'optical_ab.txt';
31     out_file_2 = 'optical_em.txt';
32     [i_count, scatt_table, flag_mech, w] = optical_rate(n_lev, h, q, Vt,
        density, de, DefPot_gamma_gamma, phonon_gamma_gamma, am, af,
        i_count, out_file_1, out_file_2, scatt_table, flag_mech, w);
33 end
34
35 x=zeros(n_lev, 2);
36 max_scatt_mech=i_count;
37 for i = 1:n_lev
38     ee = i de;
39     tot_scat(i)=scatt_table(i,1)+scatt_table(i,2)+scatt_table(i,3)+
        scatt_table(i,4)+scatt_table(i,5);
40     x= [ee tot_scat(i)];
41 end
42 dlmwrite('total_scat.txt', x);
43 %normalize scattering table
44 [scatt_table, tau_max]=renormalize_table(n_lev, max_scatt_mech,
        scatt_table);
45 fprintf('Mechanisms in the gamma valley = %d \n', i_count);
46 if(i_count > 0)
47     i = transpose(1:n_lev);
48     ee = i de;
49     dlmwrite('gamma_table_renormalized.txt', [ee scatt_table(:, :)]);
50 end
51 end

```

3. Function to compute acoustic phonon absorption and emission scattering rates

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % Subroutine for the calculation of acoustic phonons
3 % scattering rate
4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 function [i_count , scatt_table , flag_mech , w] = acoustic_rate (n_lev , h , q
    , kb , tem , density , sound_velocity , de , sigma_gamma , am , af , i_count ,
    out_file_1 , out_file_2 , scatt_table )
6 % Calculate constant
7
8 const = ((sigma_gamma q)^2) ((kb tem)^3) sqrt(am)/(2^(5/2) pi h^4
    density sound_velocity^4);
9 es=am ( sound_velocity^2)/2;
10 if ((4 af es/q)>= 1)
11     fprintf('4alphae greater than 1 %d %d %d' , 4 af es/q , af , es/q);
12     exit(code);
13 end
14
15 limit=es/(1-(4 af/q es));
16 c=4 sqrt(es)/(kb tem (1-(4 af/q es)));
17
18 % (a) Scattering rate – absorption
19
20 i_count = i_count + 1;
21 x=zeros(n_lev,2);
22 for i = 1: n_lev
23     ee = de i;
24     ge = ee (1+ af ee) q;
25     fe=(1+2 af ee);
26     if (ge <= limit)
27         x1=c ( sqrt(es) fe-sqrt(ge));
28         x2=c ( sqrt(es) fe+sqrt(ge));
29         f11=f1(x1);

```

```

30         f12=f1(x2);
31         f21=f2(x1);
32         f22=f2(x2);
33     else
34         x1=0;
35         x2=c ( sqrt(es) fe+sqrt(ge));
36         f11=f1(x1);
37         f12=f1(x2);
38         f21=f2(x1);
39         f22=f2(x2);
40     end
41     acoustic = const/sqrt(ge) ( fe ( f12-f11)+(2 af/q kb tem) ( f22-f21
42         ));
43     scatt_table(i,i_count) = acoustic;
44     x(i,:)=[ee acoustic];
45 end
46 dlmwrite(out_file_1,x);
47 flag_mech(i_count) = 2;
48 w(i_count) = 1;
49
50 %      (b) Scattering rate – emission
51
52 i_count = i_count + 1;
53 x=zeros(n_lev,2);
54 for i = 1: n_lev
55     ee = de i;
56     ge = ee (1+af ee) q;
57     fe=(1+2 af ee);
58     if (ge <= limit)
59         acoustic=0;
60     else
61         x1=0;
62         x2=c ( sqrt(ge)-sqrt(es) fe);
63         f11=g1(x1);

```

```

64         f12=g1(x2);
65         f21=g2(x1);
66         f22=g2(x2);
67         acoustic = const/sqrt(ge) ( fe ( f12-f11) -(2 af/q kb tem) ( f22
        -f21));
68     end
69     scatt_table(i,i_count) = acoustic;
70     x(i,:)=[ee acoustic];
71 end
72 dlmwrite(out_file_2 ,x);
73 flag_mech(i_count) = 2;
74 w(i_count) = -1;
75 end
76
77 %! f1
78 function out=f1(x)
79 xa=3.5;
80 if (x <= xa)
81     out=(x^2/2)-(x^3/6)+(x^4/48)-(x^6/4320)+(x^8/241920)-(x
        ^10/12096000)+(x^12/622702080);
82 else
83     out=(xa^2/2)-(xa^3/6)+(xa^4/48)-(xa^6/4320)+(xa^8/241920)-(xa
        ^10/12096000)+(xa^12/622702080)+exp(-xa) ( xa^2+2 xa+2)-exp(-x
        ) ( x^2+2 x+2);
84 end
85 end
86
87 %! f2
88 function out=f2(x)
89 xa=3.5;
90 if (x <= xa)
91     out=(x^3/3)-(x^4/8)+(x^5/60)-(x^7/5040)+(x^9/272160)-(x
        ^11/143305600)+(x^13/622702080);
92 else
93     out=(xa^3/3)-(xa^4/8)+(xa^5/60)-(xa^7/5040)+(xa^9/272160)-(xa

```

```

          ^11/143305600)+(xa^13/622702080)+exp(-xa) ( xa^3+3 xa^2+6 xa
          +6)-exp(-x) ( x^3+3 x^2+6 x+6);
94 end
95 end
96
97 %! g1
98 function out=g1(x)
99 xa=4;
100 if (x <= xa)
101     out=(x^2/2)+(x^3/6)+(x^6/4320)+(x^8/241920)-(x^10/12096000)+(x
          ^12/622702080);
102 else
103     out=(xa^2/2)+(xa^3/6)+(xa^6/4320)+(xa^8/241920)-(xa^10/12096000)
          +(xa^12/622702080)+exp(-xa) ( xa^2+2 xa+2)-xa^3/3-exp(-x) ( x
          ^2+2 x+2)+x^3/3;
104 end
105 end
106
107 %! g2
108 function out=g2(x)
109 xa=4;
110 if (x <= xa)
111     out=(x^3/3)+(x^4/8)+(x^5/60)-(x^7/5040)+(x^9/272160)-(x
          ^11/143305600)+(x^13/622702080);
112 else
113     out=(xa^3/3)+(xa^4/8)+(xa^5/60)-(xa^7/5040)+(xa^9/272160)-(xa
          ^11/143305600)+(xa^13/622702080)+exp(-xa) ( xa^3+3 xa^2+6 xa
          +6)-xa^4/4-exp(-x) ( x^3+3 x^2+6 x+6)+x^4/4;
114 end
115 end
116
117
118 %%%%%%%%%% END %%%%%%%%%%

```

4. Function to compute optical phonon absorption and emission scattering rates

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % Generic subroutine for the calculation of
3 % INTERVALLEY PHONONS scattering rate
4 % (absorption + emission)
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 function [i_count , scatt_table , flag_mech ,w] = optical_rate (n_lev ,h ,q ,
    Vt , density , de ,DefPot_gamma_gamma ,phonon_gamma_gamma ,am , af , i_count
    , out_file_1 , out_file_2 , scatt_table , flag_mech ,w)
7 % Calculate constants
8 w0=phonon_gamma_gamma; %coupling constant
9 rnq = 1/(exp(w0/Vt)-1);%phonon DOS
10 const = (DefPot_gamma_gamma^2) ((am q)^(3/2))/(sqrt(2) pi h h
    density w0);
11
12 % (a) Scattering rate – absorption
13 i_count = i_count + 1;
14 ab = rnq const; %absorption
15 x=zeros(n_lev ,2);
16 for i = 1: n_lev
17     ee = de i;
18     ef = ee + w0; %new energy value
19     gf = ef (1+af ef);% gamma function = E(1+aE)
20     if (ef <= 0)
21         absorption = 0;
22     else
23         factor = sqrt(gf) (1+2 af ef);
24         absorption = ab factor;
25     end
26     scatt_table(i ,i_count) = absorption;
27     x(i ,:)= [ee absorption];
28 end
29 dlmwrite(out_file_1 ,x);
30 flag_mech(i_count) = 1;

```

```

31 w(i_count) = w0;
32
33 % (b) Scattering rate – emission
34 i_count = i_count + 1;
35 em = (1+rnq) const; %emission
36 x=zeros(n_lev,2);
37
38 for i = 1: n_lev
39     ee = de i;
40     ef = ee - w0;%new energy
41     gf = ef (1+af ef);% gamma function
42     if(ef <= 0)
43         emission = 0;
44     else
45         factor = sqrt(gf) (1+2 af ef);
46         emission = em factor;
47     end
48     scatt_table(i,i_count) = emission;
49     x(i,:)=[ee emission];
50 end
51 dlmwrite(out_file_2 ,x);
52 flag_mech(i_count) = 1;
53 w(i_count) = - w0;
54 end

```

5. Function to calculate ionized impurity scattering rate

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % Subroutine for the calculation of COULOMB SCATTERING rate
3 % Assumption ==> elastic scattering process
4 % (Brooks–Herring approach)
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 function [i_count , scatt_table , flag_mech ,w]=Coulomb_BH(n_lev ,am0,h,q,
    doping_density ,eps_low ,de,am,af ,Debye_length ,i_count , out_file_1 ,
    scatt_table , flag_mech ,w)

```

```

7
8 % Calculate constants
9
10 fprintf('Debye Length= %d',Debye_length);
11 z=2; %charge on impurity
12 const = doping_density z z (q^4)/(16 sqrt(2 am) pi ((eps_low)^2));
13 a=(2 pi doping_density)^(-1/3);
14
15 % Calculate scattering rate:
16 i_count = i_count + 1;
17 x=zeros(n_lev,2);
18 for i = 1: n_lev
19     ee = de i;
20     ge = ee (1+af ee); %energy considering non- parabolicity
21     k=sqrt(2 am0 q)/h sqrt(ee (1+af ee)) sqrt(am/am0); % wave
        vector calculation by dispersion relation
22     vg=h k/(am (1+2 af ee)); % group velocity
23     factor=4 k k (Debye_length^2);
24     zeta=log(1+factor)-(factor/(1+factor));
25     scatt_rate=const zeta (1+2 af ee)/((ge q)^(3/2));
26     scattr=vg/a (1-exp(-a scatt_rate/vg));
27     scatt_table(i,i_count) = scattr;
28     x(i,:)=[ee 1/scattr];
29 end
30 dlmwrite(out_file_1 ,x);
31
32 flag_mech(i_count) = 3;
33 w(i_count) = 0;
34 end

```

6. Function to normalize scattering table

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % Generic subroutine that renormalizes the scattering table
3 % for a given valley

```

```

4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 function [scatt_table,tau_max]= renormalize_table(n_lev,
6         max_scatt_mech,scatt_table)
7     if(max_scatt_mech >= 1)
8         if(max_scatt_mech > 1)
9             for i = 2: max_scatt_mech
10                for k = 1: n_lev
11                    scatt_table(k,i) = scatt_table(k,i-1) + scatt_table(
12                        k,i);
13                end
14            end
15        end
16        i_max = max_scatt_mech;
17        tau = 0;
18        for i = 1:n_lev
19            if(scatt_table(i,i_max) > tau)
20                tau = scatt_table(i,i_max);
21            end
22        end
23        for i = 1: max_scatt_mech
24            for k = 1: n_lev
25                scatt_table(k,i) = scatt_table(k,i)/tau;
26            end
27        end
28        tau_max = 1/tau;
29        fprintf('tau_max =%f \n',tau_max);
30    end
31 end

```

7. Function to initialize energy, momentum and position of carriers

```

1 %
2 % C

```

```

3 % C      INITIALIZE CARRIER ENERGY AND WAVEVECTOR ACCORDING TO THE
4 % C      MAXWELL-BOLTZMANN STATISTICS
5 % C
6 %
7 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7 function [p,energy , n, f]=init (L, Efermi , Eg, n_lev , nsim , h, q, am0, Vt ,
      doping_density , de , amd, amd1, amd2, af , tau_max, p, energy , f)
8 sume = 0;
9 nol=zeros(n_lev ,1);%electron density
10 no2=zeros(n_lev ,1);%hole density
11 const=sqrt(2) amd1^(3/2)/pi^2/h^3 q;%Conduction Band
12 const1=sqrt(2) amd2^(3/2)/pi^2/h^3 q;%Valence Band
13 %electron dist calculation from DOS
14 for i=1:n_lev
15     ee=i de;
16     g=sqrt(ee q (1+ af ee)) (1+2 af ee) const de;
17     dist=1/(exp((ee-Efermi)/Vt)+1);
18     nol(i)=(g dist);
19 end
20 %hole dist calculation from DOS
21 for i=1:n_lev
22     ee=i de;
23     g=sqrt(ee q (1+ af ee)) (1+2 af ee) const1 de;
24     dist=1/(exp((ee+Eg+Efermi)/Vt)+1);
25     no2(i)=(g dist);
26 end
27 tot=0;
28 %total #electrons calculation
29 for i=1:n_lev
30     tot=tot+nol(i);
31     Ffreq(i)=tot/sum(nol); % fraction of carriers <= energy i de
32     x(i,:)=[i de Ffreq(i) nol(i)];
33 end
34 fprintf('Sum= %d Doping= %d' ,sum(nol)-sum(no2) , doping_density);

```

```

35 for i = 1: nsim
36     rr= rand(1,1);
37     e = -(1.5 Vt) log(rr);
38     sume = sume + e;
39     %!      Initial real space position
40     rx=L rand(1,1);
41     ry=L rand(1,1);
42     rz=L rand(1,1);
43     %!      Initial wavevector
44     k=sqrt(2 am0 q)/h sqrt(e (1+ af e));
45     rr=rand(1,1);
46     fai=2 pi rr;
47     rr=rand(1,1);
48     ct=1-2 rr;
49     st=sqrt(1-ct ct);
50     kz=k st cos(fai);
51     ky=k st sin(fai);
52     kx=k ct;
53     while ((kx<0) || (st==0)) % only keep kx>0
54         rr=rand(1,1);
55         ct=1-2 rr;
56         st=sqrt(1d0-ct ct);
57         kz=k st cos(fai);
58         ky=k st sin(fai);
59         kx=k ct; % Injection in x-direction
60     end
61     %!      Initial free-flight
62     rr=rand(1,1);
63     while(rr<=1e-5)
64         rr=rand(1,1);
65     end
66     tc = -(log(rr)) tau_max; %initial free flight time
67     %!      Map particle attributes
68     p(i,:)=[kx ky kz tc rx ry rz];
69     energy(i) = e;

```

```

70 end
71 %Find initial dist and write to file
72 [n, f]=fermi(n_lev, nsim, h, q, doping_density, de, amd, af, energy, f);
73 x=zeros(n_lev, 2);
74 for i=1:n_lev
75     ee=i de;
76     x(i, :)= [ee f(i)];
77 end
78 dlmwrite('initial f.txt', x);
79 sume = sume/nsim;
80 fprintf('Average carrier energy - initial distribution \nEnergy = %f
        \n', sume);
81 end

```

8. Function to compute energy distribution from carrier energies

```

1 % !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
2 % ! Calculate fermi distribution
3 function [n, f]=fermi(n_lev, nsim, h, q, doping_density, de, amd, af, energy,
    f)
4 const=sqrt(2) (amd^(3/2))/pi^2/h^3 q;
5 n=zeros(n_lev, 1);
6
7 for i=1:nsim
8     loc=ceil(energy(i)/de);
9     if(loc==0)
10        loc=1;
11    end
12    if(loc>n_lev)
13        loc=n_lev;
14    end
15    n(loc)=n(loc)+1;
16 end
17 for i=1:n_lev
18    ee = i de;

```



```

21     ry = p(i,6);
22     rz = p(i,7);
23     e = energy(i);
24     %Index to the spatial grid – used for tracking phonon
        energy exchanges
25     %in each slice
26     ns=ceil(rz nslice/L);
27     if (ns ==0)
28         ns=1;
29     end
30     if (ns > nslice)
31         ns=nslice;
32     end
33     %C     Initial free–flight of the carriers
34     dte = dtau;
35     if(dte >= dt)
36         dt2=dt;
37     else
38         dt2=dte;
39     end
40
41     %free– flight
42     [ kx , ky , kz , e , rx , ry , rz , vdrift ]= kdrift (eps_low , Ez , nslice , by , L
        , fx , fy , fz , aml , amt1 , amt2 , am0 , h , q , am , af , dt2 , kx , ky , kz , e , rx
        , ry , rz , vdrift );
43     velocity=velocity+vdrift;
44     time=time+dt2;
45     while(dte < dt)
46         dte2=dte;
47         % scatter
48         [ freq , kx , ky , kz , e , deltaq ]= scatter_carrier (h,
            Energy_debye , am0 , q , kb , tem , density , sound_velocity ,
            sigma_gamma , am , af , n_lev , de , max_scatt_mech ,
            scatt_table , flag_mech , w , freq , kx , ky , kz , e , f );
49     phonon_energy (ns)=phonon_energy (ns)+deltaq;

```



```

30 dkx1 = -q/h tau ( fx) sqrt(am0/amt1);%due to E
31 dkx2 = q/h tau ( velz by) sqrt(am0/amt1);% due to B
32 dky = -q/h tau fy sqrt(am0/amt2);
33 dkz1 = -q/h tau fz sqrt(am0/aml);%due to E
34 dkz2 = -q/h tau ( velx by) sqrt(am0/aml); % due to B
35
36 %changes in rx,ry,rz using equation of motion x=ut + 1/2at^2
37 %ut=velocity term
38 %1/2at^2= acceleration term
39 drx1 = -q/amt1/2 (fx-velz by) tau^2; % acceleration term
40 drx2 = h kx tau sqrt(amt1/am0)/amt1/(1+2 af e); %velocity term
41 dry1 = -q/amt2/2 fy tau^2; % acceleration term
42 dry2 = h ky tau sqrt(amt2/am0)/amt2/(1+2 af e); %velocity term
43 drz1 = -q/aml/2 (fz+velx by) tau^2; % acceleration term
44 drz2 = h kz tau sqrt(aml/am0)/aml/(1+2 af e); %velocity term
45
46 kx = kx+dkx1+dkx2;
47 ky = ky+dky;
48 kz = kz+dkz1+dkz2;
49
50 rx = rx+drx1+drx2;
51 ry = ry+dry1+dry2;
52 rz = rz+drz1+drz2;
53
54 %Boundary Conditions
55 if (rx >= L && kx >0)
56     rx=rx-L; %periodic BC
57 end
58 if (rx<=0 && kx<0)
59     rx=rx+L; %periodic BC
60 end
61 if (ry<=0 && ky<=0)
62     ry=-ry;
63     ky=-ky;%reflective BC
64 end

```

```

65 if (ry>=L && ky>=0)
66     ry=2 L-ry;
67     ky=-ky;%reflective BC
68 end
69 if (rz<=0 && kz<=0)
70     rz=-rz;
71     kz=-kz;%reflective BC
72 end
73 if (rz<=0 && kz>=0)
74     rz=0;
75 end
76 if (rz>=L && kz>0)
77     rz=2 L-rz;
78     kz=-kz; %reflective BC
79 end
80 if (rz>=L && kz<0)
81     rz=L;
82 end
83 sk = kx^2+ky^2+kz^2;
84 k = sqrt(sk);
85 gk=h^2/am0/q/2 sk; % /q for converting to eV
86 e1=e;
87 e = 2 gk/(1+sqrt(1+4 af gk));
88 vdrift=-(e-e1)/(fx);
89 end

```

11. Function where carrier undergoes scattering

```

1 %
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %
3 %     SELECT SCATTERING MECHANISM AND PERFORM
4 %     THE SCATTERING PART THAT MODIFIES PARTICLE ATRIBUTES
5 %     (kx, ky, kz, iv, energy)

```

```

6 %
7 %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8 function [freq ,kx ,ky ,kz ,e ,deltaq]=scatter_carrier (h ,Energy_debye ,am0
   ,q ,kb ,tem ,density , sound_velocity ,sigma_gamma ,am , af , n_lev , de ,
   max_scatt_mech , scatt_table , flag_mech ,w ,freq , kx , ky , kz , e , f)
9
10 % Calculate index to the scattering table
11
12     loc = ceil(e/de); %loc=index to sc table
13     if (loc < 0)
14         fprintf('loc= %d e=%d' ,loc ,e);
15     end
16     if(loc == 0)
17         loc=1;
18     end
19     if(loc > n_lev)
20         loc=n_lev;
21     end
22
23 % Select scattering mechanism
24 % i_fix= 1(Ac Ab) , 2(Ac Em) , 3(Ionized Imp) , 4(Op Ab) , 5(Op Em) , 6(
   Self)
25 % freq(6,1) => counter for each type of scattering
26 % flag_mech = 1 (op ab, op em) , 2 (Ac Ab, Ac Em) , 3 (Ionized Imp)
27
28
29     i_top = max_scatt_mech;% max number of scattering phenomena
30     rr = rand(1,1);
31     if(rr >= scatt_table(loc , i_top))
32         freq(i_top+1)=freq(i_top+1)+1; %Self scattering
33         deltaq=0;
34     else
35         if(rr < scatt_table(loc ,1))

```

```

36         i_fix = 1;
37         freq(i_fix)=freq(i_fix)+1;
38     else
39         if(i_top > 1)
40             for i=1:i_top-1
41                 bound_lower = scatt_table(loc,i);
42                 bound_upper = scatt_table(loc,i+1);
43                 if (rr >= bound_lower && rr < bound_upper)
44                     i_fix = i + 1;
45                     freq(i_fix)=freq(i_fix)+1;
46                     break;
47                 end
48             end
49         end
50
51     end
52     select_mech = flag_mech(i_fix);
53     if(select_mech == 1)
54         [e,kx,ky,kz,deltaq]=optical_phonon(n_lev,am0,q,h,de,af,
55             ,i_fix,e,w,kx,ky,kz,f);
56     elseif(select_mech==2)
57         [e,kx,ky,kz,deltaq]=acoustic_phonon(kb,tem,density,
58             sigma_gamma,n_lev,am0,h,q,sound_velocity,de,am,af,
59             i_fix,e,w,kx,ky,kz,f);
60     elseif(select_mech==3)
61         [e,kx,ky,kz,deltaq]=Coulomb_angle_BH(af,Energy_debye,e
62             ,kx,ky,kz);
63     end
64 end
65 end
66 %
67 %
68 %
69 %
70 %
71 %
72 %
73 %
74 %
75 %
76 %
77 %
78 %
79 %
80 %
81 %
82 %
83 %
84 %
85 %
86 %
87 %
88 %
89 %
90 %
91 %
92 %
93 %
94 %
95 %
96 %
97 %
98 %
99 %

```

```

66 % C
67 % C      SCATTERING SUBROUTINES
68 % C      (CHANGE ENERGY AND WAVEVECTORS OF PARTICLES)
69 % C
70 % C      In the definition of the scattering rates , a variable
      called
71 % C      'flag_mech' has been defined.  The values assigned to this
72 % C      variable correspond to:
73 % C
74 % C      1 ==> Isotropic scattering (optical phonon, intervalley)
75 % C      2 ==> Polar_optical ==> (acoustic phonon) anisotropic
      scattering (small angle)
76 % C      3 ==> Coulomb scattering ==> small-angle scattering
77 % C
78 %
      %%%%%%%%%%%
79 %
80 %
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
81 % C      ISOTROPIC SCATTERING PROCESS
82 % C      uniform probability density for scattering in all
      directions
83 %
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
84 function [e , kx , ky , kz , deltaq]=optical_phonon ( n_lev , am0 , q , h , de , af ,
      i_fix , e , w , kx , ky , kz , f )
85
86 %C      Update carrier energy
87 enew = e + w(i_fix);
88 %! Check if final electron state is occupied
89 rr=rand(1,1);

```



```

150         deltaq=0;
151         return
152     end
153     qph=acph(h, q, kb, tem, density, sound_velocity, sigma_gamma, am, af, e,
            qmin, qmax, w(i_fix));
154     enew = e - h qph sound_velocity/q;
155 end
156 if (enew<0)
157     fprintf('enew=%d', e);
158 end
159 rr= rand(1,1);
160 loc=round(enew/de);
161 if(loc == 0)
162     loc=1;
163 end
164 if(loc > n_lev)
165     loc=n_lev;
166 end
167 if (rr<=f(loc))
168     deltaq=0;
169     return
170 end
171 qph=qph sqrt(am0/am);
172
173 %C    Calculate the rotation angles
174 cth0 = kz/k;
175 sth0 = kxy/k;
176 cfi0 = kx/kxy;
177 sfi0 = ky/kxy;
178
179 %C    Randomize momentum in the rotated coordinate system
180 kp = sqrt(2 am0 q)/h sqrt(enew (1+ af enew));
181 gnew = enew (1+ af enew);
182 cth = (kp^2+k^2-qph^2)/(2 kp k);
183 if ((cth>1) || (cth<-1))

```

```

184     fprintf('costheta greater than 1 %d',cth);
185     deltaq=0;
186     return
187 end
188 sth = sqrt(1-cth cth);
189 rr=rand(1,1);
190 %rr=0.6;
191 fi = 2 pi rr;
192 cfi = cos(fi);
193 sfi = sin(fi);
194 kxp = kp sth cfi;
195 kyp = kp sth sfi;
196 kzp = kp cth;
197 %C     Return back to the original coordinate system
198 kx = kxp cfi0 cth0-kyp sfi0+kzp cfi0 sth0;
199 ky = kxp sfi0 cth0+kyp cfi0+kzp sfi0 sth0;
200 kz = -kxp sth0+kzp cth0;
201 while ((kx==0)&&(ky==0))
202     rr=rand(1,1);
203     fi = 2 pi rr;
204     cfi = cos(fi);
205     sfi = sin(fi);
206     kxp = kp sth cfi;
207     kyp = kp sth sfi;
208     kzp = kp cth;
209 %C     Return back to the original coordinate system
210 kx = kxp cfi0 cth0-kyp sfi0+kzp cfi0 sth0;
211 ky = kxp sfi0 cth0+kyp cfi0+kzp sfi0 sth0;
212 kz = -kxp sth0+kzp cth0;
213 end
214 deltaq=e-ew;
215 e = ew;
216 %fprintf('%d %d %d %d',kx,ky,kz,e);
217 end
218

```



```

249 kxp = k sth cfi;
250 kyp = k sth sfi;
251 kzp = k cth;
252
253 %!      Return back to the original coordinate system
254 kx = kxp cfi0 cth0-kyp sfi0+kzp cfi0 sth0;
255 ky = kxp sfi0 cth0+kyp cfi0+kzp sfi0 sth0;
256 kz = -kxp sth0+kzp cth0;
257 while ((kx==0)&&(ky==0))
258     rr= rand(1,1);
259     cth = 1-2 rr/(1+4(1-rr) ge/Energy_debye);
260     while ((cth>1) || (cth<-1))
261         rr= rand(1,1);
262         cth = 1-2 rr/(1+4(1-rr) ge/Energy_debye);
263     end
264     sth = sqrt(1-cth cth);
265     rr= rand(1,1);
266     fi = 2 pi rr;
267     cfi = cos(fi);
268     sfi = sin(fi);
269     kxp = k sth cfi;
270     kyp = k sth sfi;
271     kzp = k cth;
272 %!      Return back to the original coordinate system
273 kx = kxp cfi0 cth0-kyp sfi0+kzp cfi0 sth0;
274 ky = kxp sfi0 cth0+kyp cfi0+kzp sfi0 sth0;
275 kz = -kxp sth0+kzp cth0;
276 end
277 deltaq=0;
278 end
279
280 function qph= acph(h,q, kb, tem, density, sound_velocity, sigma_gamma, am,
    af, e, qmin, qmax, w)
281 c_l = density sound_velocity^4;
282 const = (sigma_gamma q)^2 (kb tem)^3 sqrt(am)/(2^(5/2) pi h^4 c_l)/

```

```

    sqrt(e (1+ af e) q);
283 a=h qmin sound_velocity/(kb tem);
284 b=h qmax sound_velocity/(kb tem);
285 dq=(qmax-qmin)/50;
286 %prob dist for choosing a wavelength between qmax and qmin
287 for i=1:50
288     qph=qmin+i dq;
289     x=h qph sound_velocity/(kb tem);
290     if (x==0)
291         p(i)=0;
292     else
293         p(i)=const x x (1+2 af e+(2 af/q x kb tem w));
294         if (w==1)
295             p(i)=p(i)/(exp(x)-1);
296         else
297             p(i)=p(i) ((1/(exp(x)-1))+1);
298         end
299     end
300 end
301 c=1.5 max(p);
302 r1=rand(1,1);
303 while (r1==0)
304     r1=rand(1,1);
305 end
306 r2=rand(1,1);
307 x=a+(b-a) r1;
308 f1=r2 c;
309 f=const x x (1+2 af e+(2 af/q x kb tem w));
310 if (w==1)
311     f=f/(exp(x)-1);
312 else
313     f=f (1/(exp(x)-1)+1);
314 end
315 while (f1 > f)
316     r1=rand(1,1);

```

```

317     while (r1==0)
318         r1=rand(1,1);
319     end
320     r2=rand(1,1);
321     x=a+(b-a) * r1;
322     f1=r2 * c;
323     f=const * x * x * (1+2 * af * e +(2 * af/q * x * kb * tem * w));
324     if (w==1)
325         f=f/(exp(x)-1);
326     else
327         f=f * (1/(exp(x)-1)+1);
328     end
329 end
330 qph=kb * tem * x/(h * sound_velocity);
331 end

```

12. Function that solves Poisson's equation in z-direction

```

1  function [rho , v , Ez]=poissonsolver(q , z , L , nslice , eps_low ,
2      doping_density , nsim)
3  dz=L/nslice ;
4  loc=ceil (z/dz-0.5)+1;
5  for i=1:nsim ,
6      if (loc(i)<=0)
7          loc(i)=1;
8      end
9      if (loc(i)>=nslice+1)
10         loc(i)=nslice+1;
11     end
12 end
13 rho=zeros (nslice+1,1);%initialize
14 for i=1:nslice+1
15     rho(i)=q * ( doping_density-(sum(loc==i) * (L/dz) * doping_density/nsim
16         ));

```

```

16     if i==1 || i==nslice+1
17         rho(i)=q ( doping_density-(sum(loc==i) (2 L/dz)
18             doping_density/nsim));
19     end
20
21     rho=smooth(rho,5);
22
23     %constructing matrix
24     d=-2*ones(nslice+1,1);
25     d(1)=1;d(nslice+1)=1;
26     du=ones(nslice,1);
27     du(1)=0;
28     dl=ones(nslice,1);
29     dl(nslice)=0;
30     A=diag(d)+diag(du,1)+diag(dl,-1);
31     %for Neumann BC
32     A(nslice+1,nslice)=-1;
33     %poisson equation and boundary conditions
34     w=-dz/dz rho(1:nslice)/eps_low;
35     w(1)=0;
36     w(nslice+1)=0;
37     %inverting matrix
38     %v=w/A;
39     v=inv(A) w;
40     %calculating electric field
41     v=smooth(v);%potential gradient
42     Ez=-gradient(v)/dz;%Hall-field
43     end

```

13. Function to initialize final average quantities

```

1     function [ velocity_x , velocity_y , velocity_z , vel_squared , dist , sume ,
2             sumekd]=init_final(n_lev)
3     dist=zeros(n_lev,1);

```

```

3 vel_squared=zeros(n_lev,1);
4 velocity_x=0;
5 velocity_y=0;
6 velocity_z=0;
7 sume=0;
8 sumekd=0;
9 end

```

14. Function to compute final averages

```

1 function [velocity_x, velocity_y, velocity_z, vel_squared, dist, sume,
sumekd, Tel, n, count, count1, count2, kxavg, kyavg, kzavg]=final_avg(
amt1, amt2, aml, hmt1, hmt2, hml, n_lev, nsim, am0, h, q, kb, energy0, tem,
doping_density, de, amd, af, hhm, energy, p, velocity_x, velocity_y,
velocity_z, vel_squared, dist, sume, sumekd, n, velocity, count, count1,
count2, f)
2
3 velx_sum = 0;
4 vely_sum = 0;
5 velz_sum = 0;
6 sumel = 0;
7 sumekd1=0;
8 kxavg=0;
9 kyavg=0;
10 kzavg=0;
11 for i = 1:nsim
12     ee = energy(i);
13     denom = 1/(1+2 af ee);
14     velx = hmt1 p(i,1) denom sqrt(amt1/am0);
15     vely = hmt2 p(i,2) denom sqrt(amt2/am0);
16     %velz = hmt1 p(i,1) denom sqrt(amt1/am0);
17     velz = hml p(i,3) denom sqrt(aml/am0);
18     kxavg=kxavg+p(i,1);
19     kyavg=kyavg+p(i,2);
20     kzavg=kzavg+p(i,3);

```

```

21     velx_sum = velx_sum + velx;
22     vely_sum = vely_sum + vely;
23     velz_sum = velz_sum + velz;
24     sumel = sumel + energy(i);
25     %!           sumekdl=sumekdl+energykd(i)
26 end
27 kxavg=kxavg/nsim;
28 kyavg=kyavg/nsim;
29 kzavg=kzavg/nsim;
30 %fprintf('kavg %d %d %d \n',kxavg,kyavg,kzavg);
31 kxdiff=0;
32 kydiff=0;
33 kzdiff=0;
34 e1=0;
35 for i=1:nsim
36     kxdiff=(p(i,1)-kxavg);
37     kydiff=(p(i,2)-kyavg);
38     kzdiff=(p(i,3)-kzavg);
39     k=hhm((kxdiff kxdiff)+(kydiff kydiff)+(kzdiff kzdiff));
40     e1 = (2 k/(1+sqrt(1+4 af k)))+e1;
41     %!           sumekdl=sumekdl+e1
42 end
43 e1=e1/nsim;
44 velocity_x=((velx_sum/nsim)+velocity_x)/2;
45 velocity_y=((vely_sum/nsim)+velocity_y)/2;
46 velocity_z=(velz_sum/nsim)+velocity_z;
47 count=count+1;
48 if (sume~=0)
49     sume=((sumel/nsim)+sume)/2;
50 else
51     sume=sumel/nsim;
52 end
53 if (sumekd~=0)
54     sumekd=(e1+sumekd)/2;
55 else

```

```

56     sumekd=e1;
57 end
58 Tel=(2 q ( sumekd-energy0) / (3 kb))+tem;
59 for i=1:nsim
60     loc = round(energy(i)/de);
61     if(loc==0)
62         loc=1;
63     end
64     if(loc>n_lev)
65         loc=n_lev;
66     end
67     velx=hmt1 p(i,1)/(1+2 af energy(i)) sqrt(amt1/am0);
68     %!           velz=hml p(i,3)/(1+af2 energy(i)) sqrt(aml/am0)
69     if (velx~=0)
70         vel_squared(loc)=velx velx+vel_squared(loc);
71         % variable that counts #carriers in an energy bin – doesn't
           reset
72         % for every iteration , adds up over all time iterations
           count1(loc)=count1(loc)+1;
73     end
74 end
75 end
76 lx=max(p(:,5))-min(p(:,5));
77 ly=max(p(:,6))-min(p(:,6));
78 lz=max(p(:,7))-min(p(:,7));
79 vol=lx ly lz;
80
81 [no, f]=fermi(n_lev, nsim, h, q, doping_density, de, amd, af, energy, f);
82 for i=1:n_lev
83     ee = i de;
84     n(i)=no(i)+n(i);
85     if (f(i)~=0)
86         dist(i)=f(i)+dist(i);
87         count2(i)=count2(i)+1;
88     end
89 end

```

90 end

15. Function to compute final seebeck coefficient, thermal and electrical conductivities

```
1 function [s, seeb, ke]=transport(n_lev, h, q, kb, Efermi, tem, de, amd, af,
    tot_scat, velocity_x, velocity_y, velocity_z, vel_squared, dist, Tel)
2
3 for i=1:n_lev
4     ene(i)=i de;
5     df(i)=(-q/(2 kb tem)) (1/(1+cosh((ene(i)-Efermi) q/(kb tem))));
6 end
7 % ! Use slope for calculating df/de from MC simulation
8 % ! call slope(n_lev, f, ene, df)
9 const=8 pi sqrt(2) (amd^(3/2))/(h 2 pi)^3;
10
11 % ! velocity=velocity_x velocity_x+velocity_y velocity_y+velocity_z
    velocity_z
12 for i=1:n_lev
13     g = sqrt(ene(i) q (1+af ene(i))) const (1+2 af ene(i));
14     sigma(i)=(vel_squared(i)) g/tot_scat(i);
15     % signal(i)=q q (velavg_z 2) g/tot_scat(i)
16 end
17
18 s=0;
19 a=0;
20 b=0;
21 c=0;
22 for i=1:n_lev
23     s=(sigma(i) (-df(i)) de)+s;
24     a=(sigma(i) (-df(i)) de (ene(i)-Efermi) q)+a;
25     b=(sigma(i) (-df(i)) de ene(i) q)+b;
26     c=(sigma(i) (-df(i)) de (ene(i) q)^2)+c;
27     %! s(2)=(signal(i) (-df(i)) ene(i))+s(2)
28     %! a(2)=(signal(i) (-df(i)) ene(i) (ene(i)-Efermi) q/(kb tem))+
        a(2)
```

```

29 end
30
31 seeb=1/(q tem) ( a/s);
32 %!seeb(2)=kb a(2)/(s(2) q)
33 ke=1/tem ( c-(b^2/s));
34 s=q q s;
35
36 x=zeros(n_lev,3);
37 for i=1:n_lev
38     x(i,:)= [ene(i) vel_squared(i) df(i)];
39 end
40 dlmwrite('time 1.txt',x);
41
42 end

```

16. Function to output final calculated values

```

1 function write_final(Efermi, n_lev, tem, de, velocity_x, velocity_y,
2     velocity_z, sume, sumekd, Tel, f, n, s, seeb, freq, ke, vel_squared)
3 file_1='final f.txt';
4 file_2='final average.txt';
5 file_3='velocity(energy).txt';
6 [m, pos]=max(n);
7 loc=pos(1);
8 ee=loc de;
9 seebeck=(ee-Efermi)/tem;
10 fprintf('velocity_x = %d \n', velocity_x);
11 fprintf('velocity_y = %d \n', velocity_y);
12 fprintf('velocity_z = %d \n', velocity_z);
13 fprintf('energy = %d \n', sume);
14 fprintf('energykd= %d \n', sumekd);
15 fprintf('electron temperature= %d \n', Tel);
16 fprintf('electrical conductivity= %d \n', s);
17 fprintf('seebeck= %d \n', seeb);
18 fprintf('seebeckave= %d \n', seebeck);

```

```

18 fprintf('electronic thermal conductivity= %d \n',ke);
19
20 kp=1.6;
21 z=seeb seeb s tem/(kp+ke);
22 fprintf('ZT= %d \n',z);
23
24 for i=1:6
25     fprintf('Freq %d = %d \n',i,freq(i));
26 end
27 x=zeros(n_lev,3);
28 y=zeros(n_lev,2);
29 for i=1:n_lev
30     ee=i de;
31     x(i,:)=[ee f(i) n(i)];
32     y(i,:)=[ee vel_squared(i)];
33 end
34 dlmwrite(file_1,x);
35 dlmwrite(file_3,y);
36 end

```