Monte Carlo Simulated Heat Transport in Semiconductor Nanostructures

by

GRAHAM GIBSON

A thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Science (M.Sc) in Computational Science

Office of Graduate Studies

Laurentian University

Sudbury, Ontario, Canada

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE
Laurentian Université/Université Laurentienne
Office of Graduate Studies/Bureau des études supérieures

Title of Thesis Titre de la thèse	Monte Carlo Simulated Heat Transport in Semiconductor Nanostructures	
Name of Candidate Nom du candidat	Gibson, Graham	
Degree Diplôme	Master of Science	
Department/Program Département/Programme	Computational Sciences	Date of Defence Date de la soutenance October 31, 2022

APPROVED/APPROUVÉ

Thesis Examiners/Examinateurs de thèse:

Dr. Ralf Meyer
(Supervisor/Directeur(trice) de thèse)

Dr. Gennady Chitov
(Committee member/Membre du comité)

Dr. Aaron Langille
(Committee member/Membre du comité)

Dr. Rachel Wortis
(External Examiner/Examinateur externe)

Approved for the Office of Graduate Studies
Approuvé pour le Bureau des études supérieures
Tammy Eger, PhD
Vice-President Research (Office of Graduate Studies)
Vice-rectrice à la recherche (Bureau des études supérieures)
Laurentian University / Université Laurentienne

Abstract

Nanoscale energy transport is a topic of considerable interest as heat transport at these scales can no longer be accurately predicted by diffusion theory. An alternative approach is to use the Boltzmann transport equation, but this equation is challenging to solve in the case of phonon transport, and its exact resolution is currently one of the open research subjects in mathematics.

A program has been developed to study nanoscale heat transport by solving the Boltzmann transport equation using two variations of a phonon Monte-Carlo method. The first variation is primarily derived from the works of Mazumber and Majumber (2001). The second variation follows the process of Peraud and Hadjiconstantinou (2012). While both variations follow methodology from existing works, the implementation details are unique. The simulation procedures differ from existing methods by incorporating a ‘system evolution’ algorithm that allows temperatures throughout the system to be periodically updated while simulating phonons one-by-one.

The resulting software can rapidly simulate heat transport in relatively complex geometries. The Monte Carlo portion of the software is implemented using parallelized C++ code. Simulating phonons one-by-one makes the parallelization scheme natural and straightforward, although more sophisticated parallelization schemes may result in further computational speedup. The user input is a self-documenting JSON file generated via a Python script.

The software is used to study thermal transport through various silicon and germanium nanostructures. Benchmark simulation testing shows that the temperature profiles produced by the simulations largely agree with analytical results and results from the literature, as does the predicted thermal conductivity. However, the thermal conductivity is quite sensitive to the relaxation rates that are used.

While both variations of the phonon Monte Carlo method presented in this study strike a good balance between accuracy and efficiency and retain an intuitive connection to the problem physics, a noticeable difference in computational efficiency and precision is observed. With the exceptions of low-temperature ranges and possibly systems with extreme temperature differences, the second variation should be preferred when considering computational performance and precision.

Acknowledgements

To Dr. R. Meyer. I consider this project the culmination of much that you have taught me over the past five years. From my first lines of C++ to parallel programming and techniques de méthodes numériques, I've used it all here in some form or another. It's been an honour to have been your student and to have you as an advisor.

Table of Contents

Abstract **ii**

Acknowledgements **iv**

Table of Contents **v**

List of Tables **viii**

List of Figures **ix**

1 Introduction **1**

2 Modeling Heat Transport **5**

2.1 General Heat Conduction 6

2.2 Fourier's Law 8

2.3 Lattice Vibrations and Phonons 9

2.3.1 Dispersion Relations 12

2.3.2 Phonon Distribution 16

2.3.3 Relaxation Times 20

2.4 The Boltzmann Transport Equation 27

3 The Phonon Monte Carlo Method **30**

3.1 Previous Works 30

3.2 The Monte Carlo Method 32

3.3 Overview 35

3.4 Full Simulation 37

3.4.1 Phonon Creation 39

3.4.2 Energy-Based Formulation 41

3.4.3 Phonon Initialization 42

3.4.4 Surface Interactions 47

3.4.5 Phonon Origination 51

3.4.6 Drifting 52

3.4.7 Scattering 55

3.4.8 Measurements 57

3.4.9 Temperature Calculations 58

3.5 Deviational Simulation 59

3.5.1 Equilibrium Temperature and Deviational Particles 60

3.5.2 Linearized Boltzmann Equation 62

3.5.3 Phonon Origination 64

3.5.4 Temperature and Flux Calculations 65

3.6 System Evolution 65

3.7 Advantages of the Energy-Based Formulation 68

3.8 Time-Step Shortcomings 70

4 Results **72**

4.1 Nanowires 73

4.1.1 Thermal Conductivity 73

4.1.2 Ballistic Regime 78

4.1.3 Diffusive Regime 80

4.2 Kinked Nanowires 84

4.3 Thin Wafers 94

4.4 Simulation Comparison 96

5 Summary **98**

5.1 The Phonon Monte Carlo Method 98

5.2 Software Details 99

5.3 Possible Improvements 101

Bibliography **103**

Appendix A Simulation Code **109**

Appendix B Sample JSON Input File **116**

List of Tables

Table 2.1 Data used to produce silicon and germanium dispersion curves. . 15
Table 2.2 Relaxation time parameters for silicon and germanium. 25

Table 3.1 Combined CDF and probability table for silicon at 300 K. . . . 45
Table 3.2 Combined group velocity weighted CDF and probability table for silicon at 300 K. 48
Table 3.3 Combined relaxation time-weighted CDF and probability table for silicon at 300 K. 57

Table 4.1 Common simulation settings for the thermal conductivity benchmarks. 75
Table 4.2 Common simulation settings for the ballistic regime benchmarks. 78
Table 4.3 Emitting surface temperatures for the profile in Figure 4.7. . . . 80
Table 4.4 Common settings for the kinked wire simulations. 85
Table 4.5 Common settings for the wafer simulations. 94

List of Figures

Figure 1.1 CPU clock rates and power consumption. 2

Figure 2.1 Conduction demonstrated by heating a metal bar over a fire. . 6

Figure 2.2 Temperature profile across a solid material. 7

Figure 2.3 Heat conduction through a metal bar. 9

Figure 2.4 Crystal structure of a diamond. 10

Figure 2.5 Diffuse and ballistic transport. 11

Figure 2.6 Phonon vibration modes. 13

Figure 2.7 Phonon dispersion curves for silicon and germanium. 14

Figure 2.8 Group velocities of silicon and germanium. 16

Figure 2.9 Silicon and germanium densities of states. 18

Figure 2.10 Phonon-phonon scattering relaxation times 25

Figure 2.11 Relaxation times for silicon and germanium at 300 K. 26

Figure 2.12 Relaxation times for silicon and germanium at 50 K. 27

Figure 3.1 A linear geometric system subdivided into seven cells with two emitting surfaces. 36

Figure 3.2 Flowchart of the core steps in each iteration of the phonon Monte Carlo method. 38

Figure 3.3 Phonons per frequency bin for silicon and germanium. 40

Figure 3.4 Arbitrary fixed number of phonons per packet. 40

Figure 3.5 Simulated phonon packets with equivalent energy. 41

Figure 3.6 Energy per frequency bin for silicon and germanium. 43
Figure 3.7 Cumulative energy distribution functions for silicon and germanium at various temperatures. 44
Figure 3.8 Random selection of a phonon from the cumulative energy distribution function. The first random number, R_1, identifies the frequency of bin N. A second random number, R_2, determines the frequency on this interval. 45
Figure 3.9 Cumulative group velocity weighted energy distribution functions for silicon and germanium at various temperatures. . . . 48
Figure 3.10 Cumulative relaxation time-weighted energy distribution functions for silicon and germanium at various temperatures. . . . 56
Figure 3.11 Linear system used to test the system evolution algorithm. . . 67
Figure 3.12 System evolution using a 1000 nm silicon wire. 67
Figure 3.13 Steady-state convergence of the system in Figure 3.12. 68
Figure 3.14 Steady-state convergence with an errant initial equilibrium temperature. 69
Figure 3.15 time-step impact on scattering probabilities at 300K. 70
Figure 3.16 time-step impact on steady-state flux. 71

Figure 4.1 Nanowire specifications. 74
Figure 4.2 Thermal conductivity as a function of wire length in silicon at 300K. 76
Figure 4.3 Thermal conductivity as a function of wire length in silicon at 400K. 76
Figure 4.4 Thermal conductivity as a function of wire length in germanium at 300K and 400K. 77
Figure 4.5 Ballistic results for silicon with $T_H = 20$ K and $T_C = 10$ K. . . 79

Figure 4.6 Ballistic results for silicon with $T_H = 40$ K and $T_C = 30$ K. . . 79
Figure 4.7 Diffusive regime results for a 5000 nm silicon wire with varying temperature differentials. 81
Figure 4.8 Diffusive regime results for a 5000 nm silicon wire across a 250 K temperature differential. 81
Figure 4.9 Deviational and full simulation results compared with the heat diffusion equation across a 250 K temperature differential. . . 82
Figure 4.10 Deviational and full simulation results compared with the heat diffusion equation across a 4 K temperature differential. 83
Figure 4.11 Kinked nanowire schematic. 85
Figure 4.12 Temperature and flux profiles of a silicon wire with a 30-degree kink using 15 sensors. 87
Figure 4.13 Temperature and flux profiles of a silicon wire with a 30-degree kink using 3108 sensors. 88
Figure 4.14 Temperature and flux profiles of a silicon wire with a 45-degree kink produced using a deviational simulation. 89
Figure 4.15 Temperature and flux profiles of a silicon wire with a 45-degree kink produced using a full simulation. 90
Figure 4.16 Temperature and flux profiles of a silicon wire with a 60-degree kink produced using a deviational simulation. 91
Figure 4.17 Temperature and flux profiles of a silicon wire with a 60-degree kink produced using a full simulation. 92
Figure 4.18 Flux through a kinked silicon wire as a function of the kink angle. 93
Figure 4.19 Temperature and flux profiles of a thin silicon wafer. 95
Figure 4.20 Temperature and flux profiles of a thin germanium wafer. . . . 95

Chapter 1

Introduction

Nanoscale energy transport is a topic of considerable interest [1–4] as heat transport at these scales can no longer be accurately predicted by diffusion theory [5, 6]. Additionally, materials structured at these length scales have demonstrated properties not exhibited at larger scales [7, 8]. These properties have applications in many areas, from computer chips to solar photovoltaics and batteries, among others.

Device miniaturization presents us with many new opportunities, but some challenges arise. More devices per unit area result in increasingly significant amounts of heat generated per unit volume [9]. For example, the construction of transistors now occurs on the nanometer scale. The amount of heat generated in such a small space causes heat management to become a severe problem, as self-heating may lead to a substantial increase in the effective operating temperature of the device, which degrades the device's electrical performance and reliability. A study on self-heating 7 nm field-effect transistors shows that heat confinement increases by as much as 57% in the germanium channel resulting in a 100 K change in the channel temperature [10]. This sizable temperature increase will impair device performance and reliability without efficient heat removal methods. It has also been shown that peak efficiency decreases as power density increases [11]. This trend is well known in CPUs, and Figure 1.1 shows the increase in power usage of CPUs over the past forty years as their clock rates have increased.

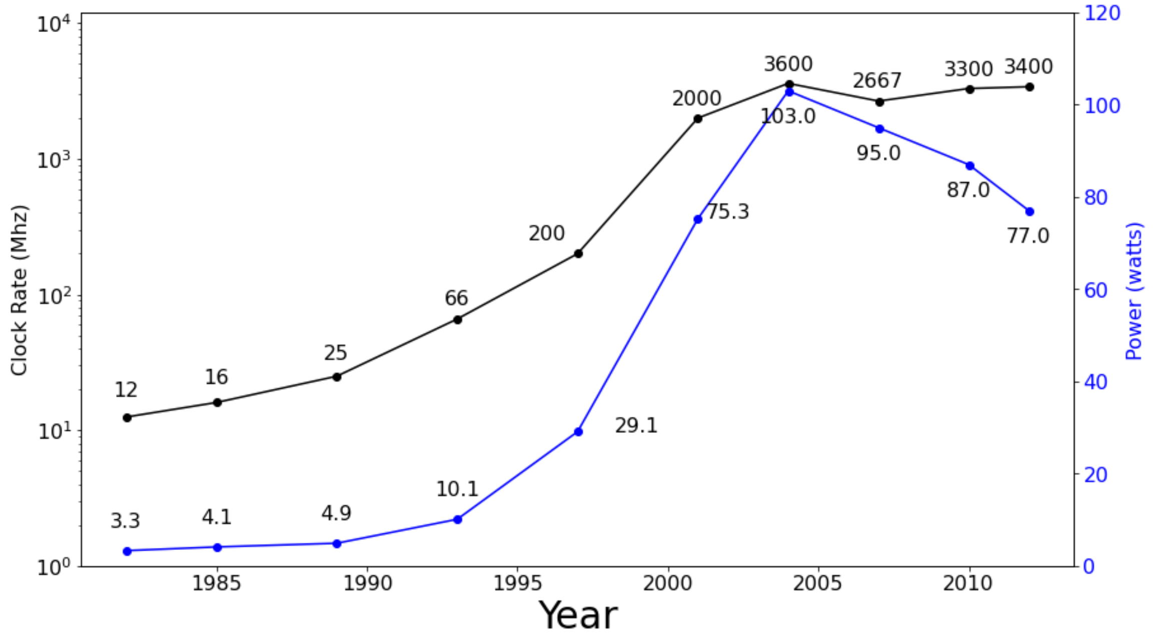

Figure 1.1: CPU clock rates and power consumption. As CPU clock rates increase, their power consumption rises dramatically. Post-2005, the emphasis has been on increasing CPU power efficiency and using multi-core systems.

Despite many advances, much remains unknown concerning the precise mechanisms by which the heat carriers in thermoelectric materials are affected by nanostructures. This lack of understanding is problematic because heat transport phenomena cannot be accurately modelled at these small scales using diffusion approximations [1]. Analyses of heat-transfer processes have shown that the thermal conductivity of nanostructures depends on the shape and size of the sample, properties of its surface, and direction of the heat flow rather than being an intrinsic property of the material as predicted by diffusion theory [12, 13]. Research has shown that the thermal conductivity of silicon nanowires acts in a very unexpected manner when their diameters become extremely small [14]. Other strange effects also appear, such as reduced material thermal conductivity and discontinuities in the temperature distribution near the system boundaries [15]. These findings have led to radically new approaches to determining the thermal conductivity of solids on the nanometer and micrometer scales [16].

Engineering devices not overly prone to self-heating at the nanoscale requires more advanced knowledge of the fundamental mechanisms to which heat carriers adhere. Non-metallic systems like semiconductors pose additional challenges since electrons are no longer the primary heat carrier. However, we still base much of our fundamental understanding on simple models that Holland and Callaway developed over 50 years ago [17, 18].

This thesis explores these challenges and outlines the development of computer software capable of simulating heat transport in semiconductor nanostructures. The simulation software uses a Monte Carlo method, which proves to be exceptionally powerful in this situation. The final software can simulate various nanoscale structures ranging from simple linear wires to wafers, kinked wires, and potentially mesoporous materials. It is hoped that this tool will provide insights into heat transport at the nanoscale leading to more efficient device design and the economic and environmental benefits that follow.

Thesis Outline

In Chapter 2, the manner in which nanoscale heat transport deviates from diffusion theory is discussed. This reveals beneficial information about the underlying heat carriers. The frequency-dependent phonon Boltzmann transport equation is introduced to address issues that occur when applying Fourier's Law at the nanoscale.

Chapter 3 details how a Monte Carlo method can solve the phonon Boltzmann transport equation using the relaxation time approximation. Two different versions of this numerical approach have been implemented. The resulting computer program calculates the energy (temperature) distribution and heat flux in silicon and germanium nanostructures subjected to any number of heat sources or sinks.

Chapter 4 discusses various results produced by the method described in Chapter 3. Most notably, the thermal conductivity of silicon and germanium nanowires predicted by the simulation are compared with known bulk values and existing simulation results.

Finally, Chapter 5 concludes the thesis and offers possibilities for future improvements to the simulation.

Chapter 2

Modeling Heat Transport

The study of heat transport constitutes a vast field of research. At the macroscopic scale, equations governing heat transport are well established. However, the analytical resolution of these equations is only possible in some instances, and while these equations generally do an excellent job of describing the steady-state of a system, they often fail to describe how a system transitions to this steady-state.

Beyond the mathematical difficulty of analytically solving these equations, there is the question of the domain on which they can be applied. This chapter presents the limits of these macroscopic equations and why it is necessary, when dealing with nanostructures, to correct them, improve them, or even establish new ones.

The first section of this chapter summarizes some intuitive ideas about general heat conduction. This summary leads into a formal depiction of heat conduction known as Fourier's Law. Next, we take a more in-depth look at the underlying mechanisms of heat transport, which helps explain why Fourier's Law is not an appropriate model to use at the nanoscale. Finally, the frequency-dependent phonon Boltzmann transport equation is introduced to address issues when applying Fourier's Law at the nanoscale.

2.1 General Heat Conduction

Heat transfer is generally considered to occur via three different mechanisms: conduction, convection and radiation. Of these, conduction is likely the mechanism through which we develop our intuitive notions about heat transfer and is the mechanism on which this study will focus.

Heat or thermal conduction occurs within a material or between two materials in contact without the involvement of mass flow and mixing. Figure 2.1 demonstrates conduction within a material in the context of a rectangular metal bar being placed over a fire[1].

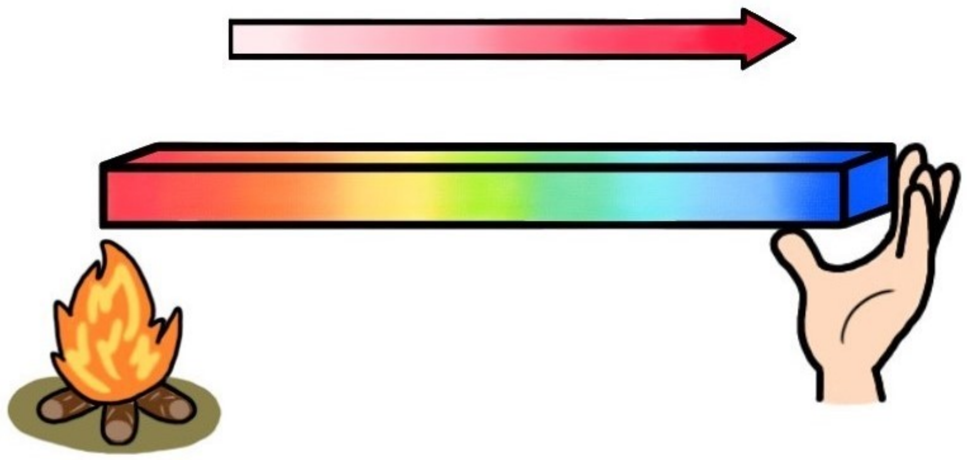

Figure 2.1: Conduction demonstrated by heating a metal bar over a fire. The flow of heat from the hot end of the rod to the cool end is an example of conduction within a material.

Intuitively, our experience tells us that the heat from the hot end of the rod will travel through the rod to the cooler end. If we do not want to burn our hands, we may be interested in the factors that affect the speed or rate of heat flow through the rod. For example, we can surmise that a more powerful heat source will increase the

[1]In Figure 2.1, the transfer of heat from the fire to the rod does not occur via conduction. However, we are less concerned about how the rod is heated and more interested in how the heat is transmitted through the rod.

likelihood of sustaining an injury when grabbing the rod to pull it out of the fire. It also seems probable that all other things being equal, grabbing the end of a shorter rod will reinforce our intuitions about heat conduction.

Perhaps more interestingly, there is a factor specific to the material itself, which is not seen in the figure above. Given a piece of wood and a piece of metal of equivalent dimensions, we can conjecture that the rate of heat flow will be greater through the metal rod than through the piece of wood.

We also want to know how the temperature changes along the length of the rod. A reasonable starting point would be that the temperatures along the length should be, at most, the temperature closest to the fire and should be at least the cool end of the rod. Assuming the temperatures along the length follow a linear path from the hot end to the cool end, we end up with Figure 2.2.

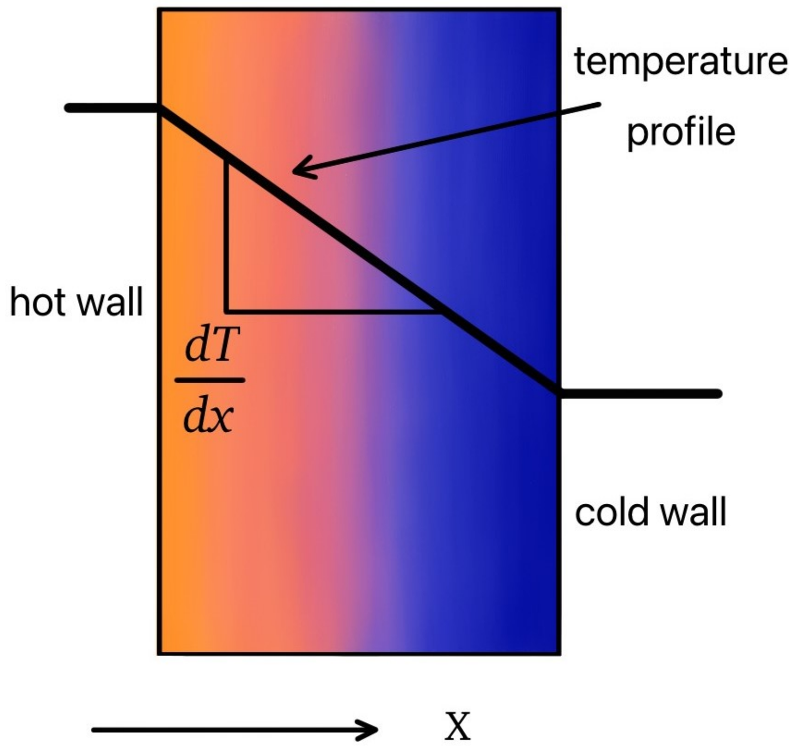

Figure 2.2: Temperature profile across a solid material.

In Figure 2.2, we see a linear temperature profile where $\frac{dT}{dx}$ is the temperature gradient.

While these intuitive notions about the nature of heat conduction may seem evident to us, mathematically expressing them is a challenging task.

Luckily, "the giants before us" have already undertaken this challenge.

2.2 Fourier's Law

In 1822, Fourier experimentally determined the law which bears his name [19]. Fourier's law formalizes the concept of a material's ability to transmit heat and can be expressed as:

$$F = -\kappa A \frac{\partial T}{\partial x} \tag{2.1}$$

The variables are defined as follows:

- F is the energy flowing through the system per unit time (W).
- κ is the thermal conductivity of the material ($W \cdot m^{-1} \cdot K^{-1}$).
- A is the cross-sectional area through which the heat flows (m^{-2}).
- $\frac{\partial T}{\partial x}$ is the temperature gradient ($K \cdot m^{-1}$).

Fourier's law captures all the intuitive notions from the previous section. The amount of heat transmitted through a material will increase as the temperature gradient increases, which occurs by increasing the temperature difference across the material or reducing the length of the material. A greater cross-sectional area will lead to more significant heat flow through the material.

Fourier's law also describes the idea of a material's intrinsic capacity to transmit heat through the concept of thermal conductivity. In Section 2.3, we take a more detailed look at what this term represents. The above relationships are depicted in Figure 2.3 where the temperature gradient is $\frac{T_H - T_L}{d}$.

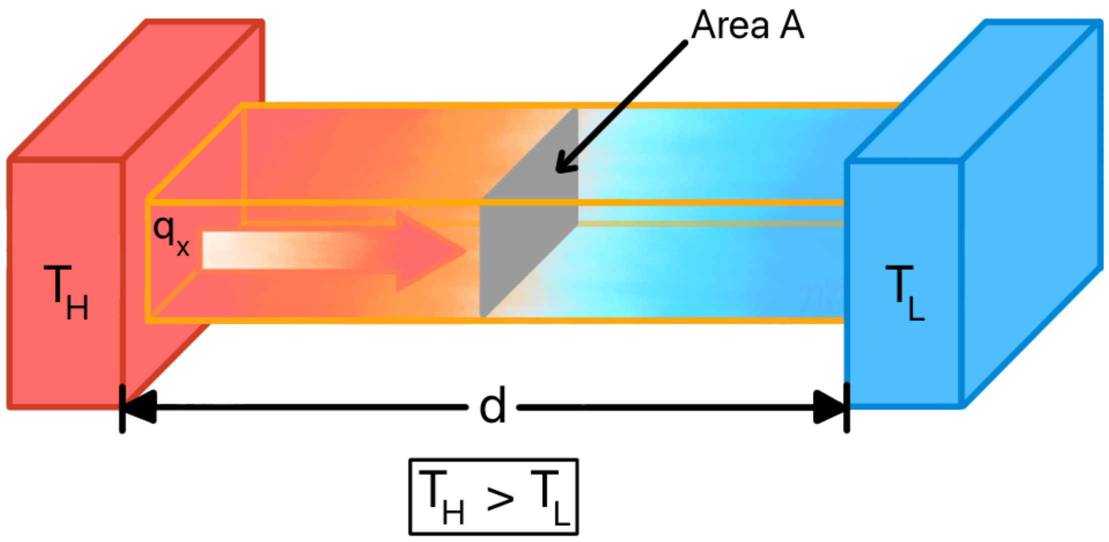

Figure 2.3: Heat conduction through a metal bar. The rate at which heat flows through the material is contingent on the temperature differential across the structure and the material thickness, length, and thermal conductivity.

Equations based on Fourier's law are robust, predictive, and not overly complex. These reasons are why equations based on Fourier's law are the bedrock of engineering analysis at the macroscopic scale.

However, these equations assume that heat transport is a diffusive process. Unfortunately, this assumption is no longer valid at nanometer length scales, and Fourier's law fails due to classical and quantum size effects [20–22]. To see why this is the case, we will dig deeper into the physics of the underlying heat transport mechanisms.

2.3 Lattice Vibrations and Phonons

Matter is generally found in solid, liquid or gas form. We can also differentiate solid materials according to their atomic organization. If a structure's constituent atoms or ions are organized in a periodic pattern, they are considered crystals [23].

The elements will crystallize according to a particular pattern depending on the pressure and temperature conditions and the number of covalent bonds. Silicon and germanium crystallize according to a mesh or lattice-like the one depicted in Figure 2.4.

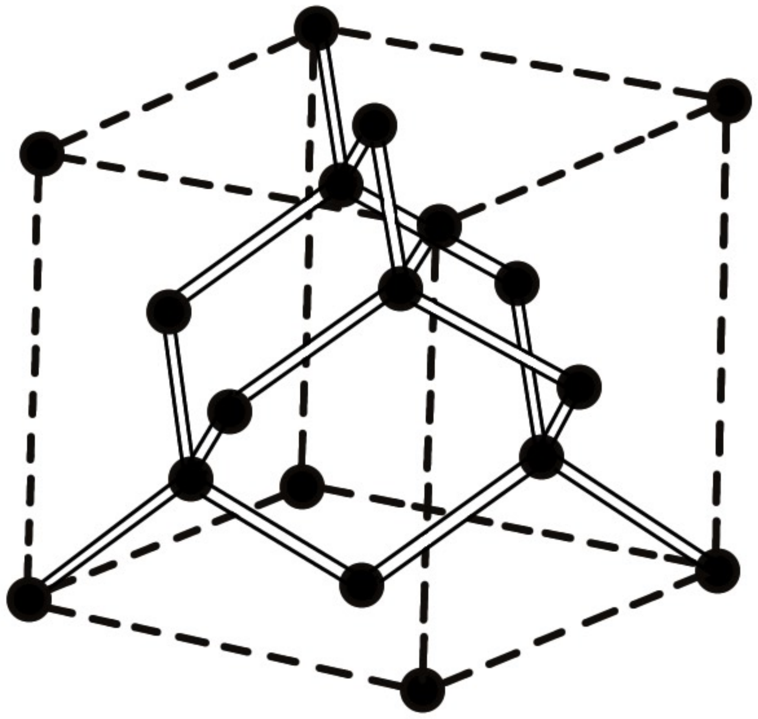

Figure 2.4: Crystal structure of a diamond.

This mesh or crystal lattice can be visualized as a complex three-dimensional array of masses representing individual atoms and springs representing atomic bonds. As atoms deviate from their equilibrium positions within this lattice, they create vibrational waves, which propagate through the crystal and carry energy. Lattice vibrational waves dominate heat conduction in dielectric materials and most semiconductors.

These lattice vibrations are quantized, and each quanta is called a phonon. A phonon is the basic energy quantum of a lattice vibration analogous to a photon, the basic energy quantum of an electromagnetic wave. Similar to photons, phonons have both wave-like and particle-like properties. Size effects appear if the characteristic system dimension is comparable to or smaller than the phonon mean free path, denoted Λ. The mean free path is the average distance phonons travel between successive collisions. What exactly constitutes a collision is the topic of Section 2.3.3.

The mean free path is often estimated from kinetic theory and is used in thermal conductivity calculations of a material/structure of interest. The thermal conductivity is typically calculated as follows [24]:

$$\kappa = \frac{1}{3} C_V v_P \Lambda \tag{2.2}$$

In Eq. (2.2) C_V is the volumetric specific heat capacity of the material and v_P. For example, the phonon mean free path in silicon is on the order of ~300 nm at room temperature in bulk materials [25].

When the characteristic dimension of the system, L, is much greater than the phonon mean free path, phonons will overwhelmingly undergo collisions before travelling the distance L. These collisions cause frequency and direction changes. This is typically the case of thermal conduction in macroscopic materials, and Fourier's law is predicated on diffuse transport being the predominant heat transport mechanism.

Conversely, the transport is said to be ballistic when L is smaller than Λ. The phonons can and often do traverse the distance L without collision. Fourier's law cannot accurately model systems in which ballistic transport dominates.

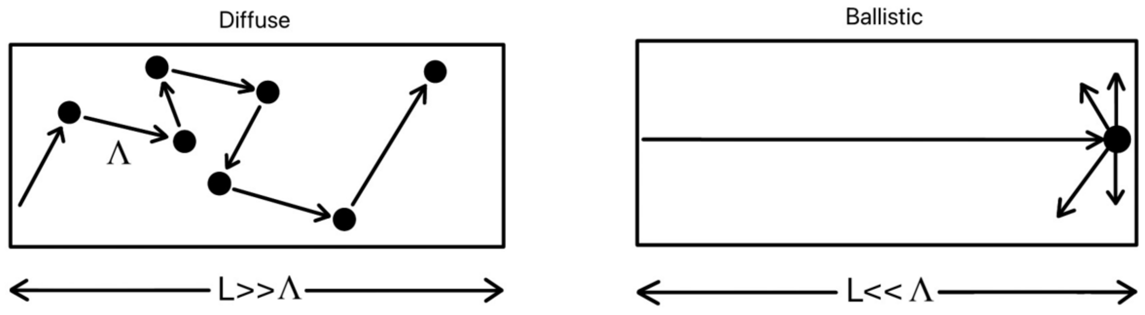

Figure 2.5: Diffuse and ballistic transport.

2.3.1 Dispersion Relations

The atoms in the crystal lattice are not fixed in place. They vibrate around an equilibrium position, generating waves that propagate throughout the structure. These vibrations can be characterized by modelling them as chains of elastically-linked atoms [23], and the equation that relates the frequency of these vibrations to their wave vector is known as a dispersion relation.

Using the characterization of elastically linked atoms, it is possible to obtain solutions for two different polarization modes [23]. When atoms vibrate along the axis of their bond parallel to the wave's motion, the polarization is referred to as longitudinal (L). When they vibrate in a perpendicular plane, the motion usually associated with waves, the polarization is referred to as transverse (T). Transverse waves typically have a lower frequency than their longitudinal counterparts.

In addition to longitudinal and transverse modes, another distinction can be made in certain materials between acoustic (A) and optical (O) modes. All materials have longitudinal acoustic and transverse acoustic phonon modes whose frequency approaches zero as the wavelength goes to zero. These phonons are called acoustic since they correspond to acoustic waves. Depending on its lattice structure, a material can also have longitudinal optical and transverse optical modes. The frequencies of these modes are generally higher than those of acoustic modes and do not approach zero as the wavelength goes to zero. The optical terminology comes from the dipole moment these vibrations generate in the ionic crystals. The presence of this dipole moment allows these vibrations to interact with photons, hence why they are referred to as being in the optical mode. Optical phonons occur at higher temperatures and with higher energy, whereas acoustic phonons occur at lower temperatures and energy [23].

Therefore we have four types of polarization modes. The four modes are referred to as TA, LA, TO, and LO using the above notation. It is possible to determine the number of branches per polarization mode, but this number depends on the crystal lattice's properties. Let p be the number of atoms in the basis, then there are $3p$ modes of vibration, of which $3(p-1)$ are optical [23]. So there are at most three acoustic modes in a given crystalline direction. Silicon and germanium, the materials of interest in this study, each have two atoms per unit cell [23]. From this, the following six polarization branches can be deduced:

- 1 LA branch
- 2 TA branches
- 1 LO branch
- 2 LA branches

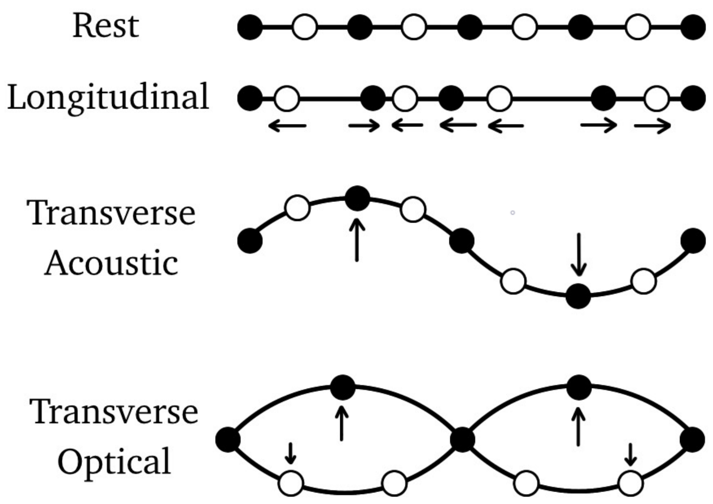

Figure 2.6: Phonon vibration modes. In-phase vibrations (acoustic) and out-of-phase vibrations (optical) are shown for the transverse polarization branch.

Research in the 1960s using diffusion of X-rays and neutron scattering experimentally uncovered the dispersion relations discussed here [26]. The curves produced in the study highlight an aspect of the crystals which has thus far not been mentioned.

Crystals are anisotropic, meaning the dispersion relations will vary depending on the phonon's direction of propagation through the crystal.

However, in this work, the lattice arrangement is assumed to be isotropic. Isotropy is a common assumption in most previous works on the subject. Using the isotropy assumption, the dispersion relations are independent of propagation direction, and this assumption leads to the dispersion relations seen in Figure 2.7. These relations are used in the Monte Carlo simulation for silicon and germanium transverse and longitudinal polarization branches.

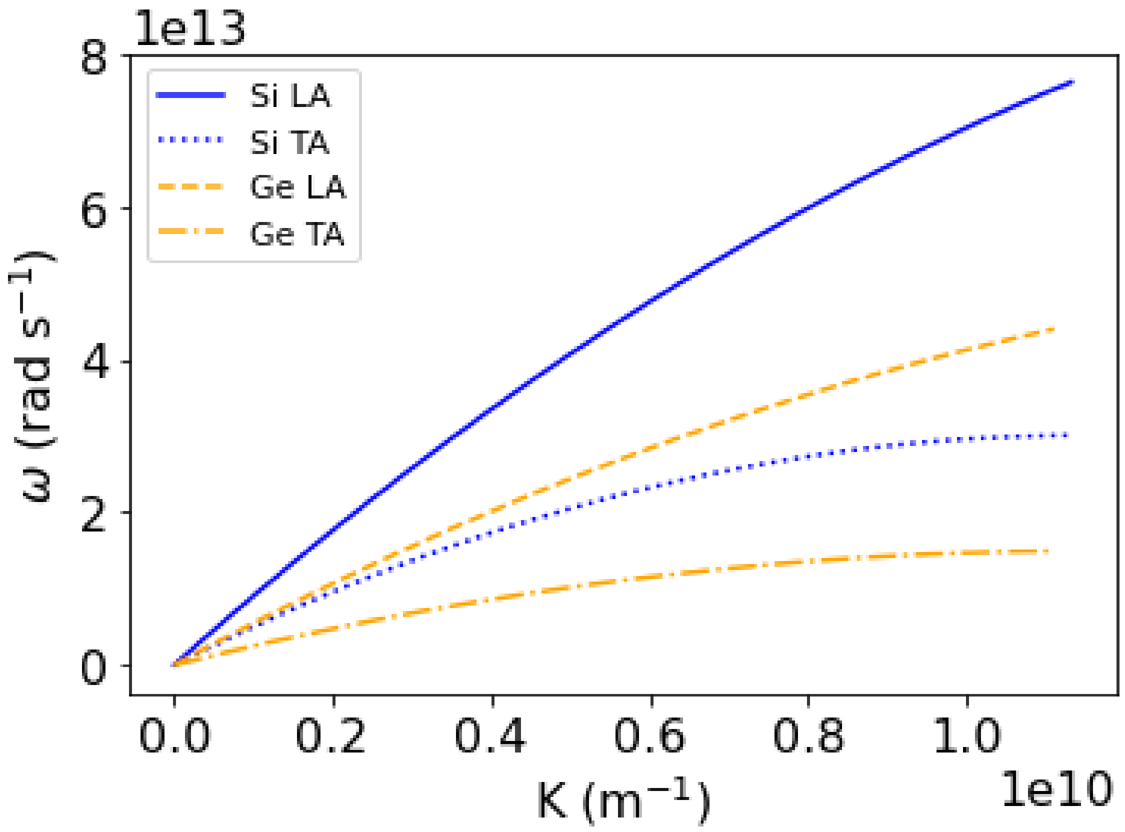

Figure 2.7: Phonon dispersion curves for silicon and germanium. $K_{max\,si} = 1.1326 \times 10^{10}$ m^{-1} and $K_{max\,ge} = 1.1105 \times 10^{10}$ m^{-1} [3] using Jean [27] parameterization data from Table 2.1.

These curves are produced by using the Jean parameterizations [27] from Table 2.1 and the following parabolic [28] fit for both silicon and germanium:

$$\omega = c_\alpha K^2 + v_\alpha K \tag{2.3}$$

Additionally, the dispersion relations allow us to solve Eq. (2.9) and provide the phonon group velocity at a particular frequency using Eq. (2.4). It should be noted that the transverse branch is doubly degenerate ($g_{TA} = 2$), whereas the longitudinal branch is non-degenerate ($g_{LA} = 1$).

This study does not take optical phonons into account. See Section 5.3 for a brief discussion on this topic.

Table 2.1: Data used to produce silicon and germanium dispersion curves.

		Jean [27]		Holland [18]	Wong [15]	Pop [28]
	Units	Si	Ge	Si	Si	Si
c_{TA}	$\times 10^{-7}$ (m^{2}s^{-1})	-2.28	-1.14	-2.01	-2.234	-2.26
v_{TA}	$\times 10^{3}$ (m s^{-1})	5.24	2.60	5.23	5.24	5.23
c_{LA}	$\times 10^{-7}$ (m^{2}s^{-1})	-2.22	-1.50	-2.26	-2.278	-2.00
v_{LA}	$\times 10^{3}$ (m s^{-1})	9.26	5.63	9.01	9.28	9.01
$\omega_{1/2}$	$\times 10^{13}$ (rad s^{-2})	2.42	1.23	2.417	2.417	-

Using the dispersion relations, it is possible to ascertain two characteristic speeds for each polarization branch. The group velocity corresponds to the statistical average speed at which phonons travel in the medium [23]. In other words, the dispersion relation's slope represents a phonon's propagation speed which also corresponds to the speed of sound within the lattice.

$$v_g(\omega, p) = \frac{\partial \omega}{\partial K} \tag{2.4}$$

At low values of K, the dispersion relation is nearly linear, and the speed of the sound becomes independent of the phonon frequencies. The phase velocity is the speed at which the phase of a plane wave with a unique frequency, and wavelength, propagates through the medium.

$$v_{\varphi}(\omega, p) = \frac{\omega}{K} \tag{2.5}$$

The phase velocity is not required for the Monte Carlo simulation and is mentioned here only for completeness.

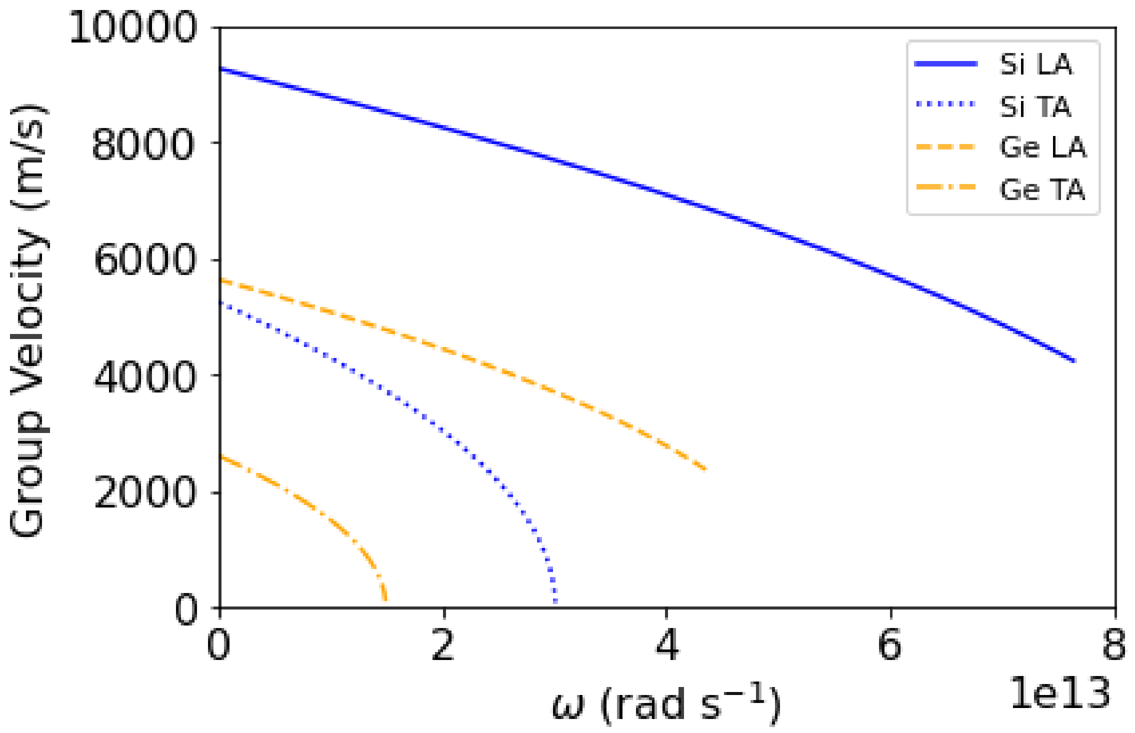

Figure 2.8: Group velocity as a function of frequency for silicon and germanium.

2.3.2 Phonon Distribution

For a system in thermodynamic equilibrium, the notion of temperature is well-defined. At a given temperature, the average number of phonons occupying a particular vibrational state can be described by the equilibrium phonon occupation number, $\langle n \rangle$.

Since phonons are bosons, this quantity can be calculated using the Bose-Einstein distribution[2] [23]:

$$\langle n_{\mathbf{K},p} \rangle = \frac{1}{\exp\left[\frac{\hbar\omega_{\mathbf{K},p}}{k_B T}\right] - 1} \tag{2.6}$$

where ω is the angular frequency of the phonon and $k_b = 1.38 \times 10^{-23}$ J/K is the Boltzmann constant. Hence, $\langle n_{\mathbf{K},p} \rangle$ represents the local thermodynamic phonon population with polarization p and wave vector $\mathbf{K}$. The total vibrational energy of the crystal can be written as [23]:

$$E = \sum_p \sum_{\mathbf{K}} \left(\langle n_{\mathbf{K},p} \rangle + \frac{1}{2} \right) \hbar\omega_{\mathbf{K},p} \tag{2.7}$$

where E is the total vibrational energy. If we assume that the phonon wave vectors are sufficiently dense over $\mathbf{K}$, which is typically the case [1], then the summation in Eq. (2.7) can be replaced by an integral. Moreover, using $D_p(\omega)$, the phonon state density, we can integrate over the frequency space rather than over wave-vector space, yielding:

$$E = \sum_p \int_\omega \left(\langle n_{\omega,p} \rangle + \frac{1}{2} \right) \hbar\omega D(\omega,p) g_p d\omega \tag{2.8}$$

$D_p(\omega)d\omega$ represents the number of vibrational modes in the frequency range $[\omega,\, \omega + d\omega]$ for polarization p and where g_p is the degeneracy of the polarization branch under consideration. In the case of an isotropic three-dimensional crystal, we have [23]:

[2]The Bose-Einstein distribution comes from the theory of statistical mechanics, which describes the motion of large numbers of atoms and molecules. In brief, the theory posits that if we know the probability distribution (in this case, the Bose-Einstein distribution) describing the likelihood that particles will behave in specific ways, then we can perform calculations and make predictions about the behaviour of large particle ensembles without knowing the precise position and velocity of each particle in the ensemble.

$$D(\omega, p)d\omega = \frac{d\mathbf{K}}{(2\pi/L)^3} = \frac{VK^2dK}{2\pi^2} \tag{2.9}$$

Using the definition of group velocity from Eq. (2.4) we can write Eq. (2.9) as:

$$D(\omega, p) = \frac{VK^2}{2\pi^2 v_g(\omega, p)} \tag{2.10}$$

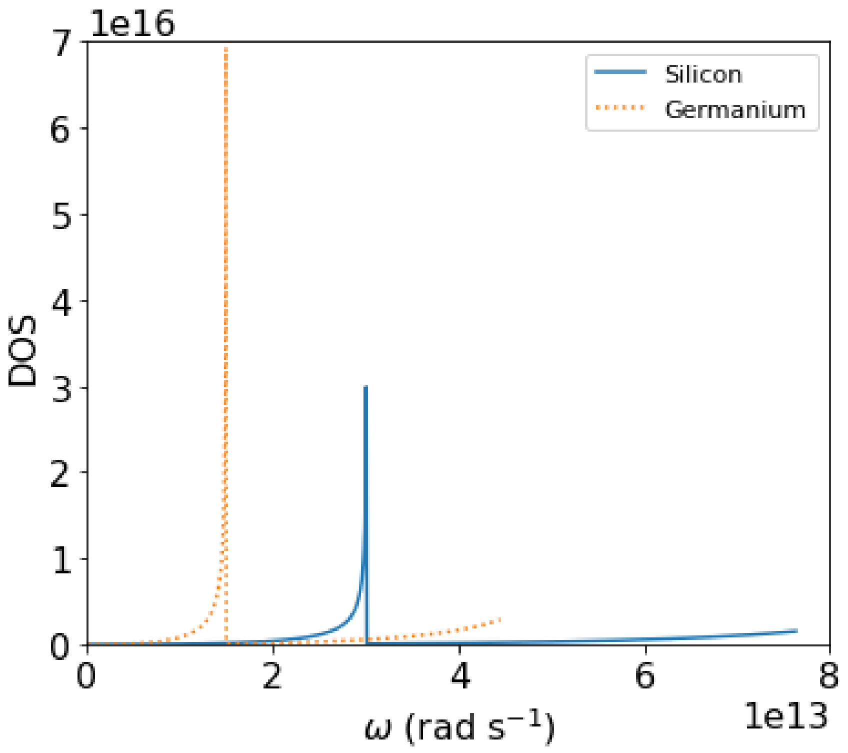

Figure 2.9: Silicon and germanium densities of states per unit volume. Calculated from Eq. (2.10) using the acoustic branches.

which allows us to obtain the following form of Eq. 2.7:

$$E = V\sum_p \int_\omega \left[\frac{\hbar\omega}{\exp\left(\frac{\hbar\omega}{k_BT}\right) - 1}\right] \frac{K^2}{2\pi^2 v_g(\omega, p)} g_p d\omega \tag{2.11}$$

Eq. (2.11) is typically used to estimate the temperature of a material using a numerical inversion. In Section 3.5, another technique to accomplish this task is presented using the following definition of heat capacity:

$$\begin{aligned} C(T) &= \left.\frac{\partial E}{\partial T}\right|_V \\ &= \sum_p \int_\omega \hbar\omega \frac{\partial}{\partial T}\left(\frac{1}{\exp\left(\frac{\hbar\omega}{k_B T}\right)-1}\right)\frac{K^2}{2\pi^2 v_g(\omega,p)} g_p d\omega \\ &= \sum_p \int_\omega \frac{de(\omega)}{dT} D_p(\omega) g_p d\omega \end{aligned} \tag{2.12}$$

where

$$e(\omega) = \frac{\hbar\omega}{\exp\left[\frac{\hbar\omega}{k_B T}\right]-1} \tag{2.13}$$

Hence $\frac{de(\omega)}{dT}$ is the derivative, with respect to temperature, of a modified Bose-Einstein distribution and $D_p(\omega)$ is defined in Eq. (2.9). We can also calculate the number of phonons in a given volume V as:

$$N = V \sum_p \int_\omega \left[\frac{1}{\exp\left(\frac{\hbar\omega}{k_B T}\right)-1}\right]\frac{K^2}{2\pi^2 v_g(\omega,p)} g_p d\omega \tag{2.14}$$

Once the number of phonons in a given volume is known, the phonon heat flux, in units of W/m^2, through that same volume can be calculated as:

$$\phi = \frac{1}{V}\sum_{n=1}^{N} \hbar\omega_n v_g(\omega_n)\cdot \mathbf{k} \tag{2.15}$$

In the Monte Carlo simulation, phonons enter and exit the system via emitting surfaces. These surface types are discussed in Section 3.4.4. To determine the energy that a surface element will emit per unit time, we can use the following equation, which allows us to express the equilibrium phonon intensity $I_{\omega,p}$ [29]:

$$I_{\omega,p} = \frac{1}{4\pi} v_g(\omega,p)\langle n_{\omega,p}\rangle \hbar\omega D(\omega,p) \tag{2.16}$$

This intensity[3] refers to the phonon energy emitted per unit time per unit area per unit solid angle per unit frequency and can be used to derive an expression for the phonon energy emitted per unit time from a surface with area A at temperature T:

$$\begin{aligned} Q &= A \int_{2\pi} \left(\sum_p I_{\omega,p} d\omega \right) \cos\theta g_p d\Omega \\ &= \frac{A}{4} \sum_p \int_\omega v_g(\omega,p) \langle n_{\omega,p} \rangle \hbar\omega D(\omega,p) g_p d\omega \end{aligned} \tag{2.17}$$

The corresponding number of phonons emitted by the same surface can be calculated using:

$$N_{face} = \frac{A}{4} \sum_p \int_\omega v_g(\omega,p) \langle n_{\omega,p} \rangle D(\omega,p) g_p d\omega \tag{2.18}$$

2.3.3 Relaxation Times

While propagating through materials, phonons are subject to collisions which may alter their properties. These alterations come in the form of changes to energy/frequency, polarization and wave vector. The average time interval between these collisions or scattering events is known as the relaxation time, τ. The mean distance between two collisions is $\Lambda = \tau \times v$, where v is the phonon velocity within that material. This is the mean free path introduced in Section 2.3. In a large-scale crystal, the following scattering processes are present:

- Phonon-phonon scattering events
- Boundary collisions

[3]The intensity in Eq. (2.16) includes the factor of $\frac{1}{4\pi}$ which is absent from the definition in Majumdar [1].

- Impurity scattering due to defects in the crystal structure

Phonon-phonon scattering events

Interatomic forces are not purely harmonic, which results in collisional processes involving three or more phonons [23]. Collisions of this type will alter the characteristics of the involved phonons. In these types of scattering events, a phonon can combine with another phonon into a third. Conversely, a phonon can be broken down into two other phonons. There are two types of ternary processes, and they are referred to as Normal or N-processes and Umklapp or U-processes.

N-processes are scattering events that lead to the dissociation or recombination of phonons where momentum and energy are conserved. As the temperature of a system increases, the likelihood of scattering more processes involving several phonons also increases. Recall that Eq. (2.14) gives the number of states at high temperatures. Its evolution with temperature is given by the Bose-Einstein function, which, for large values of T, is proportional to T, as seen in the following approximation:

$$\frac{1}{\exp\left[\frac{\hbar\omega}{k_BT}\right]-1} \approx \frac{k_BT}{\hbar\omega} \tag{2.19}$$

Since a phonon has a better chance of being scattered by another phonon as their numbers increase, the occurrence of anharmonic processes increases with temperature. These N-processes conserve both frequency and momentum, which can be written as:

$$\mathbf{K_1} + \mathbf{K_2} = \mathbf{K_3}$$

$$\omega_1 + \omega_2 = \omega_3$$

In this case, the norm of the vector $\mathbf{K_3}$ must be less than the value of $2K_{max}$. If this is not the case, the process is referred to as an Umklapp or U-process. Because Normal processes retain momentum, they do not contribute to the limitation of thermal conductivity, which is a resistive phenomenon.

Umklapp processes are resistive and are the primary dictator of thermal conductivity at high temperatures. This is because U-processes conserve frequency, but they do not conserve momentum. This relationship can be written as:

$$\mathbf{K_1} + \mathbf{K_2} = \mathbf{K_3} + \mathbf{G}$$

$$\omega_1 + \omega_2 = \omega_3$$

where $\mathbf{G}$ is the reciprocal lattice vector not equal to zero. Thus, not all phonons can undergo an Umklapp process since the associated wave vector must be sufficiently large. At low temperatures, phonons are predominately low in energy; hence, the low wave vectors are the most densely populated. As the temperature rises, higher energy phonons become more common and since phonon frequency is an increasing function with respect to the wave vector, the probability of an Umklapp process occurring increases. This phenomenon coincides with the observation of thermal conductivity decreasing as the temperature of a system increases.

Boundary collisions

On the macroscale, the volume of a given material is large enough that the collisions between phonons and boundaries are a relatively rare event. This phenomenon is no longer negligible when we are interested in structures at the nanoscale. More specifically, when we are interested in modelling structures where the characteristic dimension L is less than or of the same order of magnitude as the mean free path ($L \leq \Lambda$).

Unlike a photon, a phonon is confined to the crystalline medium in which it propagates. A phonon cannot escape from a solid environment. Upon impact with a boundary, a phonon will reflect away from the border. This complex interaction depends on many properties, including the surface condition and the phonon wavelength. In this study, the standard approach of allowing the user the specify the degree of surface specularity is taken. Reflections involving this surface will be either specular or diffuse based on the degree of specularity. A specular reflection is perfectly reflective like a mirror, whereas, in a diffuse reflection, the phonon is sent in a random direction in the crystalline space. The exact process is detailed in Section 3.4.7.

Reflections on boundaries are considered elastic, with energy being conserved. In other words, the phonon frequency is unchanged with $\omega = \omega'$. In addition, the phonon polarization is assumed to be unchanged. Only the direction of the phonon wave vector is affected.

When the system dimensions are tiny, thermal conductivity is no longer an intrinsic property of the material but becomes an explicit function of the system dimensions due to these boundary collisions [30].

Impurity scattering

A crystal is theoretically an infinite periodic medium. In reality, most crystals contain irregularities. Atoms may be missing in the pattern, or foreign atoms may have taken the place of atoms which would normally constitute the lattice formation. These punctual irregularities induce variations in the lattice structure and are responsible for impurity scattering events.

This scattering process is elastic, like boundary collisions, and polarization is also preserved. Only the direction of phonon propagation is affected. The relaxation time associated with impurity scattering events can be calculated as follows:

$$\mathcal{T}_I^{-1} = B_I \omega^4 \tag{2.20}$$

where B_I is an adjustable parameter which depends on the impurity concentration of the material [31, 32]. As we will see at the end of this Section, impurity scattering is the dominant form of scattering at low temperatures.

Relaxation time expressions

The calculation of relaxation times can be theoretically complex [33, 34]. Depending on the materials studied and the methods used, different expressions can be considered [4, 35–37]. For this study, the following formalism of relaxation times established by Holland [18] is used:

$$\begin{aligned}
\mathcal{T}_{N_{LA}}^{-1} &= B_L \omega^2 T^3 \\
\mathcal{T}_{U_{LA}}^{-1} &= B_L \omega^2 T^3 \\
\mathcal{T}_{N_{TA}}^{-1} &= \begin{cases} B_{TN} \omega T^4 & \text{if } \omega < \omega_{1/2} \\ 0 & \text{otherwise} \end{cases} \\
\mathcal{T}_{U_{TA}}^{-1} &= \begin{cases} 0 & \text{if } \omega < \omega_{1/2} \\ B_{TU} \frac{\omega^2}{\sinh(\frac{\hbar\omega}{k_B T})} & \text{otherwise} \end{cases}
\end{aligned} \tag{2.21}$$

Here, $\omega_{1/2}$ is the frequency of the TA branch such that $K = K_{max}/2$. Figure 2.10 can be reproduced using Eq. 2.21 and the data in the Jean column from Table 2.2. Many variations of this formalism can be found in [24].

Table 2.2: Relaxation time parameters for silicon and germanium.

		Jean [27]		Holland [18]	Wong [15]
	Units	Si	Ge	Si	Si
B_{TN}	$\times 10^{-13}$ (s K^{-3})	9.0	30.0	9.3	9.3
B_{TU}	$\times 10^{-18}$ (s)	1.9	1.5	5.5	1.7
B_L	$\times 10^{-24}$ (K^{-4})	1.3	2.3	2.0	2.0
B_I	$\times 10^{-45}$ (s^{-3})	1.2	24.0	0	0

Figure 2.10 shows that the increase in temperature induces a decrease in the time between phonon-phonon scattering events.

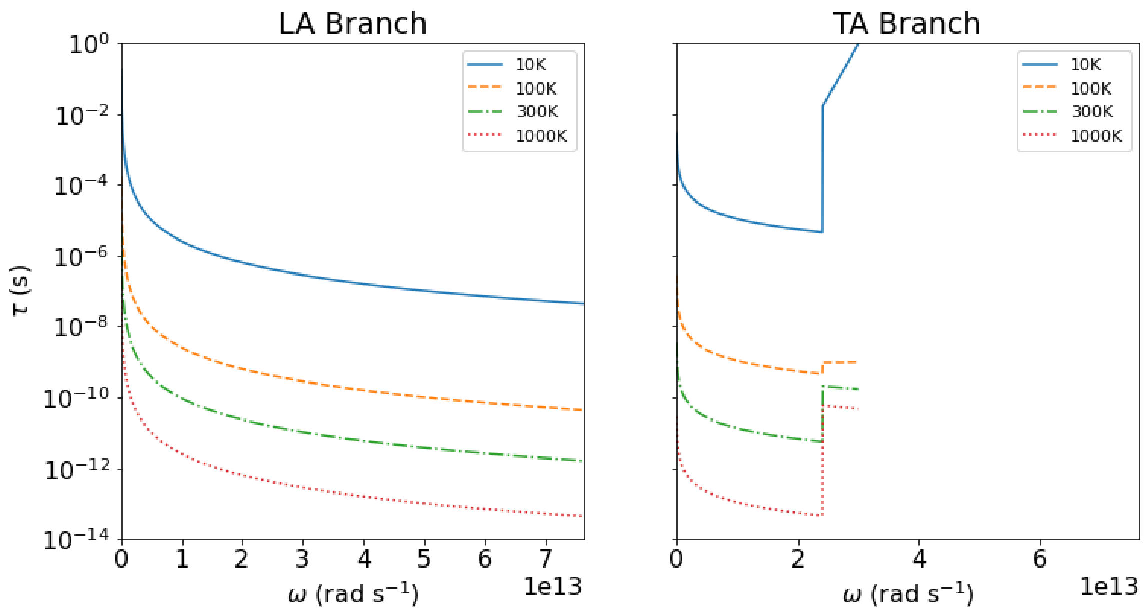

Figure 2.10: Phonon-phonon scattering relaxation times in silicon at temperatures between 10 K and 1000 K.

If the scattering processes are considered to be independent, the total relaxation time can be established using Matthiessen's rule [38]:

$$\mathcal{T}(\omega, T, p)^{-1} = \mathcal{T}_I(\omega)^{-1} + \mathcal{T}_N(\omega, T, p)^{-1} + \mathcal{T}_U(\omega, T, p)^{-1} \tag{2.22}$$

Figure 2.11 shows that at higher temperatures, the total relaxation time is governed by phonon-phonon scattering processes. In Figure 2.12, it is evident that impurity scattering will dominate the total relaxation time since phonon-phonon scattering events are infrequent at lower temperature ranges. In the context of this work, higher temperatures imply 250 K and above, whereas low temperatures are considered 50 K and below.

The jump in the TA branch results from the shift of Normal scattering processes to Umklapp scattering processes, as can be seen from Eq. (2.21). The model for the relaxation times only allows Umklapp scattering for phonons with $\omega \geq \omega_{1/2}$ and Normal scattering when $\omega < \omega_{1/2}$. This formalism is frequently used in the literature [3, 15, 27]. The underlying argument for this cut-off is that low-frequency TA phonons rarely undergo Umklapp scattering, and collisions involving high-frequency TA phonons rarely conserve momentum [18].

The concepts and equations in this Section form the basis for the phonon Monte Carlo method presented in Chapter 3.

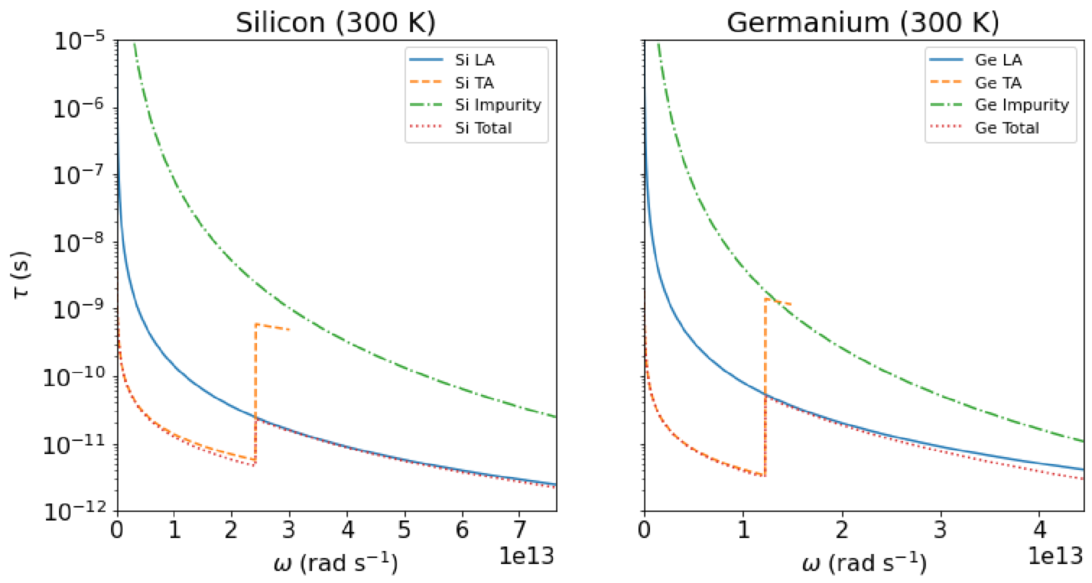

Figure 2.11: Relaxation times for silicon and germanium at 300 K. The total relaxation time is largely dictated by the transverse branch until $w = w_{1/2}$ and then the total relaxation time follows the longitudinal branch.

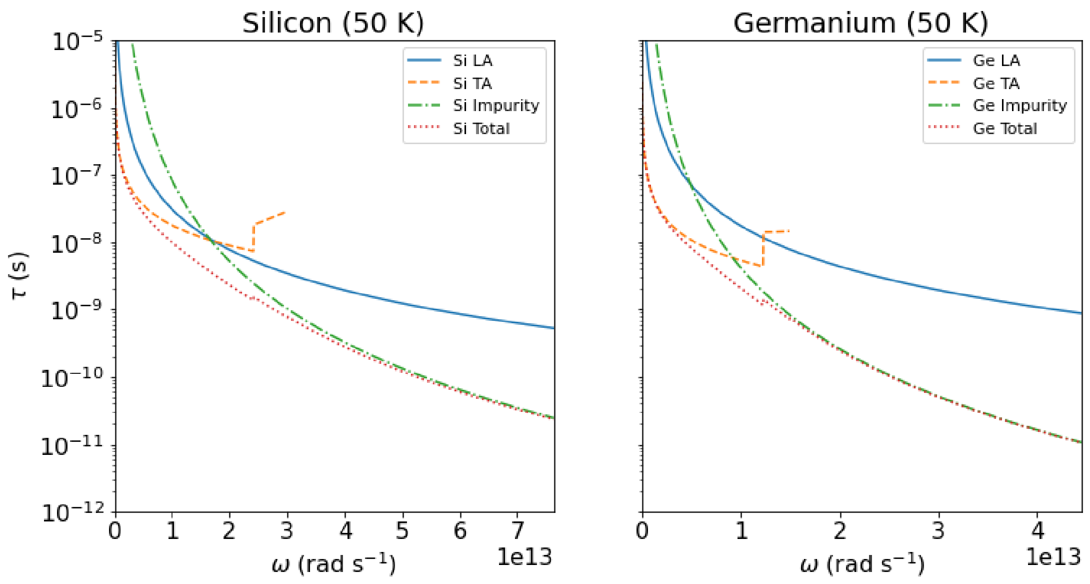

Figure 2.12: Relaxation times for silicon and germanium at 50 K. The rate of impurity scattering primarily drives the total relaxation time.

2.4 The Boltzmann Transport Equation

As discussed in Section 2.3, Fourier's law cannot accurately model nanoscale systems because it is based on the idea that heat travels diffusely through matter. At the nanoscale, this assumption does not hold. Instead, we use the Boltzmann transport equation established from the kinetic theory of gas. The Boltzmann equation describes the physical behaviours of systems in which particles are in motion and interact via collisions, taking into account both diffuse and ballistic transport. These two limiting cases are involved in the transport of phonons. At low temperatures, collision processes are almost non-existent, and ballistic transport dominates. At higher temperatures, the trend is reversed. In all other cases, the two phenomena are present.

Using the Boltzmann Equation, we can formalize the temporal evolution of phonon transport, which propagates ballistically and diffusively within a structure.

$$\frac{\partial f}{\partial t} + \nabla_{\mathbf{K}}\omega \cdot \nabla_{\mathbf{r}} f + \mathbf{F} \cdot \nabla_{\mathbf{j}} f = \left.\frac{\partial f}{\partial t}\right|_{collision} \tag{2.23}$$

The term $\mathbf{F} \cdot \nabla_{\mathbf{j}} f$ represents the action of external forces on the particles resulting from a field outside. In this study, external forces are not accounted for, and since phonons have a zero charge state, the Boltzmann transport equation can be written as:

$$\frac{\partial f}{\partial t} + \nabla_{\mathbf{K}}\omega \cdot \nabla_{\mathbf{r}} f = \left.\frac{\partial f}{\partial t}\right|_{collision} \tag{2.24}$$

This equation is related to the variation of the distribution function $f(t, \mathbf{r}, \mathbf{K})$ which depends on time t, location $\mathbf{r}$, and wave vector $\mathbf{K}$. Hence

- $\frac{\partial f}{\partial t}$ describes the temporal evolution of the distribution function f
- $\nabla_{\mathbf{K}}\omega \cdot \nabla_{\mathbf{r}} f$ describes the movement of phonons through the system. The phonon group velocity is $\nabla_{\mathbf{K}}\omega = \mathbf{v}_g$.

The second term, known as the 'collision term,' makes it possible to account for the diffuse/collision processes. For phonons, the exact form of the collision term is challenging to express [39, 40]. One of the most common approximation methods used to simplify the collision term in the Boltzmann transport equation can be found within the framework of the relaxation time approximation [41].

The relaxation time approximation is equivalent to expressing the collision term as a deviation of the distribution function from its equilibrium during an average time which corresponds to the overall relaxation time of the system. Accomplishing this consists of performing a first-order Taylor expansion on the collision term. Thus, if the distribution function of phonon f is only slightly different from the equilibrium distribution function and if the relaxation time is short enough, the collision term can be expressed as:

$$\left.\frac{\partial f}{\partial t}\right|_{collision} \simeq \frac{f - f_0}{\tau(\omega, p, T)} \tag{2.25}$$

Here, f_0 is the equilibrium distribution function, τ is the overall scattering time with ω being the phonon radial frequency, p the phonon polarization and T is the temperature. This approximation essentially linearizes the collision/scattering term of the Boltzmann transport equation. It implies that whenever a system is not in equilibrium, the collision term will restore to equilibrium following an exponential decay law:

$$f - f_0 = e^{-t/\tau} \tag{2.26}$$

The phonon Boltzmann transport equation now has the following final form:

$$\frac{\partial f}{\partial t} + \mathbf{v}_g \cdot \nabla_{\mathbf{r}} f = \frac{f - f_0}{\tau(\omega, p, T)} \tag{2.27}$$

However, even with the relaxation time approximation, this equation remains challenging to solve in the case of phonon transport, and the exact resolution of the Boltzmann transport equation is currently one of the open research subjects in mathematics [42]. In Chapter 3, a Monte Carlo solution method for the phonon Boltzmann transport equation is presented.

Chapter 3

The Phonon Monte Carlo Method

This chapter presents a Monte Carlo method as a solution to the Boltzmann transport equation. Using silicon and germanium nanostructures, the solution is then validated based on conductivity calculations and temperature profile comparisons. A step-by-step description details the modifications made to existing Monte Carlo methods.

The first section of this chapter summarizes previous works on this topic and describes the incremental improvements made over the years. The following section provides a general description of the Monte Carlo method and some advantages of using a Monte Carlo method as a numerical solution in the context of the phonon Boltzmann transport equation. Sections 3.4 and 3.5 provide an in-depth account of implementing the phonon Monte Carlo method using two different strategies. Section 3.6 describes a 'system evolution' algorithm which synergizes with the approach described in Section 3.5. The remaining sections justify using specific techniques introduced in the previous sections.

3.1 Previous Works

In the 1980s and 90s, there was tremendous advancement in the development of numerical solution techniques for the Boltzmann transport equation for charge carriers like electrons [43–47]. However, phonon transport was largely neglected. Majumdar *et* *al.* [22], Chen and Tien [48], Goodson [49], and Chen [50] presented solution strategies for both the diffusion and the ballistic limit using several simplifying assumptions.

- Dispersion effects were not considered.
- A single-polarization branch was used rather than accounting for the dual-polarization of phonon propagation.

Even with these simplifications, the developed strategies were limited to elementary geometries. This restriction is because it is challenging to solve the Boltzmann transport equation for phonons using a deterministic approach if there are no restrictions on the geometry. The number of independent variables is vast and would render any discretization scheme too complex to be practical. Additionally, accounting for the nonlinear scattering events without an overall relaxation time approximation would be exceedingly challenging as individual scattering events cannot be treated in isolation. An alternative is to solve the Boltzmann transport equation using stochastic or Monte Carlo techniques. Monte Carlo techniques have been used with success for electron transport simulations, and they have also been used for phonon transport by Peterson [51]. Peterson's work in 1994 assumes the linear Debye theory and the same simplifying assumptions mentioned above.

In 2001, Mazumder and Majumdar [1] managed to account for the complete dispersion relations in the acoustic branches. Lacroix, Joulain and Lemonnier further generalized the model by incorporating the Normal and Umklapp scattering processes and accounted for the scattering rates when re-sampling phonons during the energy conservation step [3].

Since then, many improvements have been made, especially concerning taking the collision time into account [52]. In 2004, Narumanchi [33] proposed a resolution using a finite volume method and considered both the acoustic and optical modes using a zero group velocity for the latter. Wang [53] proposed a model free from this constraint in 2007. In 2009, Terris [54] studied the influence of dispersion relations on thermal conductivity, then in 2011, Minnich [55] proposed an elegant method to approximate interface interactions between differing materials.

Peraud and Hadjiconstantinou [4], also in 2011, developed an energy-based variance-reduced formulation that samples phonons based on energy considerations. This technique has the significant benefit of avoiding the ad-hoc energy conservation scheme required in earlier models. In addition, the introduction of an equilibrium temperature significantly decreases simulation time while simultaneously reducing the variance of the results.

3.2 The Monte Carlo Method

Fundamentally, a Monte Carlo method is a numerical solution approach that uses extensive random sampling; however, the term Monte Carlo is widely used to refer to a comprehensive class of computational methods, and Monte Carlo methods can take many forms and are used in all areas of science and engineering. Their simplicity and tendency to preserve an intuitive connection to the underlying physics of the problem are probable reasons for their widespread use [56]. Monte Carlo methods also naturally lend themselves to solutions by simulation rather than numerical discretization, which can be taken advantage of in the modern age of computing. Although traditionally associated with and usually presented in the context of integration, Monte Carlo methods are also used in other fields, such as atomistic modelling, solving partial differential equations, optimization problems, and finance [56]. Another particularly relevant field where Monte Carlo methods are used is statistical physics. Typically, problems in this domain exhibit high dimensionality, making approaches based on discretization inefficient and intractable in many cases. An example is the Metropolis algorithm [57] and its variants [58], which can be used for simulating the various statistical mechanical ensembles.

This study uses a Monte Carlo method to solve the Boltzmann transport equation. Among the available analytical and numerical solutions, Monte Carlo simulations are among the most flexible and accurate [15]. Typically, researchers favour the Monte Carlo approach over other techniques because it is well suited for simulating complicated geometries and adequately accounting for the phonon dispersion relation and different polarization branches [1, 3, 59]. Random numbers determine nearly all the state variables in the system. These random numbers must be independent, or the method may not be reliable [60]. The precision of the results is, up to a point, dependent on the number of simulated phonons. There is a trade-off between accuracy and speed, as simulating higher quantities of phonons takes more computational time. However, with modern CPUs working in parallel and algorithmic improvements, notably the idea of a reference temperature [15] or an equilibrium temperature [4], the Monte Carlo solution can be exceptionally fast and accurate with virtually no memory footprint even when simulating reasonably complex nanostructures. For example, the code used in this study can produce low variance results on a 1 μm kinked nanowire in under 10 seconds.

Monte Carlo methods are well suited for simulating any geometry, including nanofilms [3], nanowires, [61], porous structures [62], or electron-phonon couplings in solid silicon [63]. Sampling of phonons is mainly done via in frequency [3] or in energy [4, 27] but can also be done via wave vector [64] or in mean free path [65]. Some alternate approaches to the Monte Carlo method include ab initio [66, 67] and molecular dynamics [68, 69] methods. Ab initio methods constitute a fundamental approach to describing atomic nuclei from the bottom up by solving the non-relativistic Schrodinger equation for all the forces and constituent nucleons between them. Molecular dynamics approaches seek to analyze the physical movements of atoms and molecules in a system for a fixed time. These techniques do not require prior knowledge of the relaxation or dispersion relations; in some cases, these properties can be determined from these methods. However, a disadvantage is that these methods do not allow us to model structures with large dimensions and remain confined to a few atomic layers for the ab-initio methods and a few million atoms for molecular dynamics simulations.

Convergence Speed

The time taken to perform a Monte Carlo method is contingent on how many random numbers need to be generated. This number is related to the precision sought on the variables that will be modelled. If a particular quantity is associated with a significant standard deviation, it will necessitate drawing a large pool of random numbers to produce an accurate representation.

For the phonon Monte Carlo simulation, the precision of the temperature (or energy) parameter will be greater than that of the heat flux calculation, given an equal amount of random number draws.

All random numbers are taken from a uniform distribution over $[0, 1]$. The variables represented by random numbers in the context of resolving the Boltzmann transport equation are as follows:

- The initial position of the phonons requires three random numbers, although it is possible only to require two random numbers if some constraints are imposed on the system, as is done in this work. See Chapter 4 and Section 5.3.
- The propagation direction of the phonons requires two random numbers.
- Determination of phonon frequency requires a single random number, although using two random numbers allows for a more continuous frequency distribution.
- Determination of phonon polarization requires a single random number.
- The scattering mechanism requires a single random number to determine which scattering process occurs.

- Certain boundary conditions may require a random number depending on the specifications of the simulation geometry.

3.3 Overview

This section provides a general overview of the Monte Carlo method as traditionally described by Mazumber and Majumber [1], Lacroix, Joulain and Lemonnier [3], and others [9, 27, 15]. The method in this study deviates from this approach in several areas described later in the chapter.

Solving the Boltzmann transport equation using the Monte Carlo method involves an initialization phase followed by a series of iterations. Before the algorithm begins, the user must supply the system's geometric specifications. These specifications are the only input required for the simulation code and are given via a JSON file, an example of which can be seen in Appendix B.

The specifications include subdividing the system into smaller cells for measurement purposes and providing the dispersion and relaxation data for the cell material. The number of spatial divisions, or cells, is fixed at the start of the simulation. This mesh is necessary to produce useful measurements of the temperature and flux throughout the system but does not intrinsically affect the simulation mechanics. A system may be specified as a single spatial region, and the simulation will proceed without issue; however, the resulting temperature and flux measurements will be averaged over the entire system, which is unlikely to be helpful. Section 4.2 provides an example of how the meshing affects the simulation outcome, and Section 3.4.8 describes the measurement process and the role of the sensors in more detail. Section 3.4.8 also describes the use of sensor objects to link cells into a single measurement area to decouple the geometric and measurement aspects of the system.

In addition, the user must specify the location of the emitting or isothermal surfaces (see Section 3.4.4). Emitting surfaces represent sections of the system that are kept at a constant temperature through which phonons enter and exit the system. The adiabatic/boundary and transition surfaces, described in Section 3.4.4, are automatically detected by the simulation code and do not require the user to specify their location explicitly.

Figure 3.1 below illustrates many of these details. A linear system has been subdivided into seven cells using six transition surfaces. An emitting surface is established on each end of the system, and the left-hand side surface is set to a higher temperature than the surface on the right-hand side. In this system, each cell has four boundary surfaces that separate the cell from the outside world. This system contains six transition surfaces that act as computational boundaries and divide the system into cells. The simulation code takes a JSON file as input to create these systems, an example of which can be seen in Appendix B. Note the JSON file only requires the user to specify the location of emitting surfaces.

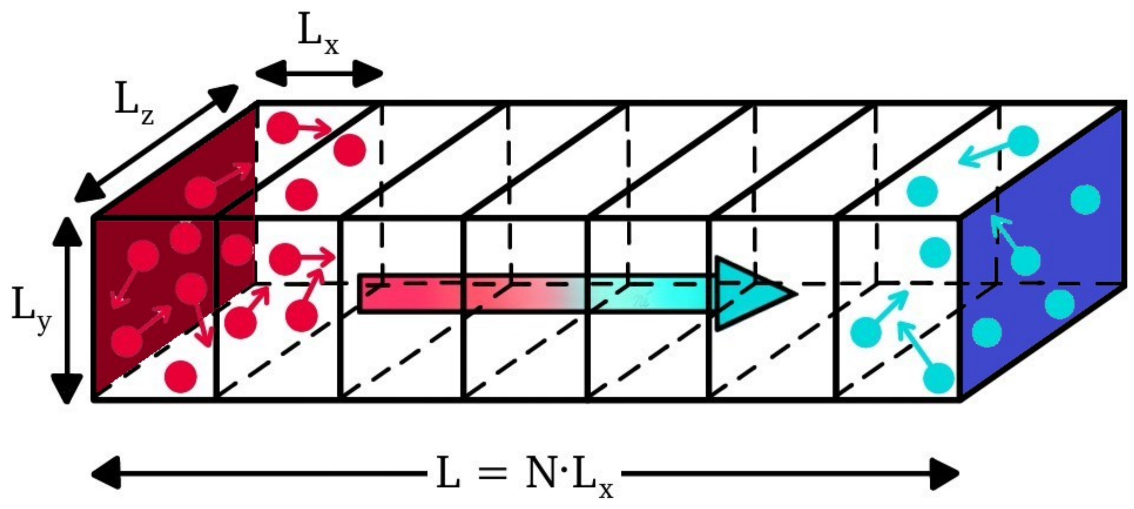

Figure 3.1: A linear geometric system subdivided into seven cells with two emitting surfaces.

Traditionally, the user will also provide total simulation time, and a 'time-step' subdivides the system into smaller time components, each representing an iteration phase. The total number of iterations equals the total simulation time divided by the time-step.

The simulation starts with an initialization phase which consists of populating each cell with the appropriate number of phonons. This number is determined from (3.2), which is discussed in the following section. Then a series of iterations begin, and each iteration involves the following steps.

1. The phonons originating from the emitting surfaces are placed randomly on the emitting surface and are given a frequency, polarization, velocity and direction propagation based on the surface material, the temperature of the surface and the surface's spatial orientation.

2. All the phonons move through the structure, where they can undergo the following interactions.

    - Leave the domain by interacting with an emitting surface.
    - Interact with the system borders.
    - Undergo a scattering or collision process.

3. At the end of each time step, the temperature and flux are calculated in each spatial division. This process is what constitutes taking a measurement.

The iterations end when the simulation time is exceeded. The above steps are depicted in the following flowchart.

3.4 Full Simulation

This work presents two variations of the phonon Monte Carlo method. The first resembles the approach described in Section 3.3 with some distinctions outlined in this section. This form of the simulation is referred to as the full simulation approach

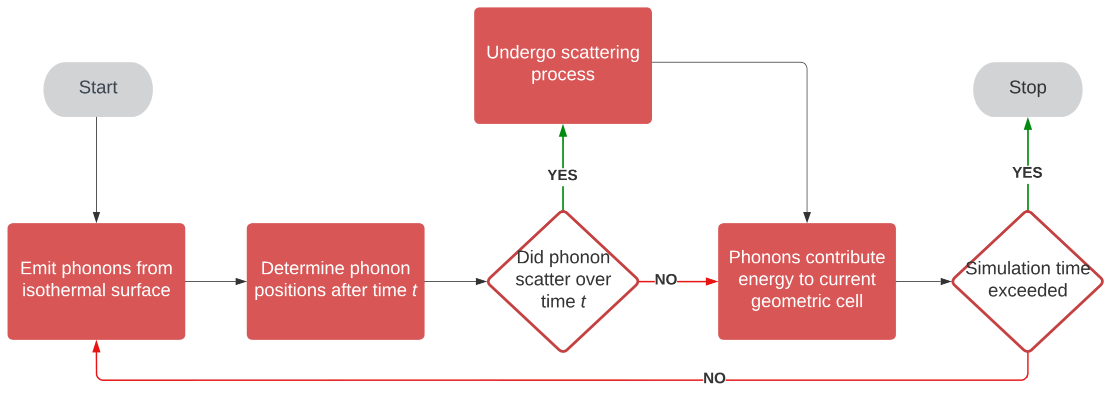
Start
Emit phonons from isothermal surface
Determine phonon positions after time *t*
Did phonon scatter over time *t*
YES
Undergo scattering process
NO
Phonons contribute energy to current geometric cell
Simulation time exceeded
YES
Stop
NO

Figure 3.2: Flowchart of the core steps in each iteration of the phonon Monte Carlo method.

for reasons that will become apparent later in this chapter. In contrast, the deviational simulation closely resembles the work of Peraud and Hadjiconstantinou [70] but, again, with some distinctions.

The rationale behind having a single piece of software that can run two variations of the same method is that the deviational simulation technique is much faster and has much less variance than the full simulation approach. However, the full simulation technique provides more reasonable results when the temperature differences increase or for low-temperature simulations. See Section 4.1.3.

The type of simulation that is run depends on the value of the equilibrium temperature[1]. The user provides this value, and if the user enters an equilibrium temperature of 0, then a full simulation is run. Otherwise, the corresponding equilibrium temperature is used for a deviational simulation.

[1]The notion of an equilibrium temperature is defined in Section 3.5. The equilibrium temperature is the value t_eq that can be seen in Appendix B.

3.4.1 Phonon Creation

The first step of the algorithm is to determine how many phonons should be in each spatial component of the system. From Eq. (2.14), repeated here for convenience, the number of phonons in a volume V is:

$$N = V \sum_p \int_\omega \left[\frac{1}{\exp\left(\frac{\hbar\omega}{k_B T}\right) - 1} \right] \frac{K^2}{2\pi^2 v_{gb,p}} g_p d\omega$$

The theoretically accessible frequencies are continuous over $[0, \omega_{max}]$ where ω_{max} is the maximum frequency present in the material. However, this spectrum must be discretized, and nearly all studies, including this one, use a uniform discretization of $N_b = 1000$ spectral or frequency bins. The bin frequency should not start at 0. Instead, the width of each frequency bin is calculated as follows:

$$\Delta\omega = \frac{\omega_{max}}{N_b} \tag{3.1}$$

The range of the discretization is then $[\Delta\omega, \omega_{max} - \Delta\omega]$ and this interval is comprised of N_b evenly spaced bins of width $\Delta\omega$. Now, the number of phonons in the volume V can be calculated as:

$$N = V \sum_p \sum_{b=1}^{N_b} \left[\frac{1}{\exp\left(\frac{\hbar\omega_{b,p}}{k_B T}\right) - 1} \right] \frac{K^2}{2\pi^2 v_{gb,p}} g_p \Delta\omega \tag{3.2}$$

From Eq. (3.2), the following histograms can be made showing the number of phonons per unit volume in each silicon and germanium frequency bin at 300 K.

These histograms indicate that the number of phonons in a given volume may be substantial. For example, $N \sim 5.5 \times 10^{29}$ phonons/m^3 in silicon at 300 K. Since it is not currently feasible to simulate such a large number of phonons, we either limit ourselves to exceedingly small volumes or a weighting factor is applied to reduce the number of phonons that must be simulated [51]. The idea is that we can group

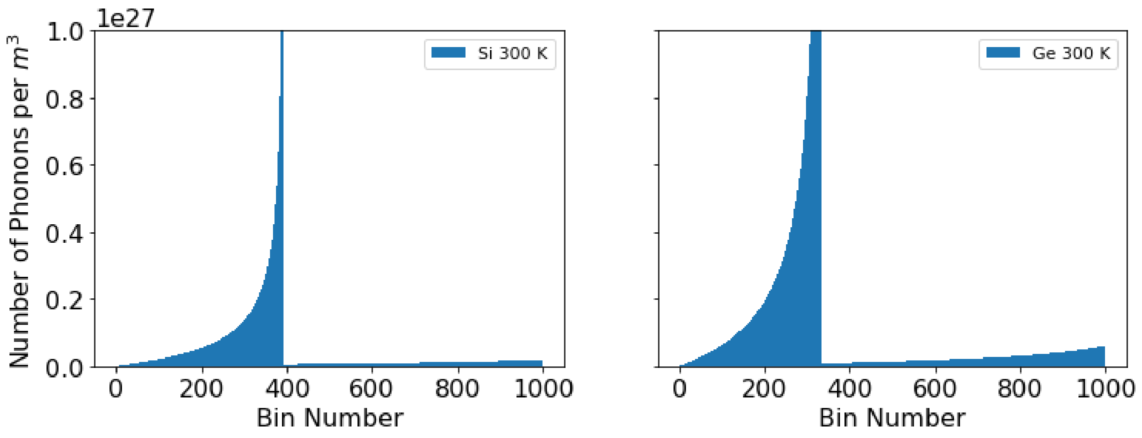

Figure 3.3: Phonons per frequency bin for silicon and germanium at 300 K. The y-axis is intentionally truncated, so the bars of the larger bin numbers are visible.

phonons of equal frequency into phonon packets, and rather than simulate phonons individually, we can simulate a lesser number of these phonon packets. The concept of a phonon packet can be seen in Figure 3.4.

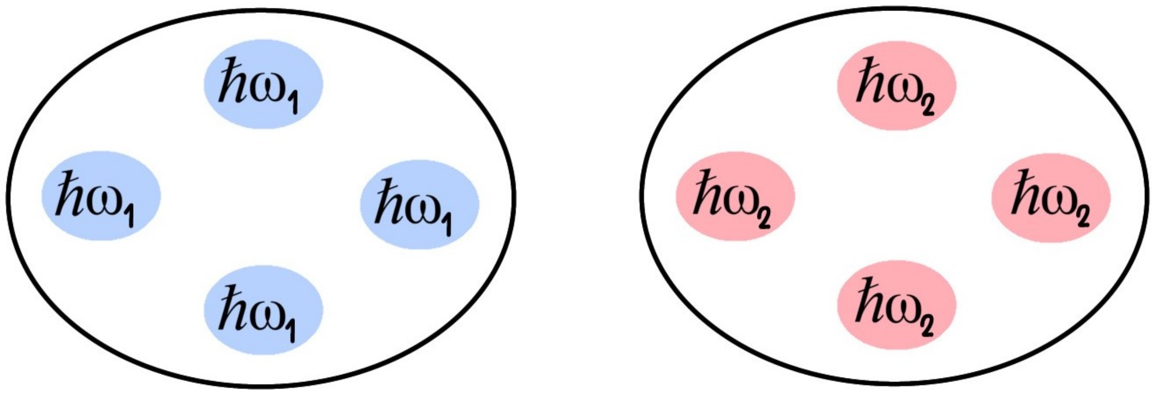

Figure 3.4: Arbitrary fixed number of phonons per packet. Here $N_1 = N_2 = 4$ implying $E_1 \neq E_2$ since $\omega_1 \neq \omega_2$.

Typically, this weighting factor is specified by the user, and the effective number of phonons that must be simulated is then calculated as follows:

$$N_{\text{eff}} = \frac{N}{N_{\text{packet}}} \tag{3.3}$$

where N_{packet} is the user-specified weighting factor. Simulating phonon packets in this manner leads to issues with energy conservation when simulating scattering events. An ad-hoc phonon addition/deletion scheme is employed to rectify the lack of energy conservation, leading to several potential pitfalls. These issues are discussed in Section 3.7. These pitfalls are part of the reason why this work uses the energy-based formalism introduced by Peraud and Hadjiconstantinou [4] for both the full and deviational simulation techniques.

3.4.2 Energy-Based Formulation

The energy-based formalism described here was first introduced by Peraud and Hadjiconstantinou in 2011 [4]. The energy-formalism follows similar steps as outlined in the previous section, except the packets contain phonons with the same frequencies and polarizations so that all the phonons packets have the same energy. This consistency implies the number of phonons per packet changes rather than the energy per packet, as depicted in Figure 3.5.

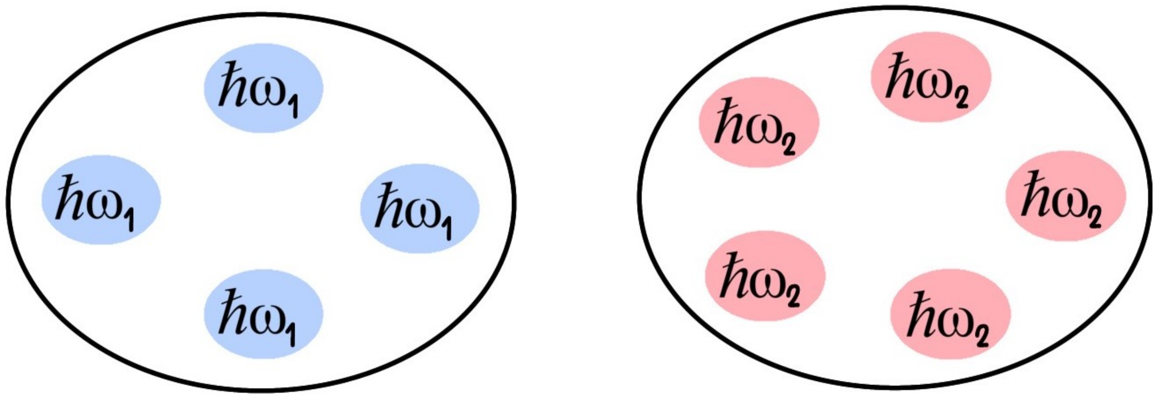

Figure 3.5: Simulated phonon packets with equivalent energy. Here $E_1 = E_2$ implying $\omega_1 = 1.25\omega_2$. The number of phonons in each phonon packet is adjusted to ensure each packet has equivalent energy.

The energy of a given packet can be calculated as $E_{\text{packet}} = n_{\text{phonons}}(\omega)\hbar\omega$, where $n_{\text{phonons}}(\omega)$ is the number of phonons in packets containing phonons of frequency ω. This number must vary with frequency to ensure E_{packet} is always the same.

A packet represents a group of several phonons, but from this point onwards, the distinction between packets and phonons will no longer be made, and the terms phonon(s) and packet(s) will be used interchangeably. The following relation gives the number of simulated packets:

$$N_{tot} = \frac{E}{E_{\text{packet}}} \tag{3.4}$$

where E is given by:

$$E = V \sum_{p} \sum_{b=1}^{N_b} \left[\frac{\hbar\omega_{b,p}}{\exp\left(\frac{\hbar\omega_{b,p}}{k_B T}\right) - 1} \right] \frac{K^2}{2\pi^2 v_{gb,p}} g_p \Delta\omega \tag{3.5}$$

In this formalism, rather than the user fixing the weighting factor N_{packet} the user fixes the total number of phonon packets N_{tot} that will be simulated[2]. If the user chooses $N_{tot} = N$, this would be equivalent to choosing $N_{\text{packet}} = 1$ in Eq. (3.3). Hence, the number of packets N_{tot} greatly impacts the energy distribution's precision. As the number of simulated phonon packets increases, the energy per packet decreases and the simulation becomes more precise but takes longer to execute.

3.4.3 Phonon Initialization

Each phonon must be assigned a frequency ω, a polarization p, and a propagation direction $\mathbf{r} = (x, y, z)$. The group velocity $v_{gb,p}$ is given by the dispersion relations.

Frequency Determination

To assign frequencies to the phonons, it is necessary to first determine the amount of energy in each frequency bin, similar to Figure 3.3. The energy in each frequency bin can be calculated using Eq. (3.5) to produce the following histograms:

[2]This is num_phonons from Appendix B.

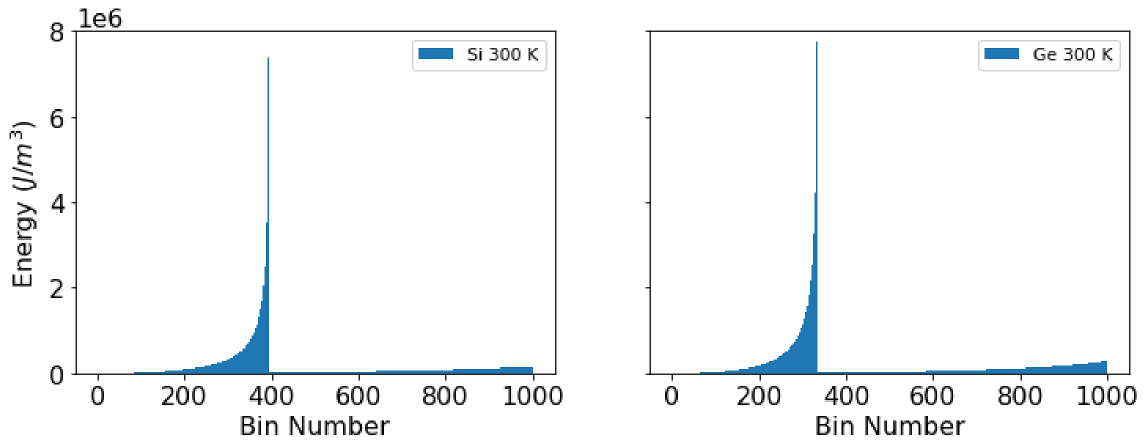

Figure 3.6: Energy per frequency bin for silicon and germanium. The energy distribution is markedly different than the phonon distribution in Figure 3.3.

Next, from the work of Mazumder and Majumdar [1], a cumulative distribution function is constructed by doing the cumulative summations of the energy per unit volume in the ith spectral bin over the energy calculated from Eq. (3.5):

$$F_i(T) = \frac{\sum_{j=1}^{i} E_j(T)}{\sum_{j=1}^{N_b} E_j(T)} \tag{3.6}$$

Using Eq. (3.6), the cumulative energy distribution function can be plotted as shown in Figure 3.7.

To determine the frequency interval associated with a phonon, the first random number[3] R_1 is drawn and knowing that $F_{i-1} \leq R_1 \leq F_i$, the corresponding value F_i can be found using a bisection algorithm, which gives the frequency bin ω_N. To allow for a more continuous distribution, a second random number R_2 can be used to choose a point on the interval associated with ω_N using the following equation:

$$\omega_i = \omega_N + (2R_2 - 1) * \Delta\omega/2 \tag{3.7}$$

[3]All random numbers are taken from a uniform distribution over $[0, 1]$.

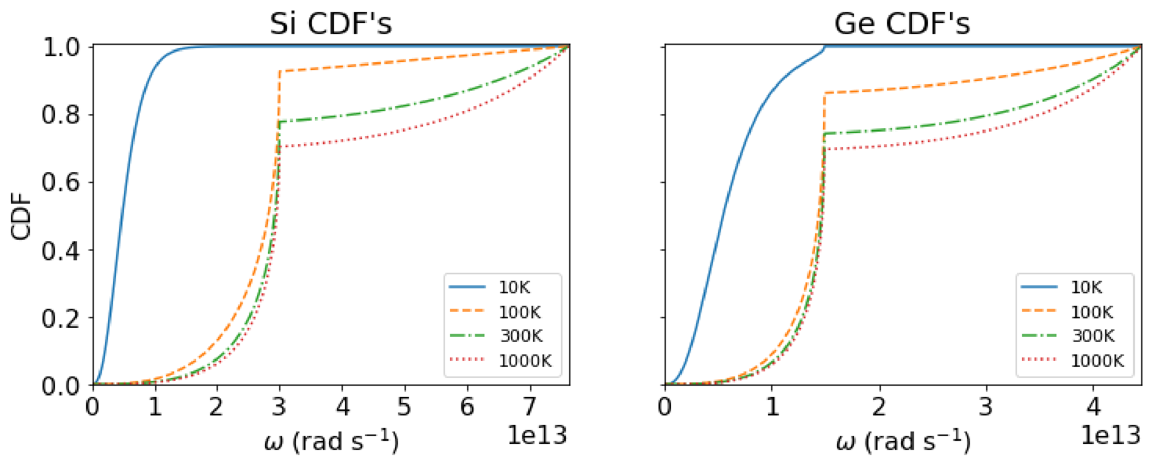

Figure 3.7: Cumulative energy distribution functions for silicon and germanium at various temperatures.

This process is illustrated in Figure 3.8.

Polarization Determination

Once the frequency is known, the polarization of the phonon can be determined in accordance with the material's dispersion curves at a given temperature. By calculating the number of accessible states for each polarization at frequency ω_i, weighted by its energy $\hbar\omega_i$, it is possible to deduce a phonon's probability of a specific polarization. The likelihood for a phonon to have LA polarization, $P_{LA}(\omega_i, T)$, can be calculated as follows:

$$P_{LA}(\omega_i, T) = \frac{E_{LA}(\omega_i, T)}{E_{LA}(\omega_i, T) + E_{TA}(\omega_i, T)} \tag{3.8}$$

If a random number, R_3, is less than P_{LA}, then the polarization is LA; otherwise, the phonon will have TA polarization. This equation only accounts for acoustic phonons. Accounting for optical phonons requires incorporating their energy contributions into the denominator.

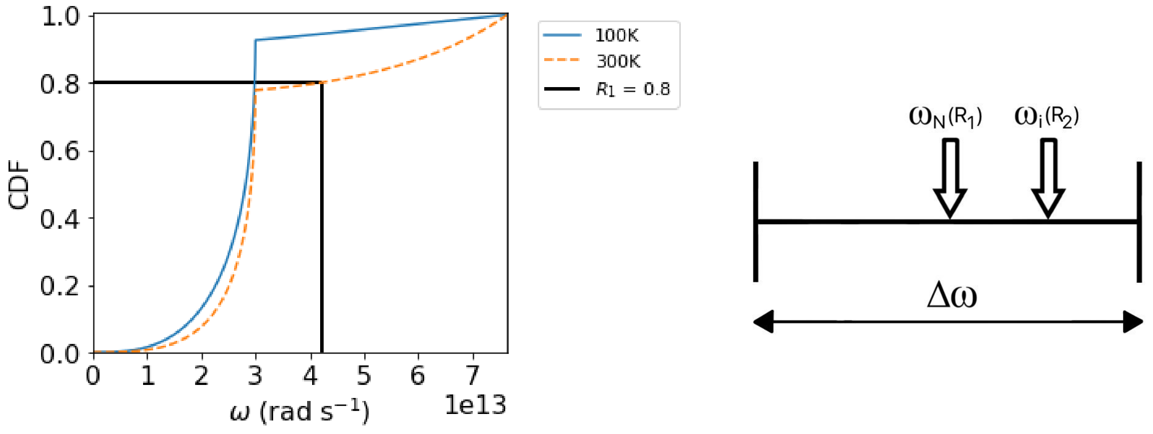

Figure 3.8: Random selection of a phonon from the cumulative energy distribution function. The first random number, R_1, identifies the frequency of bin N. A second random number, R_2, determines the frequency on this interval.

The code used in this work combines the cumulative distribution function (CDF) and polarization probabilities into a single table or array. Table 3.1 is an excerpt of this array for silicon at 300 K.

Table 3.1: Combined CDF and probability table for silicon at 300 K.

Bin #	CDF	Probability of LA phonon
1	1.9565×10^{-9}	0.0830
2	1.9590×10^{-8}	0.0829
...	...	...
394	0.7772	0.0014
395	0.7773	1.0
...	...	...
999	0.9991	1.0
1000	1.0	1.0

Velocity and Propagation Direction

At this stage, the phonon is completely characterized. From the frequency and polarization, it is possible to determine the group velocity of the phonon packet based on the dispersion relation. Inverting the dispersion relations, the group velocity v_i for a phonon of frequency ω_i can be obtained.

$$\omega_i = aK_i^2 + bK_i \tag{3.9}$$

It is straightforward to solve for K_i by using the quadratic formula. The group velocity is then:

$$v_i = \frac{\partial \omega}{\partial K} = 2aK_i + b \tag{3.10}$$

From the isotropy assumption, a phonon is equally likely to travel in any direction regardless of location within the material. The direction component of the velocity vector can thus be determined using two random numbers R_4 and R_5.

$$\begin{cases} d_x = & 2R_4 - 1 \\ d_y = & \sqrt{1-(d_x)^2} \cdot \cos(2\pi R_5) \end{cases} \tag{3.11}$$

Only two direction vectors are used in this work since assumptions make it unnecessary to account for the third dimension explicitly. The third (z) dimension is assumed to be arbitrary or infinite in size, allowing us to ignore phonon movement and surface interactions pertaining to that dimension throughout the simulation. The three assumptions that allow for this are as follows.

- All geometric cells, and the system as a whole, are assumed to have the same measurement specification in the z-dimension.
- No emitting surfaces can be located on the z-dimension surfaces.
- The z-dimension surfaces are perfectly specular.

See Section 5.3 for further discussion on this topic.

3.4.4 Surface Interactions

A phonon can interact with three types of surfaces throughout the simulation. The surfaces include physical borders (boundary and emitting surfaces) and computational boundaries (transition surfaces). Transition surfaces represent imaginary boundaries between the different cells comprising the system.

Emitting Surfaces

Emitting surfaces[4] are boundaries assumed to be at a fixed temperature T_e, and phonons enter and exit the system via these surfaces. When phonons collide with an emitting surface, they are removed from the simulation.

It is assumed that the phonons entering the system from these surfaces come from the equilibrium distribution. Thus, using Eq. (2.17) can be used to ascertain that the total energy emitted by an emitting surface of area A over a period of time Δt is:

$$E' = A\Delta t \int_{2\pi} \left(\sum_p I_{\omega,p} d\omega \right) \cos\theta g_p d\Omega \tag{3.12}$$

$$= \frac{A\Delta t}{4} \sum_p \sum_{b=1}^{N_b} v_{gb,p} \left[\frac{\hbar\omega_{b,p}}{\exp\left(\frac{\hbar\omega_{b,p}}{k_B T_e}\right) - 1} \right] D(\omega_{b,p}, p) g_p \Delta\omega \tag{3.13}$$

Hence, the total number of phonons originating from an emitting surface over time Δt is:

$$N' = \frac{E'}{E_{\text{packet}}} \tag{3.14}$$

The portion of Eq. (3.13) within the summations is the same as the original energy calculation from Eq. (3.5) except it is weighted by the group velocity for each

[4]Several studies refer to these surfaces as isothermal boundaries.

frequency bin and polarization. When phonons originate from an emitting surface, they should be drawn from the velocity-weighted distribution and not the original distribution given by Eq. (3.5). The CDFs for emitted phonons can be seen in Figure 3.9.

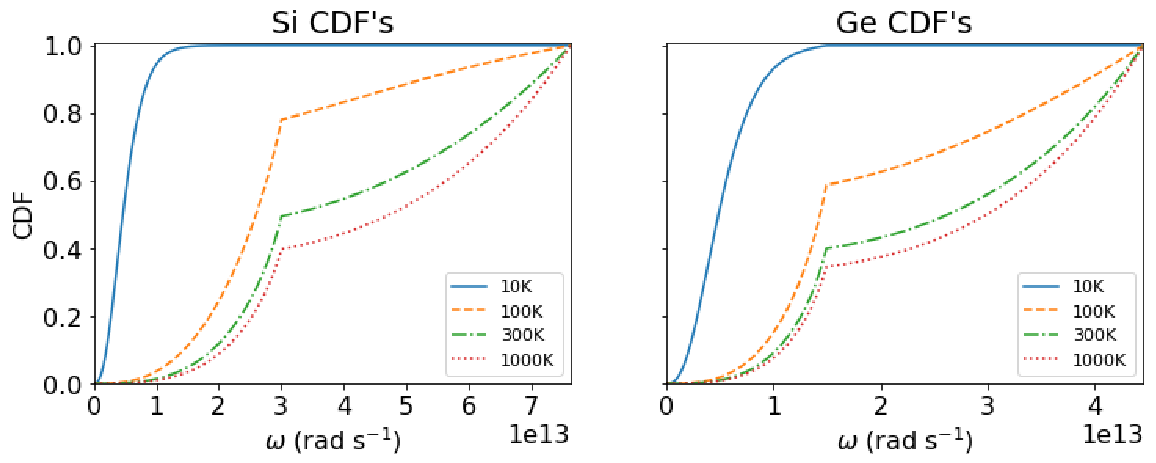

Figure 3.9: Cumulative group velocity weighted energy distribution functions for silicon and germanium at various temperatures.

Table 3.2: Combined group velocity weighted CDF and probability table for silicon at 300 K.

Bin #	CDF	Probability of LA phonon
1	4.4381×10^{-9}	0.1380
2	4.4388×10^{-8}	0.1378
...	...	...
394	0.4947	0.0491
395	0.4956	1.0
...	...	...
999	0.9985	1.0
1000	1.0	1.0

Additionally, phonons originating from emitting surfaces have biased direction vectors based on the surface's normal vector. The biased direction vectors can be deduced from the integrand in Eq. (3.12) as:

$$\begin{cases} d_x = & n_x\sqrt{R_6} - n_y\sqrt{1-R_6}\cos(2\pi R_7) \\ d_y = & n_y\sqrt{R_6} + n_x\sqrt{1-R_6}\cos(2\pi R_7) \end{cases} \tag{3.15}$$

where n_x and n_y are the components of the surface normal vector.

Lacroix, Joulain and Lemonnier [3] introduced an alternative where emitting surfaces are simulated by augmenting the system with sufficiently large boundary cells in which all particles are reinitialized at the beginning of every time-step [3]. This approach has the advantage of being simple to implement for linear systems, but it requires time discretization and would be highly challenging to implement in a structure with complex geometrical aspects. In addition, this approach increases the number of particles in the simulation, which decreases performance.

Boundary Surfaces

Boundary surfaces represent the division of the material from the outside world and are impassable to phonons. Each boundary surface is given a degree of specularity in the range [0, 1]. A surface with a specularity of 1 is considered perfectly reflective. In specular reflection, a phonon collides with a wall, and the phonon's post-collision direction vector is related to the initial direction vector by:

$$\begin{cases} d_{xf} = & n_x(-d_{yi}n_x - d_{yi}n_y) - n_y(-d_{xi}n_y + d_{yi}n_x) \\ d_{yf} = & n_y(-d_{yi}n_x - d_{yi}n_y) + n_x(-d_{xi}n_y + d_{yi}n_x) \end{cases} \tag{3.16}$$

A diffuse reflection randomizes the direction of the phonon isotropically on the unit sphere, and the resulting direction of propagation is given by Eq. (3.15).

The degree of specularity is defined by the user in the input specifications; see Appendix B. Each geometric cell can be given its own specularity value. If the surface specularity is less than 1, a random number R_8 is drawn and compared to the specularity of the surface. If R_8 is less than this value, the phonon is reflected according to Eq. (3.16). Otherwise, the phonon undergoes a diffuse reflection according to Eq. (3.15).

Some studies [71, 72] have proposed a method to assign surface specularity as a function of the phonon wave vector.

Transmission Surfaces

For simulations involving a single material, the transmission surfaces act as computational boundaries to distinguish geometric cells within the system. When a phonon reaches a transmission surface, it will now be associated with a new cell and contributes to the temperature and flux in that part of the system. Additionally, the scattering probabilities will change since the new cell will most likely be at a different temperature.

Each cell typically has its own temperature, flux recordings, and scattering rates. In Section 3.4.8, the notions of measurement areas and the decoupling of the geometric cell from the measurement area are introduced. The idea is to link many geometric cells to the same measurement area, which would allow for the construction of detailed configurations involving many small triangles without imposing severe performance penalties.

Transmission surfaces between two different materials are another matter. In this work, all the results are from systems comprised of a single material, but the simulation code allows the user to easily specify systems containing different materials. However, the interfaces between these materials will act as a transmission surface defined above, with one exception. If the frequency of a phonon passing from one material to another is incompatible with the new material, the phonon will diffusely reflect into the original material.

The details of phonon behaviour at such an interface are not well understood and are the subject of ongoing research [73–75].

3.4.5 Phonon Origination

In this study, spatial discretization is done through the use of triangles. Using triangular cells as the building block of the simulation geometry allows for great flexibility in the types of geometries that can be simulated. To randomly place a phonon within a two-dimensional triangle defined by the points p_1, p_2, and p_3, two random numbers are required [76][5]:

$$\begin{cases} x_i = & (1-\sqrt{R_9})\cdot p_{1x} + \sqrt{R_9}(1-R_{10})\cdot p_{2x} + \sqrt{R_9}R_{10}\cdot p_{3x} \\ y_i = & (1-\sqrt{R_9})\cdot p_{1y} + \sqrt{R_9}(1-R_{10})\cdot p_{2y} + \sqrt{R_9}R_{10}\cdot p_{3y} \end{cases} \tag{3.17}$$

The position, energy and velocity of each phonon can now be determined, and the boundary conditions of the simulation are specified. The next step is determining the total amount of energy within the system for the simulation duration. This is accomplished by summing the energy contributions of each emitting surface using Eq. (3.13) and the energy that is initially in the system using Eq. (3.5). Once the total energy in the system is known, E_{packet} can be determined by dividing the total system energy by the number of phonons the user has specified in the simulation input.

[5]This approach may seem computationally intensive, but $\sqrt{R_9}$ only needs to be evaluated once. A better-performing algorithm exists but requires a conditional statement, making it harder to express here. Ultimately, the performance of this algorithm is not of great concern as it is typically called much less than once per phonon and perhaps never in this case of deviational simulations.

Phonons originating in cells are given a random position within the triangular cell using Eq. (3.17), and the direction of propagation is given by Eq. (3.11). The phonon should also be given a time to expiry equal to the simulation time specified by the user. This expiry value will be used in the drifting phase described in Section 3.4.6.

Phonons originating from emitting surfaces are given a random position on the surface. Eq. (3.15) gives the direction of propagation. The time to expiry of these phonons should be a uniform random number on the interval $[0, t_{\text{final}}]$, where t_{final} is the simulation time specified by the user. This reflects the idea that phonons from emitting surfaces continuously enter the system throughout the simulation.

Whether a phonon originates in a cell or on the surface, it will follow the same life cycle described in the following sections.

3.4.6 Drifting

Phonons undergo the process of drifting, scattering and contributing their energy until the simulation time is exceeded, or the phonon leaves the system by colliding with an emitting surface.

Before describing the drifting algorithm, two important timings must be introduced. The first is the time until the next scattering event, which can be calculated as follows:

$$\Delta t_s = -\tau(\omega, p, T) \ln(R_{11}) \tag{3.18}$$

If a phonon originates on a surface, the relaxation rate calculation should use the temperature of the cell or measurement area in which that surface resides, not the surface temperature. The distinction between a measurement area and a cell is explained in Section 3.4.8. Typically, there is no distinction, and the geometric aspects of a cell are coupled with the physics governing scattering rates and temperature/flux calculations

The other timing event is when the subsequent measurement occurs. A measurement event is when the phonon contributes energy and flux to the spatial region it occupies. As input, the user provides the total simulation time and the number of measurements that should occur throughout the simulation. From here, it is straightforward to deduce the time interval on which measurement events should take place. If we let Δt_m be the time until the next measurement event takes place, then the algorithm for drifting a phonon between $t = t_{\text{begin}}$ and $t = t_{\text{final}}$ is as follows:

1. Set the minimum time Δt_{min} to the smaller of Δt_s and Δt_m.
2. Determine if the phonon will undergo a surface collision over the time interval Δt_{min} based on the phonon speed and direction of propagation.
3. If there is a surface collision, it can be one of three possibilities.
   - **Boundary surface:** Update the phonon position to be the point where the collision occurs. Alter the phonons direction of propagation based on the interaction with the surface using Eq. (3.15) or Eq. (3.16). Adjust Δt_{min} by subtracting the amount of time taken for the phonon to impact the surface. Go back to step 2.
   - **Emitting surface:** Flag that the phonon has left the system and end the algorithm for the current phonon.
   - **Transmission surface:** There are two possibilities, and in either case, the transmission surface interaction should update the phonon to reflect that the phonon has transitioned to a new geometric cell.
     - If the collision occurs with a transition surface between two cells within the same measurement area, place the phonon at the point of collision and adjust Δt_{min} subtracting the amount of time taken for the phonon to reach the transition surface. Go back to step 2.

  - If the transmission surface is between two cells in different measurement areas, a new scattering time must be calculated. Place the phonon at the point of collision and adjust Δt_m by subtracting the time taken for the phonon to reach the transition surface. Calculate a new scattering time, Δt_s, based on the temperature of the new measurement area using Eq. (3.18). Go back to step 1.

4. If no surface collision occurs within the time interval Δt_{min}, adjust the phonons position based on its velocity and direction of propagation for the period Δt_{min}. Update Δt_s and Δt_m accordingly. One of these values will be 0, indicating that either a measurement or scattering event occurs at the phonons' new position.

   - **Scattering event:** Undergo the scattering procedure described in Section 3.4.7. Find a new Δt_s using Eq. (3.18). Go back to step 1.
   - **Measurement event:** Undergo a measurement event as described in Section 3.4.8. Find the time until the next measurement event and set it to Δt_m. If the phonon has reached the end of the simulation, that is, if $\Delta t_m = \Delta t_{\text{final}}$, terminate the algorithm for the existing phonon. Otherwise, go back to step 1.

It is vital to recall that while $t_{\text{begin}} = 0$ for all phonons originating in cells, this number can vary for phonons which originate via emitting surfaces which complicates finding the initial value of Δt_m. For phonons originating via emitting surfaces, t_{begin} should be set to a random number from a uniform distribution over $[0,\ t_{\text{final}}]$ which is equivalent to $R_{12} * t_{\text{final}}$.

The three functions primarily responsible for this algorithm can be found in Appendix A (A.1, A.2, A.3).

3.4.7 Scattering

There are three possible scattering mechanisms a phonon may undergo. In both N and U-processes, the phonon frequency, polarization and group velocity must be re-sampled. For U-process and impurity scattering, a new propagation direction must be chosen using Eq. (3.11).

The original cumulative distribution function, Eq. (3.6), is not used when re-sampling the phonon properties. Re-sampling scattered phonons from the original distribution will cause the phonon distribution in the system to drift away from the original distribution. This drifting is because the probability of a phonon scattering is proportional to $\mathcal{T}^{-1}$, which means phonons of specific frequencies and polarizations are more apt to be scattered. If we draw the re-sampled from the original distribution, these phonons will be disproportionately removed from the system. To account for this, the CDF is scaled by the factor $\mathcal{T}^{-1}$ [3]. The new cumulative distribution function can be calculated as follows:

$$F_i^+(T) = \frac{\sum\limits_{j=1}^{i} E_j^+(T)}{\sum\limits_{j=1}^{N_b} E_j^+(T)} \tag{3.19}$$

where

$$E^+ = V \sum_p \sum_{b=1}^{N_b} \mathcal{T}(\omega_{b,p}, p, T)^{-1} \left[\frac{\hbar\omega_{b,p}}{\exp\left(\frac{\hbar\omega_{b,p}}{k_B T}\right) - 1} \right] D(\omega_{b,p}, p) g_p \Delta\omega \tag{3.20}$$

The weighted cumulative distribution function for silicon at 300 K can be seen in Figure 3.10.

To determine which scattering mechanism occurs, a random number, R_{12} is drawn, and if:

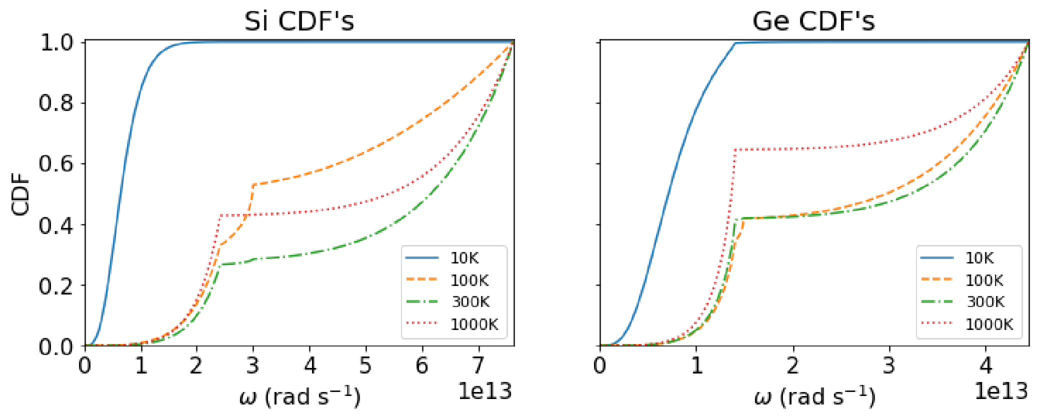

Figure 3.10: Cumulative relaxation time-weighted energy distribution functions for silicon and germanium at various temperatures.

$$R_{12} <= \frac{\mathcal{T}_N(\omega,p,T)^{-1} + \mathcal{T}_U(\omega,p,T)^{-1}}{\mathcal{T}(\omega,p,T)^{-1}} \tag{3.21}$$

then either an N-process or a U-process has occurred. The phonon frequency, velocity and polarization are re-sampled via the same process described in Section 3.4.3 using the CDF in Figure 3.10. If Eq. (3.21) is true, then the following condition is evaluated:

$$R_{12} > \frac{\mathcal{T}_N(\omega,p,T)^{-1}}{\mathcal{T}(\omega,p,T)^{-1}} \tag{3.22}$$

If Eq. (3.22) is true, then an Umklapp scattering event has occurred, and the phonon propagation direction is updated using Eq. (3.11). Finally, if Eq. (3.21) is false and both the following conditions are true:

$$\begin{cases} R_{12} > \frac{\mathcal{T}_N(\omega,p,T)^{-1}+\mathcal{T}_U(\omega,p,T)^{-1}}{\mathcal{T}(\omega,p,T)^{-1}} \\ \mathcal{T}_I(\omega,p,T)^{-1} > 0 \end{cases} \tag{3.23}$$

This means an impurity scattering event has occurred, and the phonon direction propagation is re-sampled using Eq. (3.11), but the properties are not re-sampled. Implementation code can be found in Appendix A.4.

Table 3.3: Combined relaxation time-weighted CDF and probability table for silicon at 300 K.

Bin #	CDF	Probability of LA phonon
1	5.9755×10^{-12}	3.3294×10^{-5}
2	1.6758×10^{-10}	9.9701×10^{-5}
...	...	...
394	0.2842	0.0414
395	0.2843	1.0
...	...	...
999	0.9958	1.0
1000	1.0	1.0

3.4.8 Measurements

Before discussing how phonon contributions are measured, the distinction between a cell and a measurement area will first be addressed. When creating the simulation geometry, the user must assign each geometric cell a sensor ID. Cells can only be attached to a single sensor, but a sensor can be associated with many cells. All the cells associated with a single sensor represent a measurement area. The cells control the geometric aspects, but the sensor handles the physics related to temperature/flux calculations and relaxation rates for each attached cell.

The distinction between cells and sensors/measurement areas is made because triangles are used as the fundamental building blocks of the simulation models. This decoupling allows the user to create intricate designs using many small triangles. If each of these triangles required its own scattering table, see Table 3.3, to handle relaxation times, this could eventually lead to performance issues. This is especially true for transient[6] simulations which require each cell to have its own scattering table for each measurement step. For the kinked wire simulations in Section 4.2, this would mean approximately 3000 × 5000 × 2000 doubles requiring storage throughout the simulation, and 3000 × 5000 tables would need to be updated at the end of each iteration.

The process for taking a measurement is straightforward. When a cell reaches this point in the algorithm from Section 3.4.6, the phonon contributes an energy unit to the measurement area/sensor attached to the geometric cell where the phonon currently resides. The current lifetime of the phonon must be taken into consideration here. At this stage, it is unnecessary to store the value of E_{packet}; simply increment an integer counting variable to record the phonon's energy contribution.

To record the phonon's flux contribution, the phonon's velocity vx and vy are calculated and accumulated using:

$$v_x = v_{gb,p} \cdot d_x \tag{3.24}$$

$$v_y = v_{gb,p} \cdot d_y \tag{3.25}$$

Implementation code can be found in Appendix A.5.

3.4.9 Temperature Calculations

Once all the phonons have been simulated, each measurement area will contain an integer value representing the number of energy packets the phonons have contributed. Let $\mathcal{N}_j$ be the number of energy packets inside measurement area j. To obtain the amount of energy in the measurement area at each measurement step in the simula-

[6]The software can simulate transient heat effects as described by Peraudand Hadjiconstantinou in [70], but this thesis does not go into the topic.

tion, the following calculation is used:

$$E_j = E_{\text{packet}} \mathcal{N}_j \tag{3.26}$$

Once E_j is known, the corresponding temperature T_j is calculated by numerically inverting the following expression:

$$E_j = V \sum_p \sum_{b=1}^{N_b} \left[\frac{\hbar\omega_{b,p}}{\exp\left(\frac{\hbar\omega_{b,p}}{k_B T_j}\right) - 1} \right] D(\omega_{b,p}, p) g_p \Delta\omega \tag{3.27}$$

At this point, the temperature of each measurement area is known at times equivalent to when each measurement in the system was taken.

To find the flux in measurement area j, let $\mathcal{V}x_j$ and $\mathcal{V}y_j$ be the sum of the phonon velocity contributions from Eq. (3.24) and Eq. (3.25) respectively. The flux in each measurement area is then:

$$\phi_x = \frac{\mathcal{V}x \cdot E_{\text{packet}}}{V}$$
$$\phi_y = \frac{\mathcal{V}y \cdot E_{\text{packet}}}{V}$$

where V_j is the combined volume of all the cells making up the measurement area. Implementation code can be found in Appendix A.6.

3.5 Deviational Simulation

This section is comprised of two key components. The deviational formalism introduced by Peraud and Hadjiconstantinou [4] and the linearization approach by the same authors [70].

The deviational simulation presented here is similar to the full simulation method described in Section 3.4, but the deviational simulation incorporates the concept of an equilibrium temperature and the linearization of the collision operator in the Boltzmann transport equation. However, unlike the paper [70] in which this linearization approach is introduced, local temperatures are still maintained and updated rather than using the equilibrium temperature for all scattering and temperature calculations. This, in addition to the evolution algorithm from Section 3.6, are the primary distinctions between the deviational simulation used in this work and that described by Peraud and Hadjiconstantinou in [70].

It should be noted that the deviational formalism presented by Peraud and Hadjiconstantinou [4] and the linearization approach by the same authors [70] are independent approaches. In this work, the deviational simulations use both the deviational formalism in conjunction with the linearization of the Boltzmann transport equation, whereas full simulations use neither.

The user specifies which type of simulation to run by their choice of equilibrium temperature for the simulation input. A full simulation is run if the user enters an equilibrium temperature of 0. Otherwise, a deviational simulation is used with the requested equilibrium temperature. The notion of an equilibrium temperature is explained in Section 3.5.1.

3.5.1 Equilibrium Temperature and Deviational Particles

The theoretical and statistical details behind the deviational formulation can be found in [4]. The four significant changes this paper introduces, compared to the full simulation technique discussed previously, are:

1. An equilibrium state at temperature T_{eq} from which deviations will be simulated.

2. Newly defined computational particles are referred to as deviational particles.

3. An adjusted distribution function that incorporates T_{eq}.

4. Requiring a second numerical inversion that involves frequency and polarization-dependent scattering rates.

This section describes the adjustments required to incorporate the equilibrium temperature and the deviational particles. The adjusted distribution function and second numerical inversion[7] are not used because the linearization process described in Section 3.5.2, introduces alternatives.

Equilibrium Temperature

The introduction of an equilibrium temperature T_{eq} into the simulation stems from a more general class of control-variate variance reduction methods for solving kinetic equations, which provides significant computational savings and variance reduction [4]. The reduction in statistical noise allows us to simulate systems with tiny temperature differences more accurately, see Section 4.1.3. The methodology behind the deviational formalism is to only solve for the deviation from an equilibrium distribution while the contribution of the equilibrium is added deterministically to the properties of interest [77]. The equilibrium temperature specifies the temperature of this equilibrium distribution.

Minimal changes are required to incorporate an equilibrium temperature into the existing simulation framework; the user must specify an equilibrium temperature greater than 0 in the simulation input file. The initial value of the equilibrium temperature can be challenging to ascertain, especially in systems with complex geometries and several emitting surfaces. However, the algorithm described in Section 3.6

[7]Requiring this second numerical inversion significantly complicates the process of taking energy measurements since it is no longer possible to accumulate energy packets with a single integer variable. The second numerical inversion requires frequency and polarization-dependent scattering rates. Thus each energy packet must be associated with the frequency and polarization of the contributing phonons. There is also the performance impact of required two numerical inversions, but this is generally not an issue unless a system has a significant number of measurement areas.

ensures that as long as the initial input for T_{eq} is reasonable[8], the system will evolve to a stable equilibrium temperature.

Deviational Particles

The difference between a deviational phonon and the phonons previously described is that deviational phonons have a sign, either $+1$ or -1, associated with them. Phonons with a negative sign have a cooling effect and will reduce the energy in a measurement area, whereas those with a positive sign will act the same way as they did in the full simulation. The implementation of this measurement process is unchanged from that described in the full simulation technique and can be seen in Appendix A.5.

The determination of a phonon's sign is straightforward. Using the initial temperature T_{init} of each cell, then if a phonon originates in a cell where $T_{\text{init}} > T_{eq}$ or on a surface where $T_e > T_{eq}$, then the phonon is a given a sign of $+1$. Otherwise, the phonon is given a sign of -1. Note that for full simulations, all phonons are given a sign of $+1$ since, by definition, $T_{eq} = 0$ for full simulations. This is why the measurement process described in Section 3.4.8 is unchanged if a sign variable is incorporated in the full simulation method.

3.5.2 Linearized Boltzmann Equation

The governing Boltzmann transport equation may be linearized for the minor deviations from equilibrium encountered, incurring only a small, second-order error. This linearization may lead to a simpler and more efficient simulation method that introduces no approximation [77]. The derivation of the linearized Boltzmann equation can be found in [70] and for a more thorough discussion in [77]. As it pertains to this

[8]Reasonable means the input value for T_{eq} is within the interval $[T_C, T_H]$ where T_H is the warmest emitting surface in the system, and T_C is the coolest emitting surface in the system.

study, the primary benefit of this process is avoiding the double inversion required to calculate the temperature and pseudo temperature.

Another benefit to the linearization is that, under certain circumstances, the equilibrium temperature can be used throughout the simulation domain to calculate scattering rates and temperature calculations as described in Section 3.5.4. While this theoretically means spatial discretization is no longer required, some spatial discretization is still necessary to make meaningful measurements through the simulation. This study maintains local temperatures within these measurement areas.

Analysis using a global equilibrium temperature suggests that the error resulting from linearizing the Boltzmann equation is still acceptable ($< 3\%$) up to temperature differences of 30 K when compared to results where the Boltzmann equation is not linearized, and cells maintain their local temperatures [77]. In Section 4.1.3, it is shown that maintaining local temperatures allows deviational simulations using the linearized Boltzmann equation to produce nearly identical results to the full simulation at a temperature difference of 60 K. However, a global equilibrium temperature would greatly simplify and expedite transient simulations with small temperature differentials.

To derive the linearized Boltzmann equation under the energy-method formalism, it is assumed the quantity $T - T_{eq}$ is small such that the collision operator is linearized as follows [70]:

$$\frac{e - e^{eq}}{\tau(\omega, p, T)} \approx \frac{T - T_{eq}}{\tau(\omega, p, T)} \frac{de^{eq}}{dT} \tag{3.28}$$

where T denotes the local temperature. In practice, accounting for this linearization is relatively straightforward. After some simplifications, the modified Bose-Einstein distribution function that was introduced in the energy-based formalism is replaced by its derivative:

$$\frac{de(\omega)}{dT} = \frac{(\hbar\omega)^2}{k_b T^2} \frac{\exp\left(\frac{\hbar\omega}{k_b T}\right)}{\left(\exp\left(\frac{\hbar\omega}{k_b T}\right) - 1\right)^2} \tag{3.29}$$

In every case where the modified Bose-Einstein distribution $\frac{\hbar\omega}{\exp\left(\frac{\hbar\omega}{k_B T}\right)-1}$ is used, its derivative $\frac{de(\omega)}{dT}$ should be used instead. The other required minor adjustments are discussed in the following subsections.

3.5.3 Phonon Origination

When calculating the total system energy, described in Section 3.4.5, some adjustments are made to incorporate the equilibrium temperature. The amount of energy contained in each cell is now calculated as follows:

$$\Delta E = |T_{\text{init}} - T_{eq}| \cdot V \sum_p \sum_{b=1}^{N_b} \frac{de(\omega_{b,p})}{dT_{\text{init}}} D(\omega_{b,p}, p) g_p \Delta\omega \tag{3.30}$$

where T_{init} refers to the initial temperature of the cells and is supplied by the user[9]. Similarly, the amount of energy from each emitting surface can be calculated as follows:

$$\Delta E' = |T_e - T_{eq}| \cdot \frac{A\Delta t}{4} \sum_p \sum_{b=1}^{N_b} v_{gb,p} \frac{de(\omega_{b,p})}{dT_e} D(\omega_{b,p}, p) g_p \Delta\omega \tag{3.31}$$

This implies that emitting surfaces with $T_e = T_{eq}$ will emit no phonons, and geometric cells attached to sensors with $T_{\text{init}} = T_{eq}$ will have no phonons originate within them.

[9]This is the 't_init' value associated with each sensor in the sample input file from Appendix B

3.5.4 Temperature and Flux Calculations

The energy E_j in each measurement area is calculated similarly to that described in Section 3.4.9 by Eq. (3.26), but calculating temperatures is no longer done with a numerical inversion but rather by using the equation:

$$T_j = \frac{E_j}{C \cdot V} + T_{eq} \tag{3.32}$$

where V is the volume of all the cells in the measurement area, and C is the heat capacity of the measurement area material, which is calculated as:

$$C = \sum_p \sum_{b=1}^{N_b} \frac{de(\omega_{b,p})}{dT_{\text{init}}} D(\omega_{b,p}, p) g_p \Delta\omega \tag{3.33}$$

T_{init} will likely be different from the temperature T_j in Eq. (3.32) when this calculation occurs unless the initial temperature of the measurement area is the same as the steady-state temperature.

Eq. (3.33) is a high-speed calculation compared to the typical numerical inversion approach because the quantity C has already been evaluated when creating the cumulative distribution tables since the derivative of the modified Bose-Einstein distribution is used to generate these tables.

A modification described in the following section ensures the initial temperatures are sensible while processing phonons independently, as described in Section 3.4.

3.6 System Evolution

At the beginning of a simulation, the user supplies the initial temperature of each measurement area/sensor. This temperature is used throughout the simulation when determining scattering rates and calculating the heat capacity of cells associated with that sensor using Eq. (3.33). In the case of extreme temperature differences, the sensor temperature will vary significantly throughout the simulation. Because phonons drift through the system one at a time, there is no straightforward way to update the sensor temperature during the simulation. The inability to update the sensor temperatures means the scattering rates inside each measurement area will not reflect potentially substantial changes to this initial temperature throughout the simulation. Multiple iterations of the simulation are run to account for this, and at the end of each iteration, the initial temperature of the sensors for the next run is updated with the temperature it reached at the end of the previous simulation.

The idea is that the system will eventually converge toward a steady-state where the temperatures of the measurement areas are stable within some degree of statistical uncertainty. By iteratively running the simulation and updating the initial temperature each time, then on the final iteration, the sensor temperatures, scattering rates and heat capacities will be accurate throughout the simulation. The last iteration is equivalent to a simulation where the user a priori knows the steady-state temperature of the system and uses that knowledge for the initial inputs.

For deviational simulations, this technique is also employed on the equilibrium temperature. A volume-weighted average system temperature is calculated at the end of each iteration, and the equilibrium temperature is updated to this new value. A linear system of silicon 1000 nm in the direction of interest is used to illustrate this approach. The general layout of the system can be seen in Figure 3.11. The leftmost Y-Z plane is an emitting surface with $T_H = 310$ K, and the rightmost Y-Z plane is an emitting surface with $T_C = 290$ K. The equilibrium temperature is 300 K. To magnify the effect of the system evolution, each sensor is somewhat arbitrarily assigned an initial temperature of 15 K to exaggerate the effects of the system evolution.

As seen in Figure 3.12, this causes a bizarre result on iteration one as each cell's scattering rates and heat capacities are calculated at 15 K and used throughout the simulation.

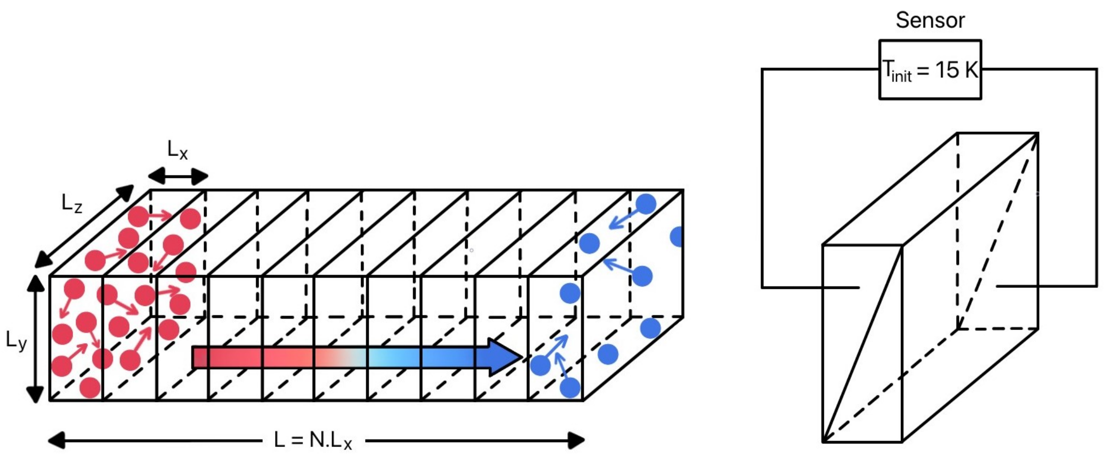

Figure 3.11: Linear system used to test the system evolution algorithm. L_z is an arbitrary measurement, see Section 5.3. Each geometric cell comprises two triangles linked to a single sensor to form one measurement area with an initial temperature of 15 K.

.

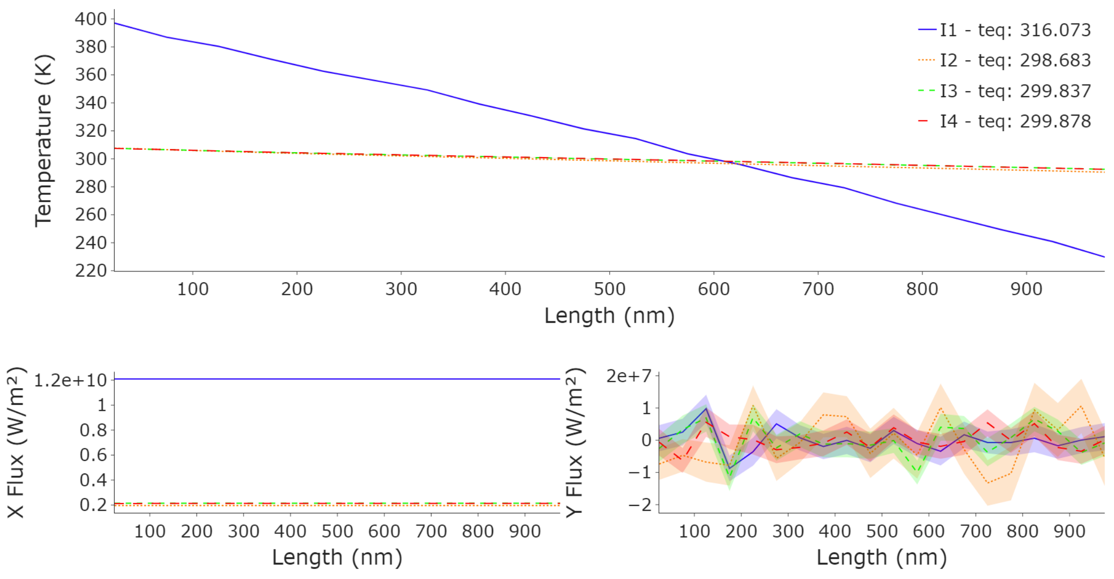

Figure 3.12: System evolution using a 1000 nm silicon wire. The initial cell temperatures are set to 15 K with $T_H = 310$ K and $T_C = 290$ K. The equilibrium temperature is initially set to 300 K.

The profiles converge rapidly, even when the starting temperatures deviate substantially from their steady-state temperatures. By the fourth iteration, the system has stabilized, see Figure 3.13. Stability is defined as 90% of the sensors' temperatures being within 0.5% of their temperature on the previous iteration and, if applicable, the equilibrium temperature being within 0.5% of its value on the previous iteration.

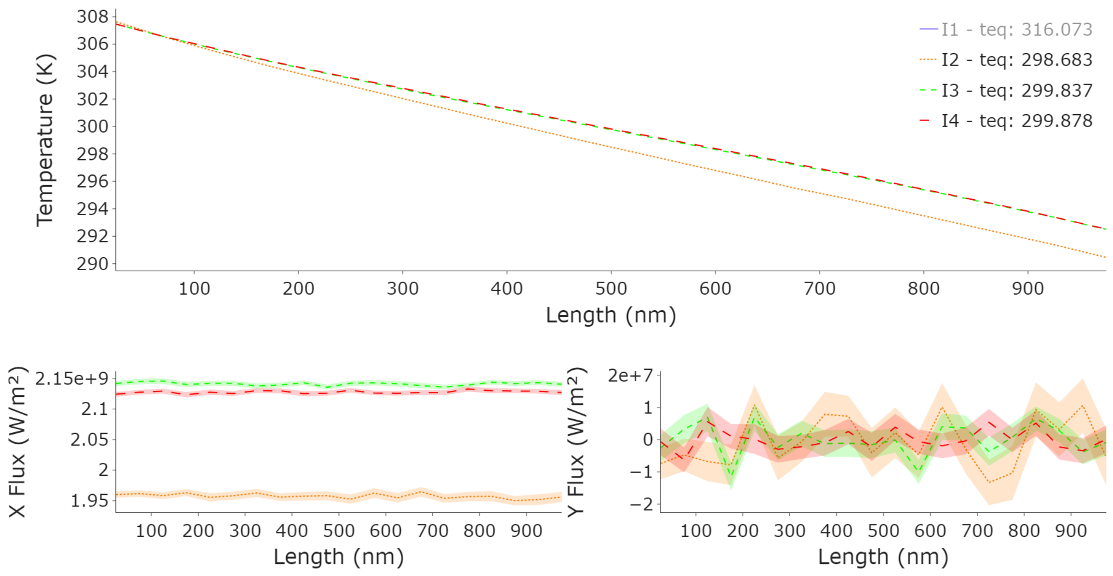

Figure 3.13: Steady-state convergence of the system in Figure 3.12. Iteration 3 and the final iteration are largely overlapping.

In Figure 3.14, the structural layout is the same as before, but the equilibrium temperature is chosen to be 290 K, and each cell is given an initial temperature of 300 K.

Despite the errant initial input for the equilibrium temperature, the convergence occurs rapidly. Convergence is generally expeditious unless the equilibrium temperature is initially set to an unreasonable value that is much greater than the warmest emitting surface in the system or much cooler than the coolest emitting surface.

3.7 Advantages of the Energy-Based Formulation

In addition to perfect energy conservation, the energy-based formulation eliminates the time discretization requirement as it pertains to scattering events. The simulation must still be interrupted periodically for the sensors to record energy and flux mea-

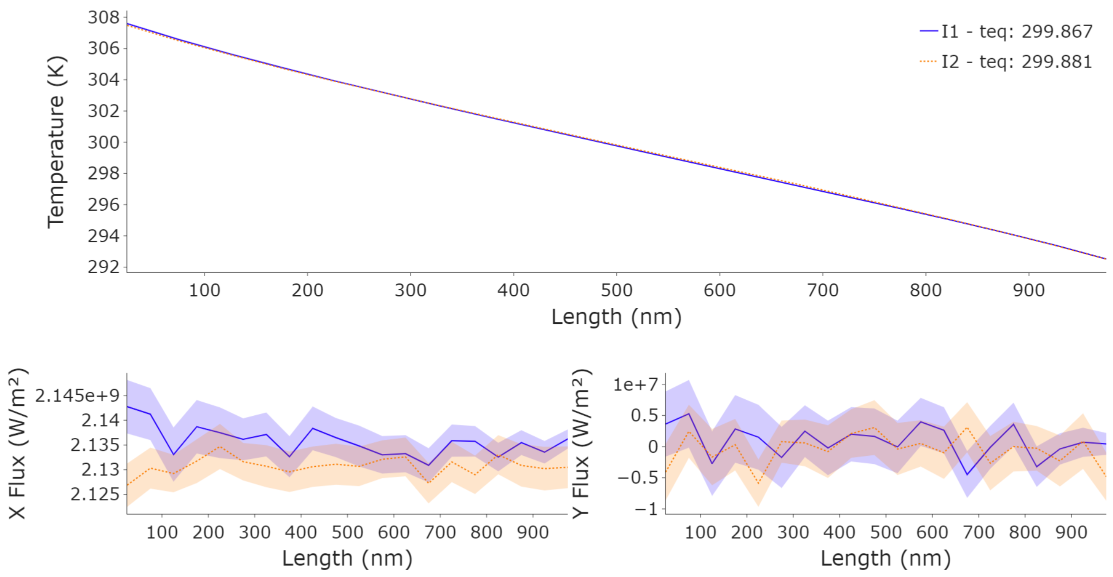

Figure 3.14: Steady-state convergence with an errant initial equilibrium temperature.

surements, but specifying a time-step to check scattering probabilities is unnecessary. Instead, Eq. (3.18) is used to calculate the time until the next scattering event occurs rather than simulating a prescribed amount of time and then calculating the probability that the phonon undergoes a scattering event during that time frame using the equation:

$$P(\omega, p, T) = 1 - \exp\left(\frac{\Delta t}{\tau(\omega, p, T)}\right) \tag{3.34}$$

This is advantageous as choosing an appropriate time-step is not necessarily straightforward. Section 3.8 discusses this in greater detail.

The absence of this requirement and taking advantage of Eq. (3.18) also allows each phonon energy packet to be processed independently from start to finish. This algorithm leads to much more natural software parallelization opportunities. In addition, it drastically reduces the memory footprint of the simulation compared to previous approaches, as memory storage is required only for a single phonon rather than all the phonons in the system.

3.8 Time-Step Shortcomings

When using Eq. (3.34) to calculate the probability of scattering over a period of time, it is crucial to use an appropriate scattering substep or simulation time-step. For example, if the chosen time step is much larger than the combined relaxation time of the medium, then the scattering probability will always be 1, which is unrealistic. A scattering probability of 1 implies that the phonon is likely to undergo more than one scattering event over the given period of time, and we will end up undercounting scattering events throughout the simulation, which will undoubtedly lead to an artificial increase in the system flux.

For example, Figure 3.15 shows that with a time-step of 5×10^{-12} s, the probability that a phonon in silicon with a frequency of 2.4×10^{13} rad s^{-1} and TA polarization will scatter over this period is $\sim 60\%$ which is quite high despite the seemingly small time-step. Figure 3.16 indicates that this time-step will undercount scattering events throughout the simulation leading to higher than expected flux measurements.

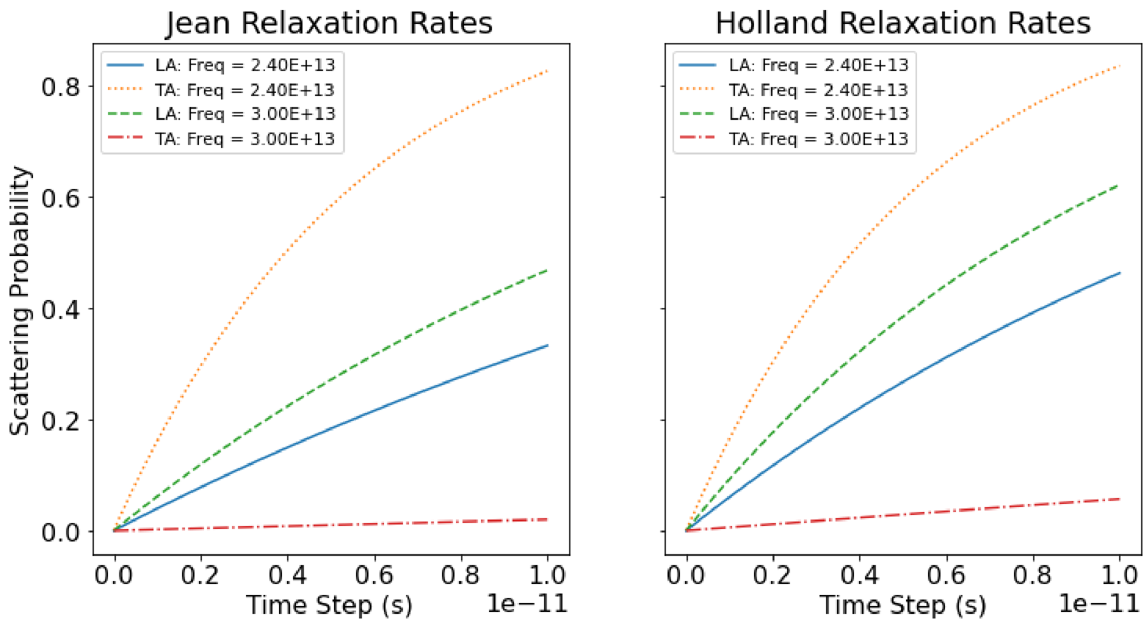

Figure 3.15: time-step impact on scattering probabilities at 300K.

In addition, it has also been suggested by Wong *et al.* [15], among others, that the ballistic distance of the fastest phonon ensemble should not exceed the size of the smallest spatial division in any given time-step. These factors are addressed by choosing a minimal time-step, but this can have an unintended effect.

The problem with having too small of a time-step stems from the energy conservation strategy used in many of the Monte Carlo methods in the literature [1, 3, 15]. The energy conservation mechanism involves an ad-hoc phonon addition/deletion scheme.

Typically, this addition/deletion scheme is run in intervals equal to the time-step used to calculate scattering probabilities. A major inconvenience with this approach is that if the time-step is too small, the phonon addition/deletion scheme can inadvertently destroy a disproportionate number of phonons with biased direction vectors, those that have originated from the emitting/isothermal surfaces. The effect is an unintended decrease in the flux through the system, which can be seen in Figure 3.16.

It is also evident from Figure 3.16 that if the time-step is too large, scattering events will occur less frequently than they otherwise would, leading to an unintended increase in flux, as mentioned above.

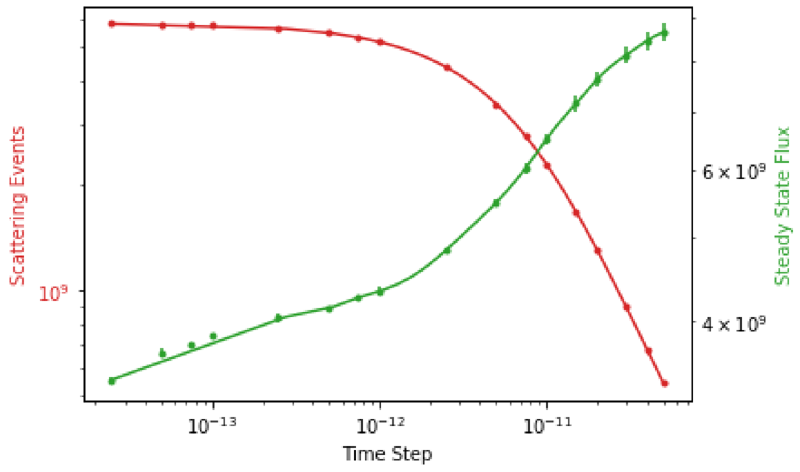

Figure 3.16: time-step impact on steady-state flux.

Chapter 4

Results

The following results are intended to verify the simulation code and display the potential of the Monte Carlo method described in Chapter 3 as a solution technique for the Boltzmann transport equation. In all cases, results are taken from systems that have evolved to a steady-state, meaning the energy and flux within each measurement area is unchanging in time within a margin of numerical uncertainty.

The first set of results in this compare the phonon Monte Carlo predicted thermal conductivity of silicon to the known result in the diffusive regime. Multiple sets of relaxation time parameters are used to test the methods' sensitivity to these inputs. Next, phonon Monte Carlo predicted temperature profiles from the diffusive and ballistic regime are compared to analytical results obtained using the Stefan Boltzmann and heat diffusion equations. In addition, results from both the full and deviational simulation approaches are displayed so the differences between these approaches can be discussed. In the following sections, results from more complex geometries are presented to showcase the utility and speed of the phonon Monte Carlo method. Again, the full simulation approach is compared to the deviational approach. Finally, the chapter finishes with a summary that compares the full simulation approach to the deviational approach.

As discussed in Section 3.4.3, but repeated here for convenience, the third (z) dimension is assumed to be of arbitrary or infinite size for all the systems presented in this chapter. This means phonon movement and surface interactions pertaining to that dimension are ignored throughout the simulation. The three assumptions that allow for this are as follows.

- All geometric cells, and the system as a whole, are assumed to have the same measurement specification in the z-dimension.
- No emitting surfaces can be located on the z-dimension surfaces.
- The z-dimension surfaces are perfectly specular.

All results have been produced using the deviational technique with the dispersion and relaxation time parameterizations by Jean [27] and without impurity scattering unless otherwise stated. In most cases, the number of phonons and simulation time far exceeds what is necessary to produce low-variance solutions.

4.1 Nanowires

The systems in this section all take the general form of rectangular segments. This type of structure is commonly used in the literature [1, 3, 15] as a proxy for nanowires for benchmark testing of the phonon Monte Carlo method.

The general geometry of the wires used in each section is depicted in Figure 4.1, which is similar to the system described by Figure 3.11 if the prior assumptions are in place. Additional details can be found in the appropriate subsections and tables.

4.1.1 Thermal Conductivity

The results in this section look at the phonon Monte Carlo predicted thermal conductivity of silicon and germanium nanowires as a function of wire length. The thermal conductivity of these materials is well studied at the macro-scale under non-extreme temperatures, and the bulk conductivities can be approximated using Eq. (4.1) and Eq. (4.2) for silicon and germanium, respectively [3].

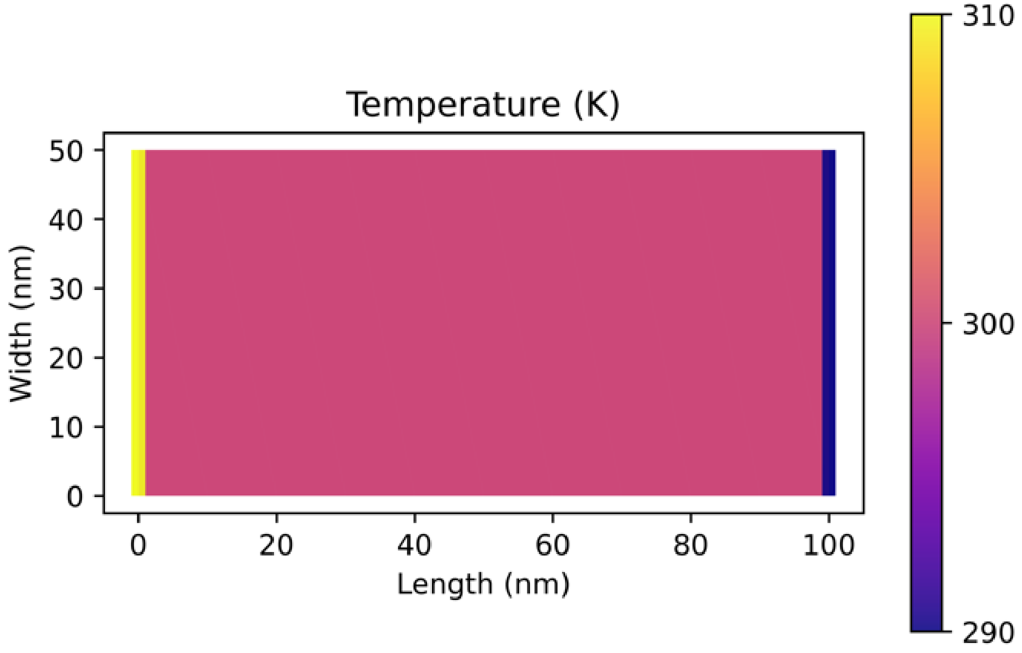

Figure 4.1: A 100 nm long and 50 nm wide silicon nanowire. This wire is split into ten measurement areas, each with dimensions $L_x = 10$ nm and $L_y = 50$ nm. The emitting surface at $x = 0$ is set to $T_H = 310$ K, and the emitting surface at $x = 100$ is set to $T_C = 290$ K. The initial temperature of each measurement area is $(T_H + T_C)/2$.

$$\kappa_{Si}(T) = \frac{e^{12.570}}{T^{1.326}} \tag{4.1}$$

$$\kappa_{Ge}(T) = \frac{e^{10.659}}{T^{1.150}} \tag{4.2}$$

The phonon Monte Carlo predictions are expected to align with these equations' results as we approach the macro-scale. The thermal conductivity is no longer an intrinsic property of the material at the nanoscale and cannot be well-approximated by these equations. The validity of the predicted results for the shorter wires will be compared to those obtained by Wong *et al.* [15].

The general geometry for all the systems can be seen in Figure 4.1. Table 4.1 shows the common simulation settings for all systems. L_x and L_y refer to the dimensions of the measurement areas. For example, a 1000 nm wire consists of 100 rectangular measurement areas of $L_x = 10$ nm and $L_y = 50$ nm. Each measurement area is comprised of 2 triangular cells.

Table 4.1: Common simulation settings for the thermal conductivity benchmarks.

Simulation time (ns)	Number of Phonons	Number of Measurements	L_x (nm)	L_y (nm)	Surface Specularity
50	50,000,000	5000	10	50	1

The wires are elongated by adding more cells. The thermal conductivity is calculated by taking the average of the final 500 (10% of total measurements) flux measurements in the x-direction from each measurement area. Each of these values is nearly identical, as expected. The average of these values is used as the system flux, q_x, in the x-direction. The thermal conductivity κ is then calculated as:

$$\kappa = \frac{q_x L}{\Delta T} \tag{4.3}$$

where L is the system length in the x-direction and ΔT is the temperature differential across the system. The temperature differential is 20 K for each system in this subsection.

Four different sets of relaxation time parameters are used. The κ_{jean} relaxation rates are taken from [27], κ_{wong} from [15] and $\kappa_{holland}$ from [18]. The different sets of relaxation time parameters are meant to test the method's sensitivity to these inputs.

Results for varying lengths of silicon and germanium nanowires can be seen in the figures below.

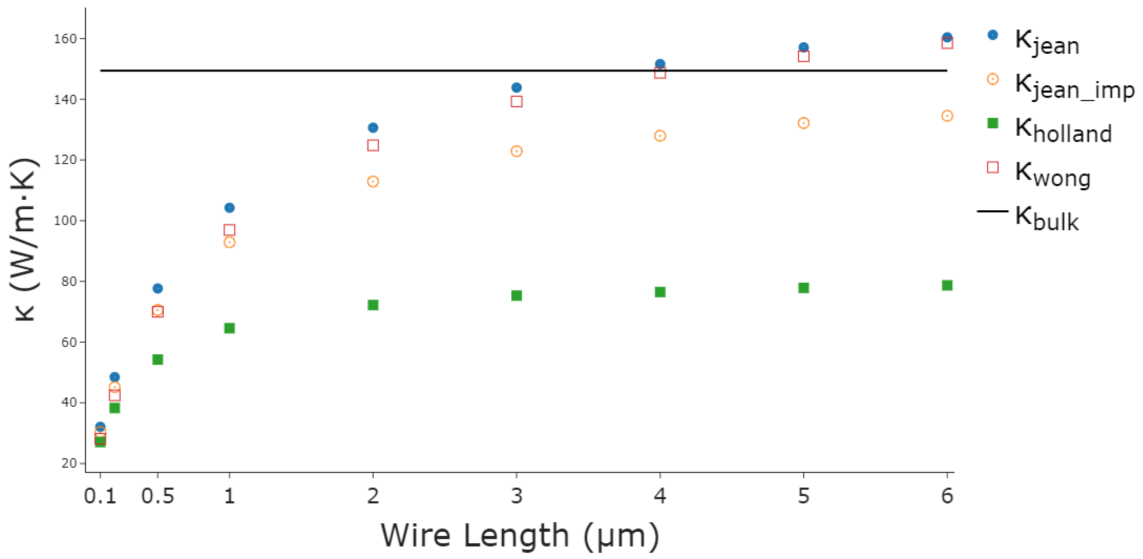

Figure 4.2: Thermal conductivity as a function of wire length in silicon at 300K. $T_H = 310$ K and $T_C = 290$ K. Error bars are present but not easily visible at this scale. This is also the case for the figures that follow.

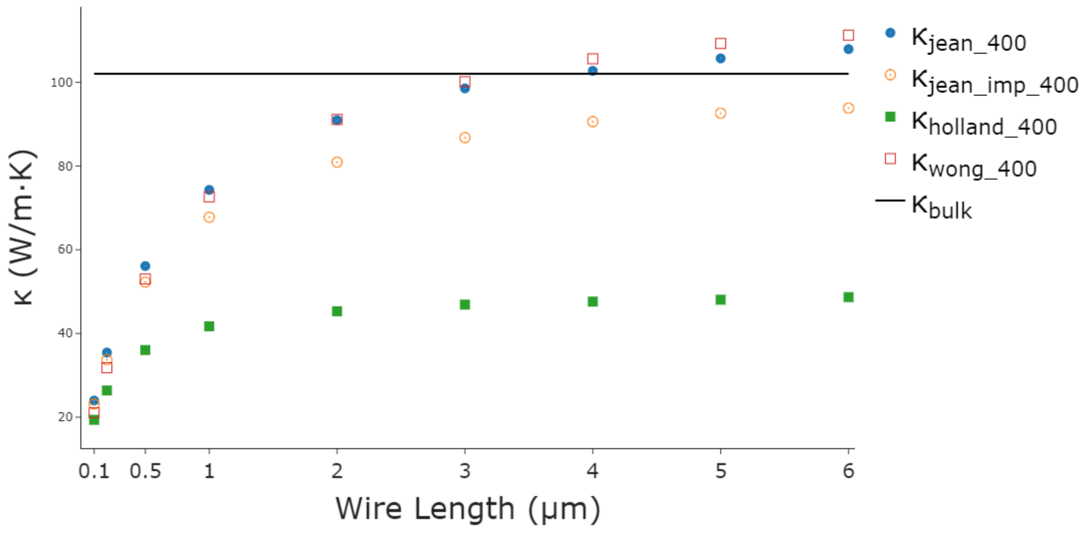

Figure 4.3: Thermal conductivity as a function of wire length in silicon at 400K. $T_H = 410$ K and $T_C = 390$ K.

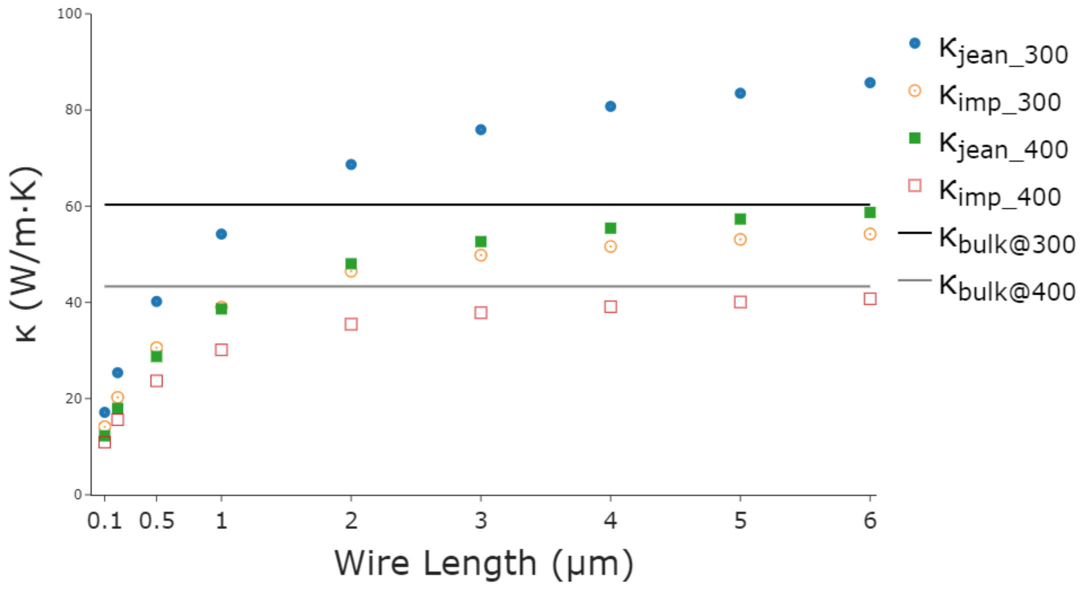

Figure 4.4: Thermal conductivity as a function of wire length in germanium at 300K and 400K. $T_H = 310K$ and $T_C = 290K$ for the 300 results and $T_H = 410K$ and $T_C = 390K$ for the 400 results.

The results from Figure 4.2 and Figure 4.3 using the Jean and Wong parameters closely match the results of similar testing by Wong *et al.* [15]. There is good agreement with the known conductivity as we approach the diffusive regime, and tuning the relaxation time parameters or the surface specularity will allow near-exact reproduction of the known bulk conductivity.

Lacroix, Joulain and Lemonnier [3] reported good accuracy for silicon thermal conductivity using the Holland rates, which is not seen here. Two potential factors may explain this.

1. The study used a rather large time step which may have artificially increased the flux by the mechanism described in Section 3.8.

2. An adjustment is made in the paper such that half of the colliding phonons keep their momentum and are not directionally resampled during collisions. Allowing

phonons to maintain their momentum after collisions will significantly increase the flux through the system and hence the thermal conductivity.

The germanium results from Figure 4.4 are similar to the silicon results. Again, it is clear that the bulk values can be matched precisely in the diffusive regime with appropriate adjustments to the relaxation time parameters and surface specularity.

4.1.2 Ballistic Regime

This section compares the full and deviational techniques at low temperatures. In this ballistic regime, where the phonon mean free path is greater than the structure length, the temperature in the steady-state can be approximated using the following constant value that follows the Stefan-Boltzmann law [3, 15]:

$$T_{\text{ballistic}} = \left(\frac{T_H^4 + T_C^4}{2} \right)^{\frac{1}{4}} \tag{4.4}$$

It is expected that the steady-state temperature predicted by the phonon Monte Carlo method will be comparable to the result from Eq. (4.4). The temperature in each measurement area along the length of the wire is calculated by taking the average of the final 500 (10% of total measurements) temperature measurements.

Some scattering may still occur even at these low temperatures, so we should not necessarily anticipate an exact match.

Common simulation settings for benchmarks presented in Figure 4.5 and Figure 4.6 can be found in Table 4.2.

Table 4.2: Common simulation settings for the ballistic regime benchmarks.

Simulation time (ns)	Number of Phonons	Number of Measurements	L_x (nm)	L_y (nm)	Surface Specularity
50	50,000,000	5000	10	50	1

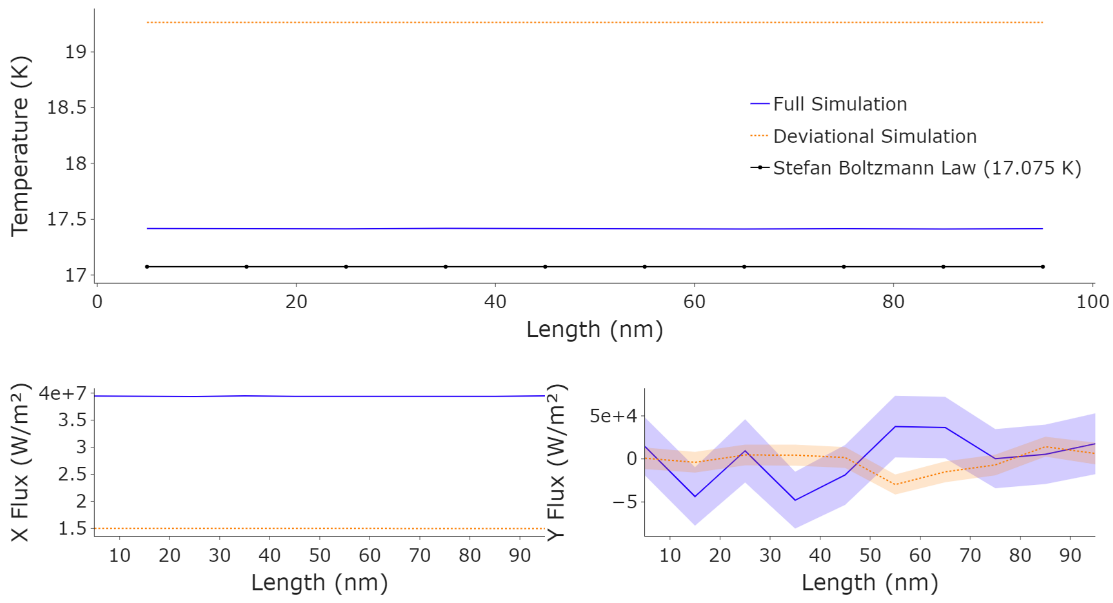

Figure 4.5: Ballistic results for 100 nm silicon with $T_H = 20$ K and $T_C = 10$ K. The standard error in the measurements is represented by the shaded areas but is not easily visible for the temperature results at this scale.

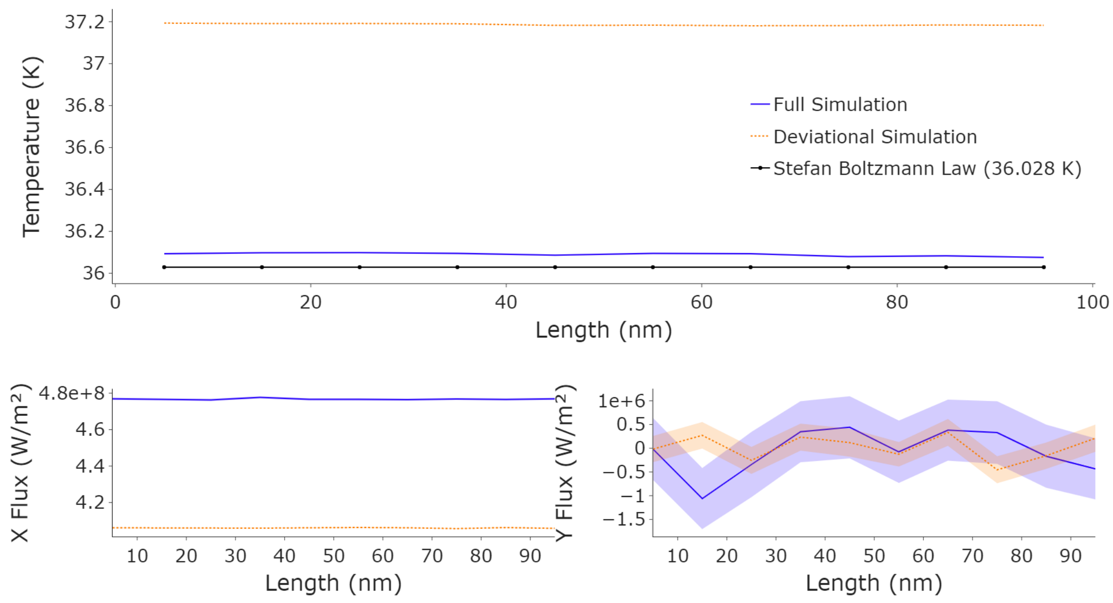

Figure 4.6: Ballistic results for 100 nm silicon with $T_H = 40$ K and $T_C = 30$ K.

Excellent agreement is found with the results from the full simulation technique and the predicted result from Eq. (4.4). The deviational simulation proves less accurate, compared with the Eq. (4.4), in the low-temperature range, where a 10 K temperature differential is relatively large. The full simulation results also agree with Wong *et al.* [15].

4.1.3 Diffusive Regime

This section compares the full and deviational techniques at higher temperatures where the mean free phonon path is much less than the system length, and heat transport is largely diffusive. For short wires, the system is in the ballistic regime where we expect a constant temperature profile like that given by Eq. (4.4). By increasing the wire length, we move from the ballistic regime to the diffusive regime, where Fourier's law has predictive power.

A 5000 nm silicon wire is used for the figures below. The values for T_H and T_C used in Figure 4.7 are specified in Table 4.3. The ordering of the temperature specifications in Table 4.3 matches the ordering of the temperature profiles in Figure 4.7. For example, the profile intersecting the y-axis at approximately 330 K has $T_H = 330$ and $T_C = 270$.

Table 4.3: Emitting surface temperatures for the profile in Figure 4.7.

T_H (K)	T_C (K)	ΔT (K)
330	270	60
315	285	30
310	290	20
302	298	4

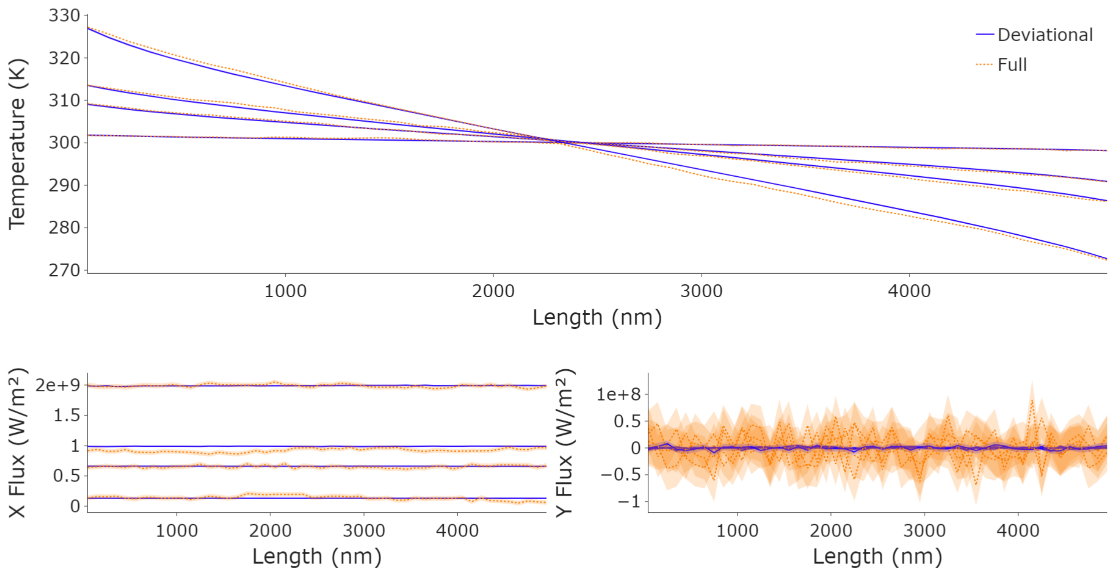

Figure 4.7: Diffusive regime results for a 5000 nm silicon wire with varying temperature differentials.

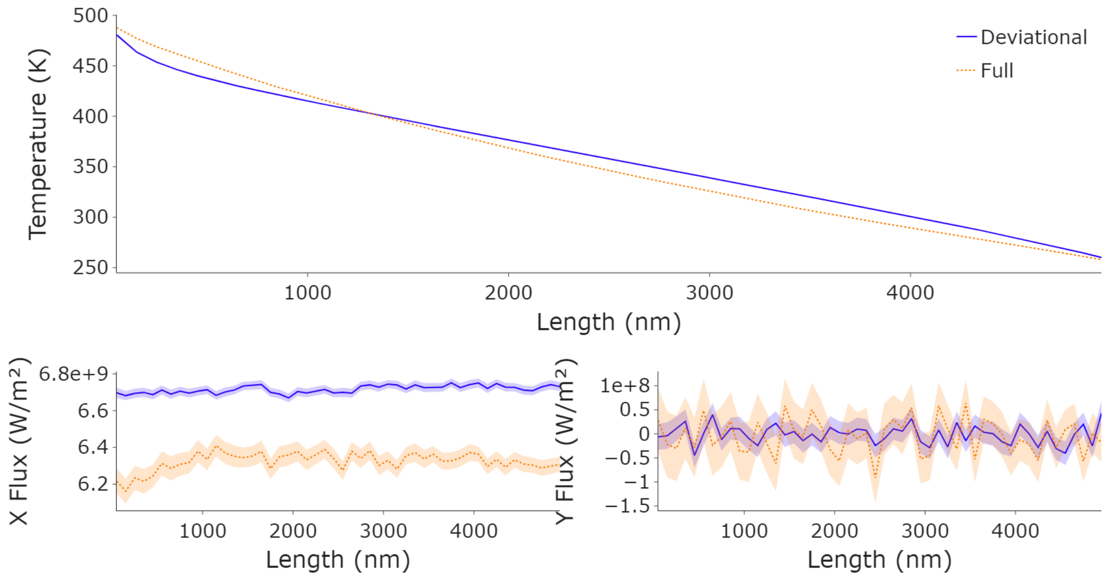

Figure 4.8: Diffusive regime results for a 5000 nm silicon wire across a 250 K temperature differential. $T_H = 500$ K and $T_C = 250$ K.

In Figure 4.7, the temperature differentials are relatively small, and we see a minimal difference between the deviational and full simulation approaches. Temperature profiles are mainly linear, and the mean (center) temperature is approximately 300 K, as expected in the diffusive regime.

However, the temperature profiles exhibit a very slight bowing effect near the hot and cool ends of the wire. The curvature of the profiles also increases as the temperature differential across the wire increases. This curvature is more pronounced in the deviational simulation result, particularly when $T_H = 330K$ and $T_C = 270K$. These bowing and curvature effects are noticeable in the Wong *et al.* results. [15].

The system in Figure 4.8 has a relatively large temperature differential across the wire, 250 K. The deviational and full simulation results diverge even further, with the deviational results exhibiting a sizable bowing effect. The flux measurements, in the direction of interest, have also diverged. In Figure 4.9, the predicted temperature profile using the heat diffusion equation is included.

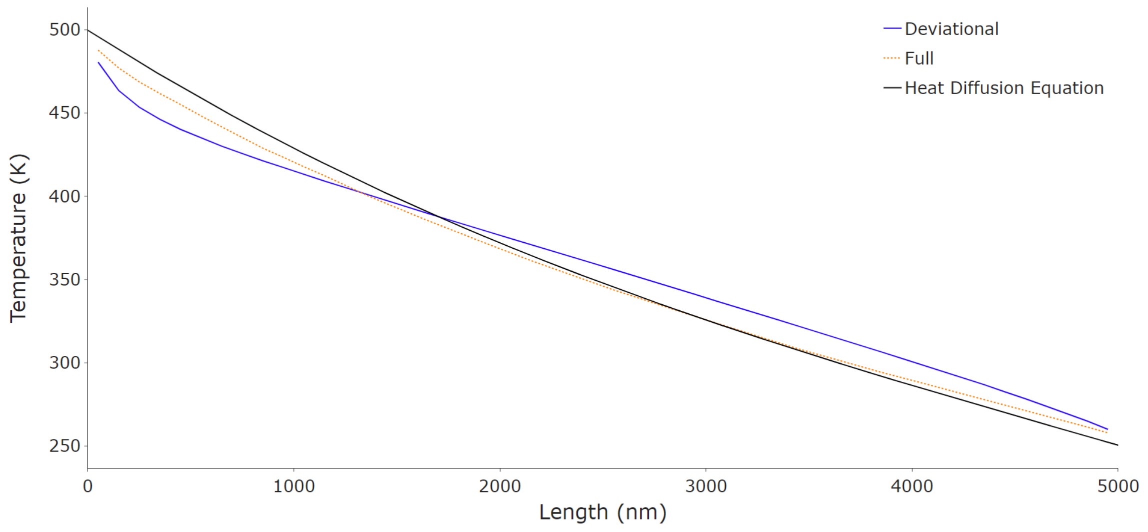

Figure 4.9: Deviational and full simulation results compared with the heat diffusion equation across a 250 K temperature differential.

Figure 4.9 is a closer look at the temperature profiles in Figure 4.8 and includes a result produced by the heat diffusion equation where the thermal conductivity is temperature dependent. This temperature dependence results in the non-linear profile in Figure 4.9.

Based on Figure 4.9, the result from the full simulation better matches that predicted by the heat diffusion equation. Further study is required to determine whether this difference is from the introduction of the equilibrium temperature and the deviational particles or if the linearization of the Boltzmann equation causes the difference. It is also plausible that all these factors may play a role.

In addition to reasonably approximating the result from the heat equation, the full simulation result in Figure 4.9 also closely matches the result from Wong *et al.* [15].

Finally, in Figure 4.10, the 4 K temperature differential result is displayed on a different scale than previously shown in Figure 4.7.

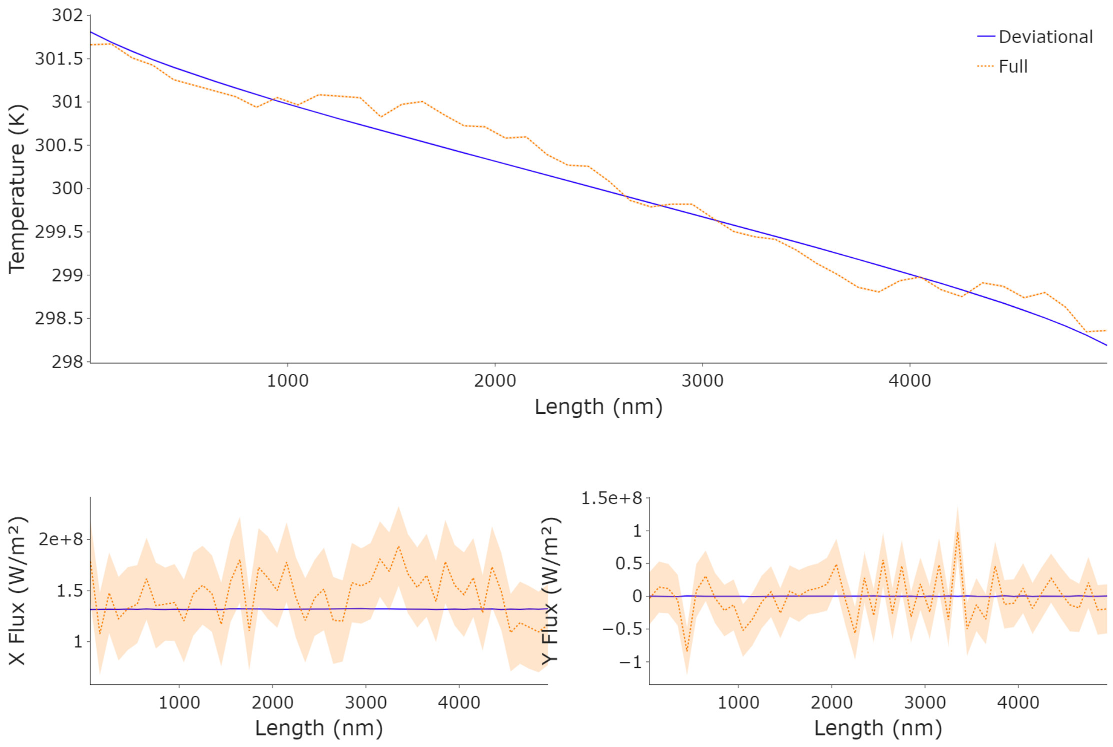

Figure 4.10: Deviational and full simulation results compared with the heat diffusion equation across a 4 K temperature differential.

At this scale, the difference in the standard error between the deviational and full simulation methods is quite apparent. More interestingly, the temperature profile for the full simulation, while still directionally accurate, shows some significant aberrations, while the deviational simulation still produces a clean result, even across a system with a tiny temperature differential.

The results from this section show that the full and deviational approaches produce similar steady-state temperature profiles and flux approximations as we approach the diffuse regime and the temperature differential across the system is 30 K or less. However, the deviational approach has much less variance and should be preferred. As the temperature differential across the system grows, the results from the two approaches start to diverge. Based on Figure 4.9, the full simulation results more closely match the result predicted by the heat diffusion equation, which should be a reasonable estimate at these length scales and temperature ranges. In all cases, the deviational simulation produces far less variance than the full simulation, about an order of magnitude lower when comparing the flux estimations.

4.2 Kinked Nanowires

This section compares the full and deviational techniques for kinked silicon wires and the relationship between flux and the kink angle of the wire. The effect of adding more measurement areas to a relatively complex geometrical system is also explored.

The generalized geometry of the wires can be seen in Figure 4.11. This figure is a schematic for half the system, but the other half of the system is symmetric, as can be seen in the full-page figures that follow. The kink angle is represented by α in the schematic.

The motivation for simulating these kinked wires is to compare the phonon Monte Carlo results with comparable results produced by molecular dynamics simulations. Such a comparison is interesting since the molecular dynamics simulations can account for isotropy in the crystal lattice but are much more computationally demanding. However, the results of this comparison are not included in this thesis to maintain the focus on the development of the phonon Monte Carlo simulation program, and the kinked wire results are included to showcase the utility and speed of the phonon Monte Carlo method. Each of the following simulations was executed in less than 10 minutes, orders of magnitude faster than a comparable molecular dynamics simulation.

Common simulation settings used to produce the results for all the figures in this subsection can be found in Table 4.4.

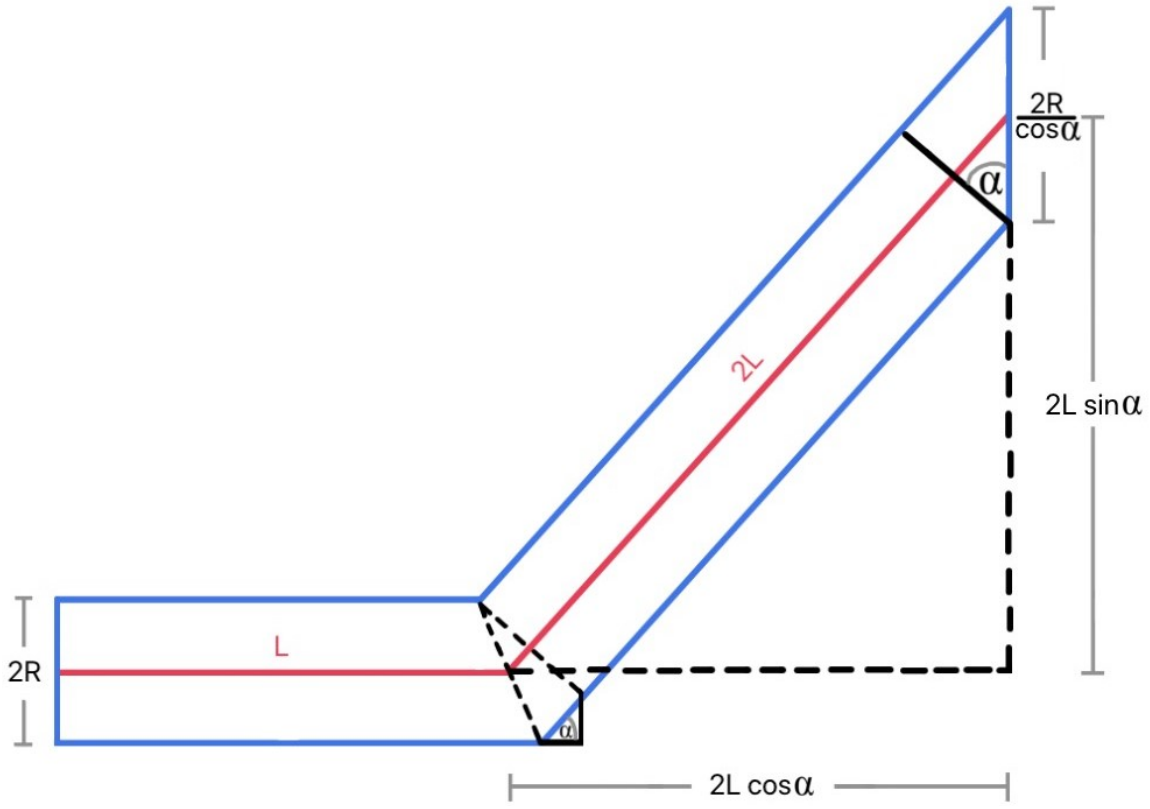

Figure 4.11: Kinked nanowire schematic.

Table 4.4: Common settings for the kinked wire simulations.

Simulation time (ns)	Number of Phonons	Number of Measurements	L (nm)	R (nm)	Surface Specularity
20	10,000,000	2000	120	$L/3$	1

There are six figures on the following pages. In Figure 4.12, a silicon wire with a 30-degree kink is simulated using 15 measurement areas/sensors. Figure 4.13 displays the same system, except it has been created with 3018 sensors. These figures are intended to show how certain thermal effects can be missed if the energy and flux measurements are averaged over too large of an area.

In Figure 4.14 and Figure 4.15, a silicon wire with a 45-degree kinked is simulated using the deviational approach, Figure 4.14, and the full simulation approach, Figure 4.15. The figures primarily showcase the differences between the two methods when simulating a relatively complex system.

Figure 4.16 and Figure 4.17 are similar, except the kink angle has been increased to 60 degrees. These figures again showcase the differences between the two simulation approaches but are primarily intended to briefly explore how the heat conduction properties are affected by the wire's kink angle.

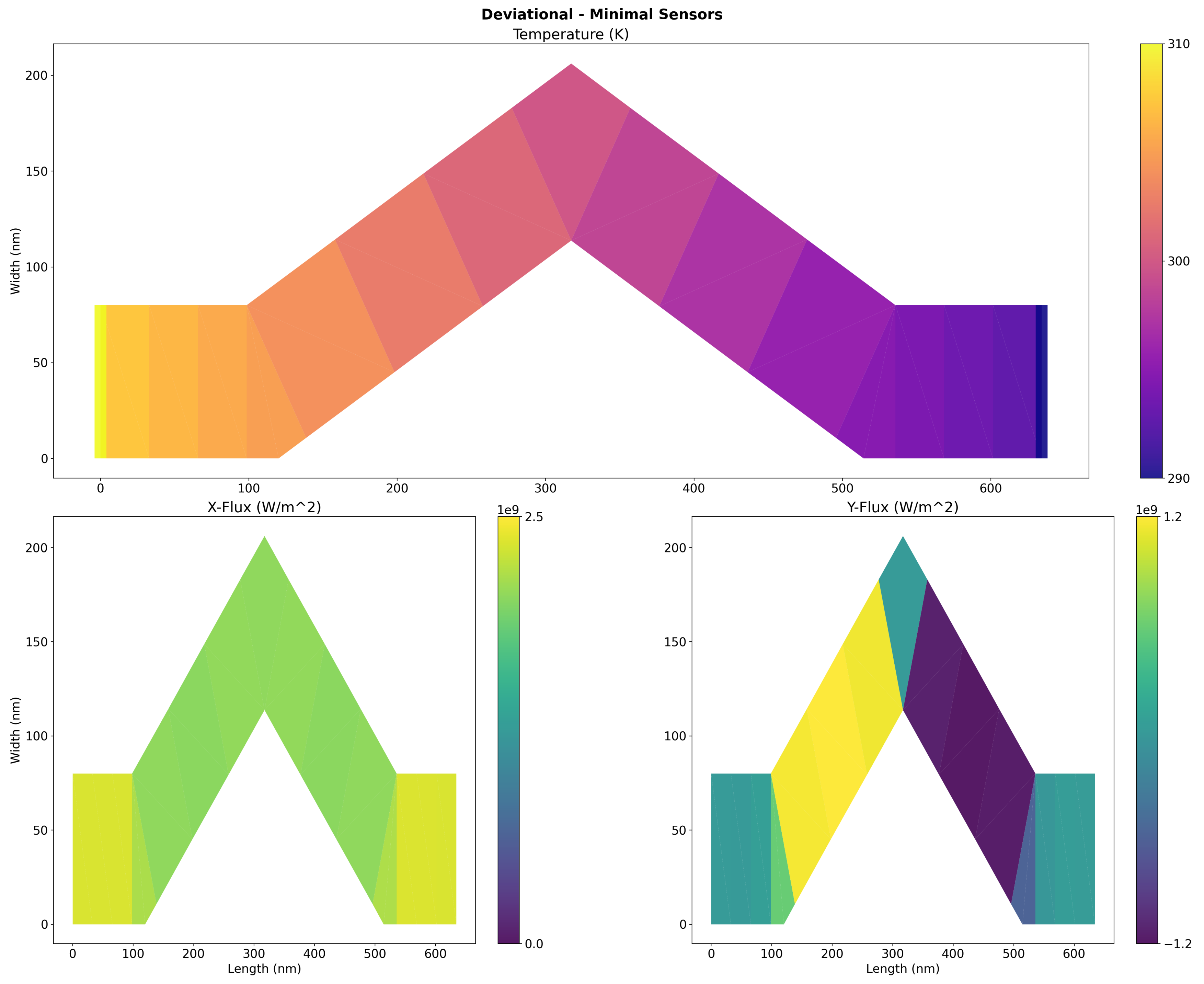

Figure 4.12: Temperature and flux profiles of a silicon wire with a 30-degree kink using 15 sensors.

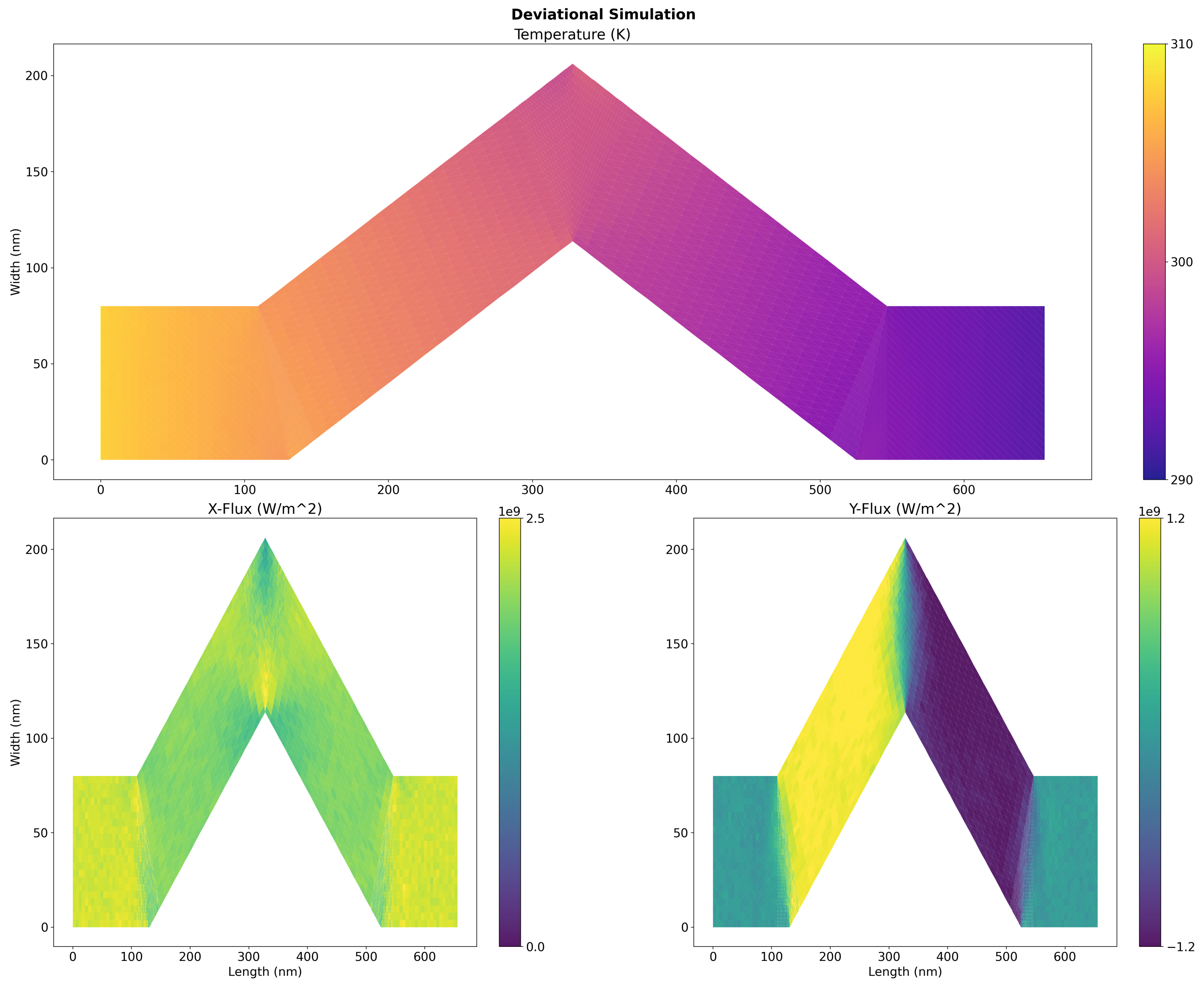

Figure 4.13: Temperature and flux profiles of a silicon wire with a 30-degree kink using 3108 sensors.

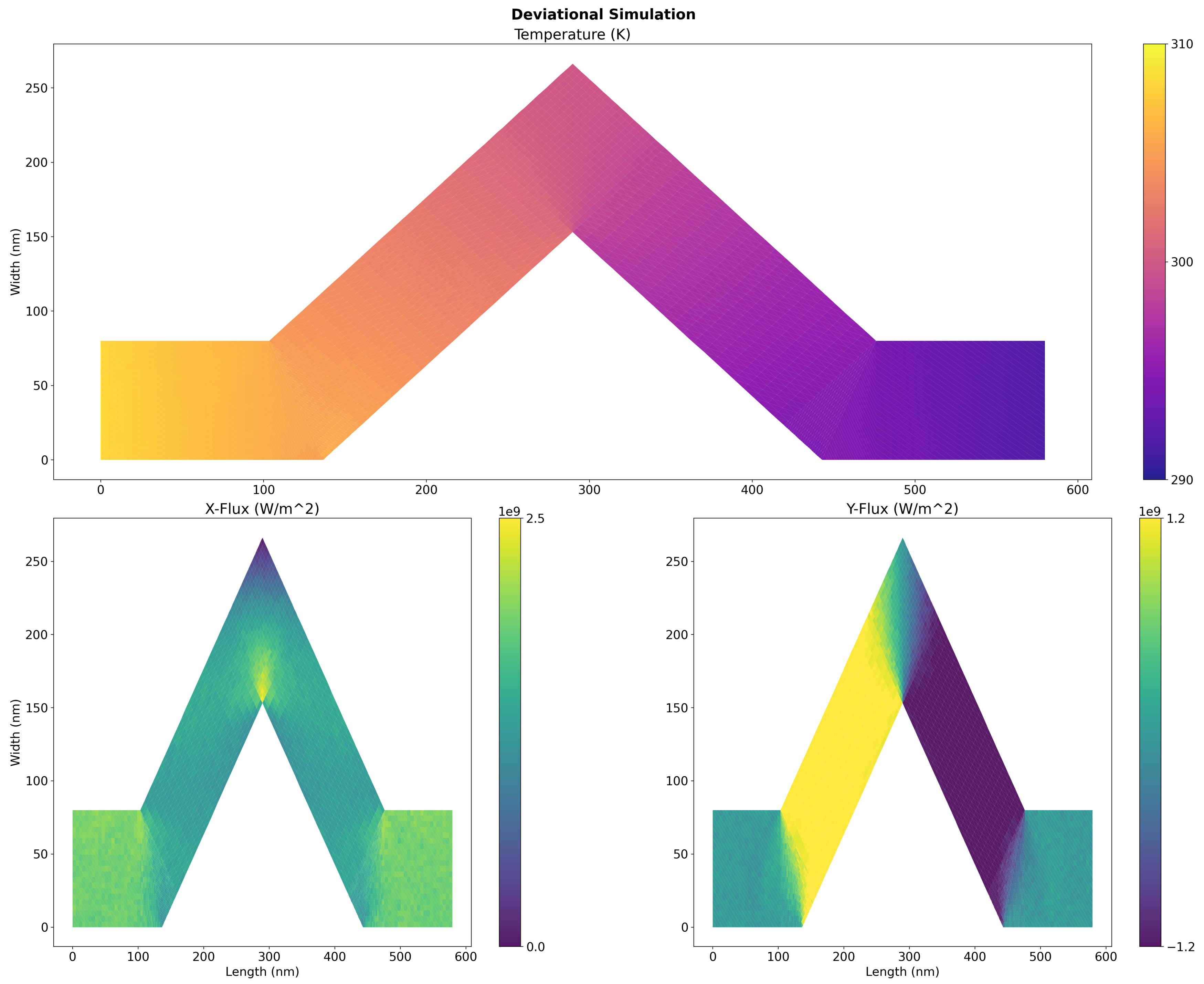

Figure 4.14: Temperature and flux profiles of a silicon wire with a 45-degree kink produced using a deviational simulation.

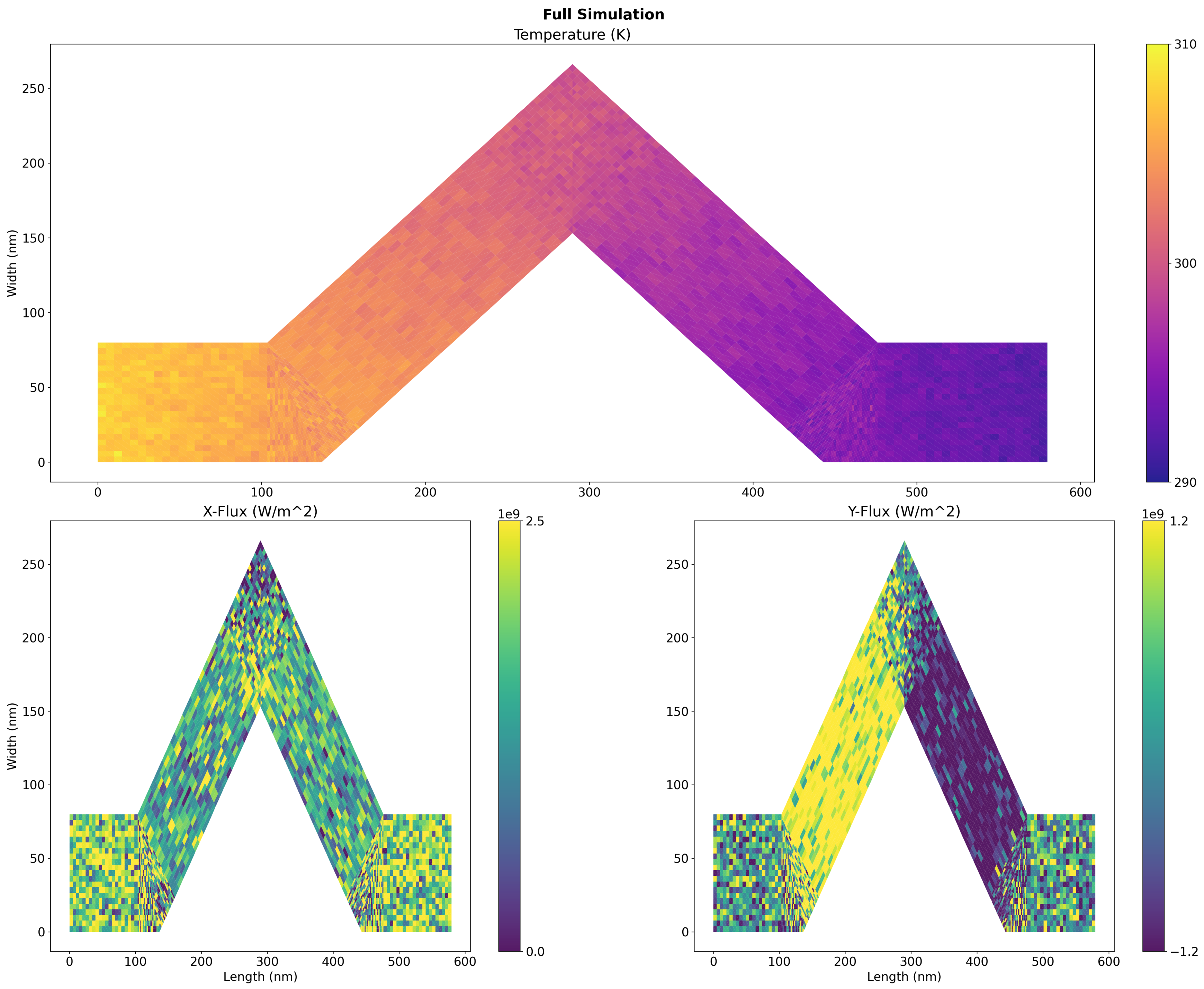

Figure 4.15: Temperature and flux profiles of a silicon wire with a 45-degree kink produced using a full simulation.

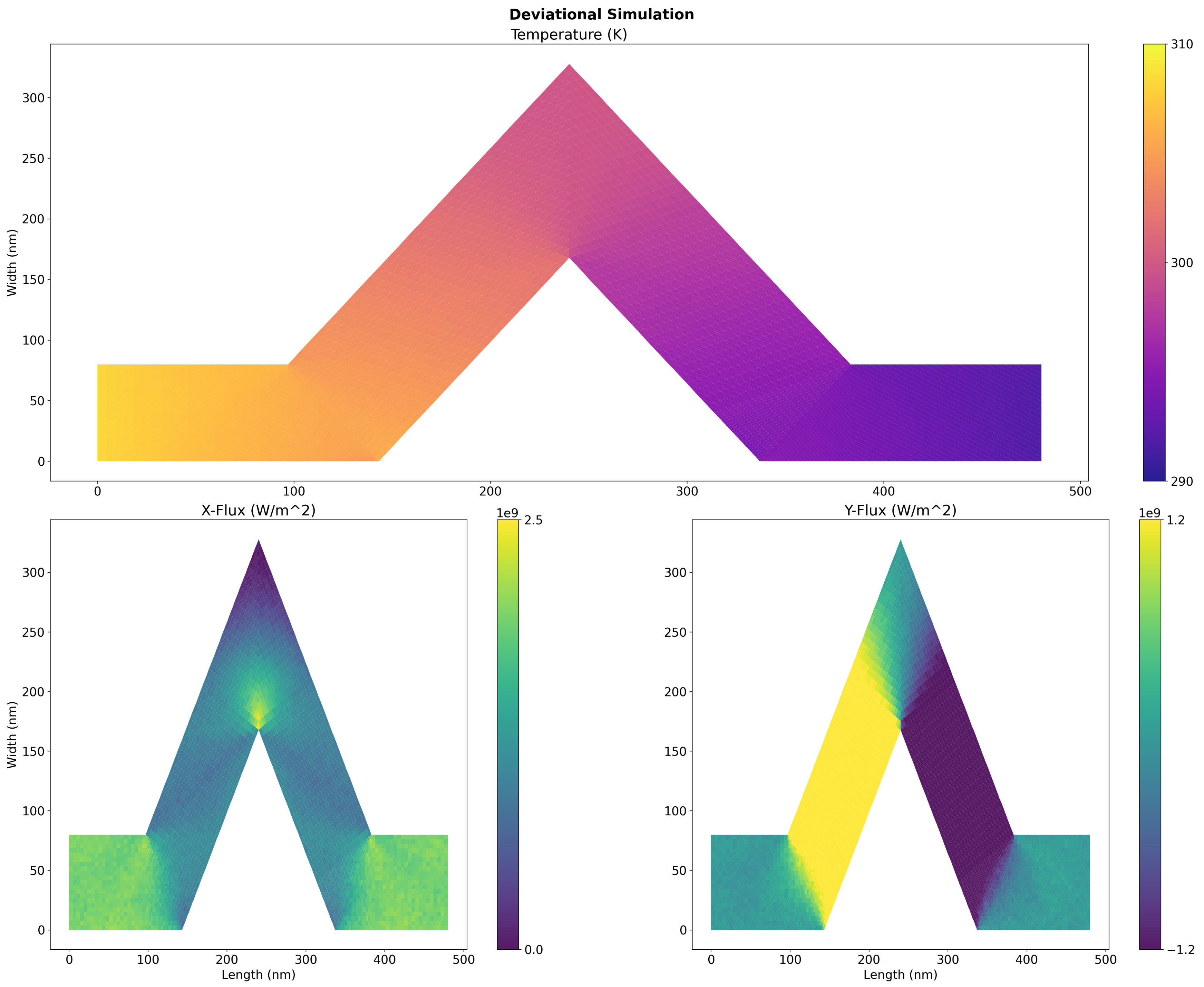

Figure 4.16: Temperature and flux profiles of a silicon wire with a 60-degree kink produced using a deviational simulation.

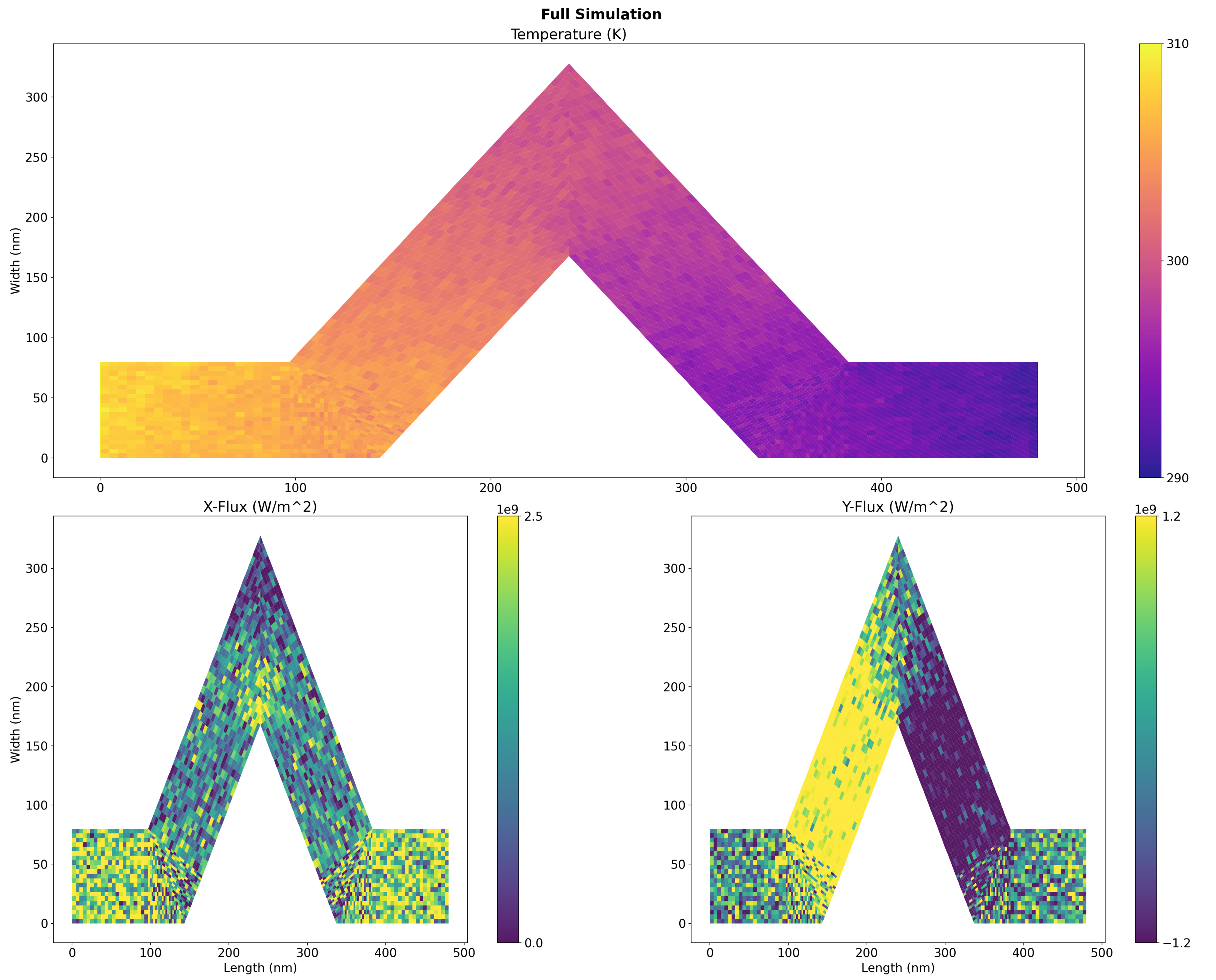

Figure 4.17: Temperature and flux profiles of a silicon wire with a 60-degree kink produced using a full simulation.

Some interesting patterns emerge as the number of sensors increases. Notably, we can see that the flux in the x-direction is not uniform through the bends and is not uniform through the angled sections.

The full simulations have too much noise to be of much use, at least with the current simulation settings. It is possible that adjusting the flux scale and running through additional phonons will reduce the noise and allow a better analysis of these results.

The system flux through the wire noticeably decreases as the kink angle increases. Figure 4.18 explores the effect of the kink angle on the flux and the effect of decreasing the phonon mean free path by increasing the rate of scattering in the system. The scattering rate is adjusted by dividing the time to scatter obtained by Eq. (3.18) by the factor in the legend.

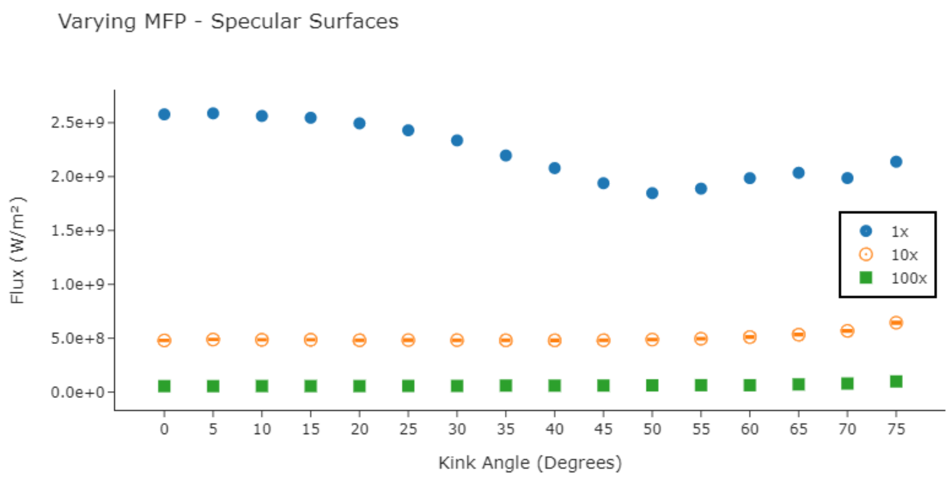

Figure 4.18: Flux through a kinked wire as a function of the kink angle. Errors bars are present but not visible at this scale.

The system flux decreases, as expected, until 55 degrees, at which point it starts climbing again. There are two possible explanations for this.

1. At high kink angles, the bends on the top kinked section of the wire start to become extremely thick, and at extreme angles, > 60, the top section of the wire starts to merge into a single fused section.

2. There may be some ballistic effects that occur with higher kink angles which allow the phonons to pass through the system with fewer redirections from surface collisions.

Increasing the scattering rate not only drastically lowers the flux, as expected, but also mitigates any effects due to the kinking. The slight upward trend at the higher angles is likely due to the first factor described above.

4.3 Thin Wafers

Silicon and germanium wafers are tested using the deviational and full simulation approaches. These wafers tests use the same temperature, and geometrical specifications described by Mazumder and Majumdar [1]. The wafers are squares of dimension 400 nm, and the left wall was set to 600 K, the top wall to 500 K, the right wall to 400 K and the bottom wall to 300 K.

Common simulation settings used to produce the results for all the figures in this subsection can be found in Table 4.5.

Table 4.5: Common settings for the wafer simulations.

Simulation time (ns)	Number of Phonons	Number of Measurements	L_x (nm)	L_y (nm)	Surface Specularity
20	60,000,000	5000	10	10	1

Figure 4.19 and Figure 4.20 are the silicon and germanium wafer results, respectively.

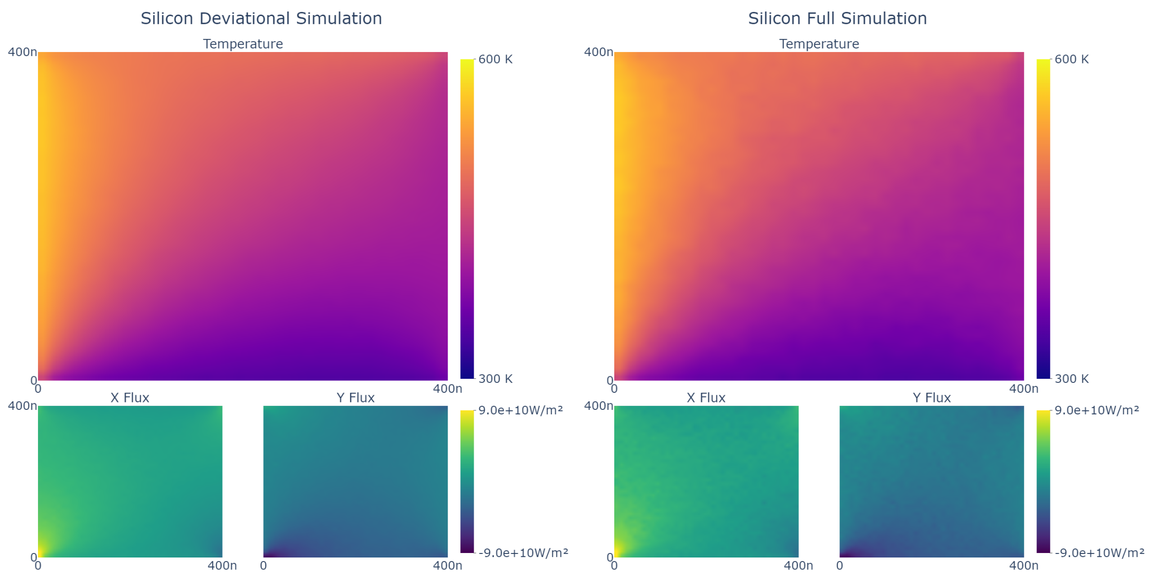

Figure 4.19: Temperature and flux profiles of a thin silicon wafer.

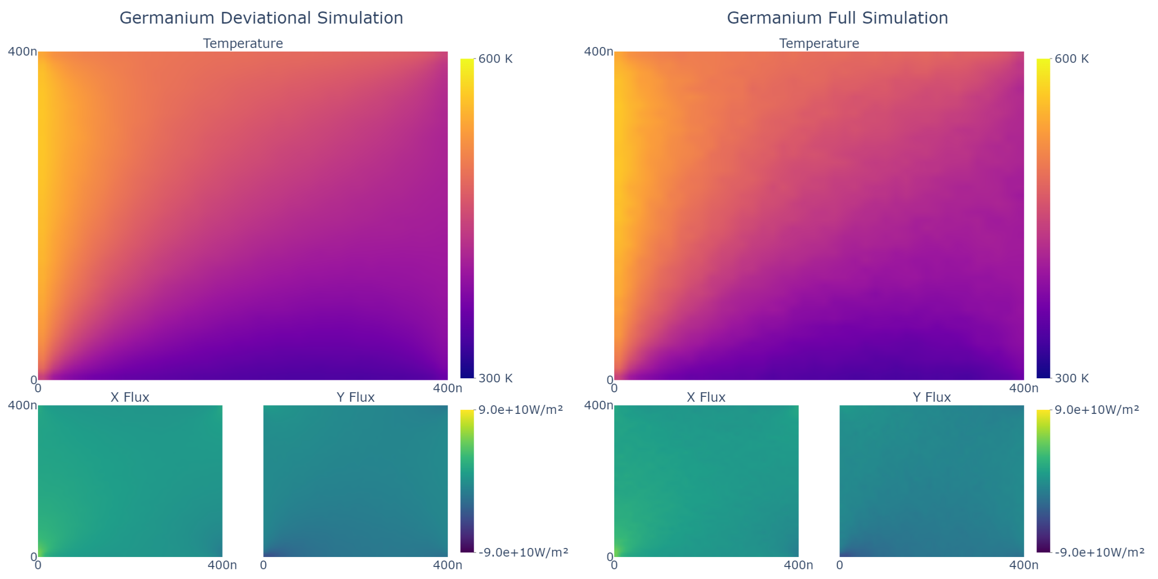

Figure 4.20: Temperature and flux profiles of a thin germanium wafer.

A direct comparison with the results from Mazumder and Majumdar [1] is difficult, but there are definite similarities in the temperature profiles. The germanium wafer exhibits less overall flux than the silicon wafer, which is expected based on the material properties. Despite the significant temperature differential across the system, there is little difference between the full and deviational simulation results.

4.4 Simulation Comparison

The full and deviational simulation types produce similar results except at low-temperature ranges and for extreme temperature differences. However, the energy and flux variance is significantly lower in the deviational simulation. For minimal temperature differences, the deviational simulation still produces coherent results, whereas those from the full simulation are unreliable due to noise effects.

In addition, the deviational simulation is generally more expedient even if an equal number of phonons are simulated. For example, the time taken to simulate the silicon wafer in Section 4.3 using the deviational approach was 239.652 seconds. The full simulation approach took 610.017 seconds. The following two factors primarily account for this difference.

- The full simulation undergoes a numerical inversion to determine the temperature in each measurement area at the end of the simulation. This numerical inversion is not required in the deviational simulation. For systems with many measurement areas, like the wafers from Section 4.3, the deviational simulation may see a significant speedup relative to the full simulation by forgoing these numerical inversions.

- The second point is more nuanced. Generally speaking, most phonons will originate via emitting surfaces for deviational simulations. For full simulations, the majority of phonons will originate in the cells. The specifics of phonon origination vary depending on the system specifications, but generally, this will be the case. Consider the linear systems from Section 4.1.3 as an example. On the first iteration, all the measurement areas are given $T_{\text{init}} = T_{eq} = (T_H + T_C)/2$. This equivalence means that, for deviational simulations, all the phonons originate from the emitting surfaces, whereas the full simulation will still have many phonons originating in the cells. This difference in phonon origination results in a massive speedup because phonons that originate from cells, on average, drift for twice as long as those that originate from emitting surfaces, as discussed in Section 3.4.6.

However, the primary performance advantage of the deviational simulation comes from the ability to produce low variance results using far fewer phonons than are required for full simulations. While no quantitative testing is done here, based on qualitative observations, the deviational simulation can generally operate with 10x fewer phonons than the full simulation leading to a direct $\sim$10x speedup without relative loss of precision.

Chapter 5

Summary

5.1 The Phonon Monte Carlo Method

This thesis details the development of computer software capable of rapidly simulating heat transport in semiconductor nanostructures. The simulation software consists of two variations of the phonon Monte Carlo method that have been implemented and tested. The first variation, referred to in this work as the full simulation approach, closely resembles the work of Lacroix, Joulain and Lemonnier [3], which is derived mainly from the work of Mazumder and Majumdar [1]. The second variation, the deviational approach, is based on the work of Peraud and Hadjiconstantinou [70].

In Chapter 4, results were shown from three types of simulations. The first simulation sets used straight germanium and silicon nanowires as benchmarks to validate the phonon Monte Carlo method's predictive capabilities concerning thermal conductivity and steady-state temperature profiles. The second set of simulations involved kinked silicon nanowires and is meant to showcase the phonon Monte Carlo method's ability to simulate systems with some degree of geometric complexity. Finally, the results from simulations using square silicon and germanium wafers are compared to results from previous works. In addition, there is a brief on the computational efficiency of the two variations.

The benchmarks simulation results show that the simulated temperature profiles largely agree with theoretical results and results from the literature, as does the predicted thermal conductivity. However, the thermal conductivity is quite sensitive to the relaxation rates that are used. The phonon Monte Carlo method definitely shows promise as a tool to simulate geometrically complex nano-scaled devices, provided the relaxation times are adjusted accordingly.

In general, the phonon Monte Carlo method presented in this study is computationally efficient, retains an intuitive connection to the problem physics and strikes a good balance between accuracy and efficiency. The deviational simulation approach should be preferred over the full simulation approach except for low-temperature ranges and possibly for systems with extreme temperature differences.

5.2 Software Details

The current software theoretically allows for unrestricted 2D geometrical configurations. Any 2D geometry with multiple heat sources and heat sinks is feasible, including porous structures and rounded edges. However, it can be difficult to specify these geometries, depending on their complexity. This specification can be done in a few lines of code for simple structures. A library of pre-built configurations was produced for all systems described in Section 4. The linear pre-built code can be found in Appendix A.7. Appendix A.8 displays the use of this pre-built configuration. In Appendix A.7, it can be seen that the material behaviour can be modified by altering the following three properties.

- The dispersion relations. The phonon Monte Carlo method requires this information apriori. No testing was done on how changing these values affects the model results.

- The relaxation rates. Changing the relaxation rates affects the number and ratio of scattering processes during the simulation. Results from Section 4.1.1 show that this input can significantly affect the model output.
- The degree of specular reflection at boundary/adiabatic surfaces. All surfaces will have the same degree of specularity for the linear pre-built system, but it is possible to customize this on a cell-by-cell basis.

The phonon Monte Carlo portion of the software is written in C++ and can perform the simulations quite rapidly. All the individual results from Section 4 were simulated in less than ten minutes, and the smaller linear systems only took a few seconds to simulate. The following four factors primarily influence the runtime of the simulation.

1. The number of phonons that are simulated. Increasing the number of phonons will increase the program runtime and reduce the variance in the results up to a point.
2. The simulation duration. Increasing the simulation duration will generally increase the program runtime. It is important to ensure that the simulation time is long enough for the system to reach a steady state.
3. The number of measurements. The more often we allow a phonon to contribute its energy and flux to a measurement area, the slower the program's runtime. The code is optimized to only record measurements in the last 10% of the measurement steps, so the negative effects of this factor are minimized.
4. The number of measurement areas/sensors. Increasing the number of measurement areas in a system will increase the program runtime. This is primarily a factor for full simulations because more measurement areas mean more numerical inversions to obtain the measurement area's temperature.

The full and deviational simulations will have similar runtimes given equal inputs unless the number of measurement areas is extreme. The primary benefit of the deviational method, in terms of performance, is that it can achieve the same variance as the full simulation using far fewer phonons. Based on qualitative observations, the deviational simulation will have the same amount of noise using 10x fewer phonons than the corresponding full simulation.

No formal study was done on the convergence speed and the effects of the above factors. However, due to the speed of the software, it was possible to use huge numbers to guarantee results with minimal variance that are easily reproducible. A more in-depth look at these factors would be helpful.

5.3 Possible Improvements

Geometrical Limitations

Allowing for unrestricted 3D geometries would considerably improve the existing model. A relatively simple intermediate step would allow the user to specify a single measurement that represents the z-axis dimension for the entire system. This would let the user set emitting surfaces and alter surface specularity in that plane. Unrestricted 3D geometries would necessitate changing the fundamental building block of the models from triangles to tetrahedrons and changing the geometric mathematics from lines intersecting lines to planes intersection planes and lines intersecting planes. Additionally, visualizing the resulting 3D structures would not be trivial.

Material Interfaces

It is straightforward to create models that are composed of various types of materials. However, the current model code does not handle interface interaction between different materials. At the time of writing, when a phonon passes from one material to another, it will backscatter if the incoming phonon is incompatible with the new material, but no other interactions are considered. These interface interactions are complex, but some promising techniques can be found in [55] and [78].

Optical Phonons

The current model does not consider the optical phonon branches. The low group velocity of optical phonons prohibits them from a significant role in thermal conductivity. However, they decay into acoustic phonons, affecting relaxation times and capacitive properties of the material [33].

Computational Performance

The simulations in Chapter 4 were run with an excessive number of phonons to ensure the results have minimal variance and are easily reproducible. Generally, this excessive number of phonons led to few iterations/evolutions being needed before the system stabilized. It may be worth exploring what happens if a smaller number of phonons and a high number of iterations are used instead. It seems probable that there is an optimal combination which minimizes variance and maximizes performance. However, this will most likely vary significantly depending on the simulated system's geometric details.

Steady-State Detection

The software currently requires the user to specify the simulation duration. A better approach would be to implement a steady-state detection mechanism that stops the program once the mechanism detects that the system has reached the steady state. This adjustment should be relatively easy to implement by stopping the simulation when the flux across each cell is constant within some margin of error for a certain amount of time.

Bibliography

[1] S. Mazumder and A. Majumdar, “Monte Carlo study of phonon transport in solid thin films including dispersion and polarization,” *J. Heat Transfer*, vol. 123, no. 4, pp. 749–759, 2001.

[2] G. Chen and A. Shakouri, “Heat transfer in nanostructures for solid-state energy conversion,” *J. Heat Transfer*, vol. 124, no. 2, pp. 242–252, 2002.

[3] D. Lacroix, K. Joulain, and D. Lemonnier, “Monte Carlo transient phonon transport in silicon and germanium at nanoscales,” *Physical Review B*, vol. 72, no. 6, p. 064305, 2005.

[4] J.-P. M. Péraud and N. G. Hadjiconstantinou, “Efficient simulation of multidimensional phonon transport using energy-based variance-reduced Monte Carlo formulations,” *Physical Review B*, vol. 84, no. 20, p. 205331, 2011.

[5] G. Chen, “Nonlocal and nonequilibrium heat conduction in the vicinity of nanoparticles,” *J. Heat Transfer*, 1996.

[6] G. Mahan and J. Sofo, “The best thermoelectric,” *Proceedings of the National Academy of Sciences*, vol. 93, no. 15, pp. 7436–7439, 1996.

[7] R. Venkatasubramanian, E. Siivola, T. Colpitts, and B. O’quinn, “Thin-film thermoelectric devices with high room-temperature figures of merit,” *Nature*, vol. 413, no. 6856, pp. 597–602, 2001.

[8] Wang *et al.*, “Enhanced thermoelectric figure of merit in nanostructured n-type silicon germanium bulk alloy,” *Applied Physics Letters*, vol. 93, no. 19, p. 193121, 2008.

[9] A. Mittal and S. Mazumder, “Monte Carlo study of phonon heat conduction in silicon thin films including contributions of optical phonons,” *Journal of Heat Transfer*, 2010.

[10] D. Jang *et al.*, “Self-heating on bulk finfet from 14nm down to 7nm node,” in *2015 IEEE International Electron Devices Meeting (IEDM)*, pp. 11–6, IEEE, 2015.

[11] D. Friedman, “International solid-state circuits conference trends 2013,” 2013.

[12] D. G. Cahill, W. K. Ford, K. E. Goodson, G. D. Mahan, A. Majumdar, H. J. Maris, R. Merlin, and S. R. Phillpot, “Nanoscale thermal transport,” *Journal of applied physics*, vol. 93, no. 2, pp. 793–818, 2003.

[13] T. S. Fisher, *Thermal energy at the nanoscale*, vol. 3. World Scientific Publishing Company, 2013.

[14] Y. Zhou, X. Zhang, and M. Hu, “Nonmonotonic diameter dependence of thermal conductivity of extremely thin si nanowires: competition between hydrodynamic phonon flow and boundary scattering,” *Nano letters*, vol. 17, no. 2, pp. 1269–1276, 2017.

[15] B. T. Wong, M. Francoeur, and M. P. Mengüç, “A Monte Carlo simulation for phonon transport within silicon structures at nanoscales with heat generation,” *International Journal of Heat and Mass Transfer*, vol. 54, no. 9-10, pp. 1825–1838, 2011.

[16] V. Khvesyuk and A. Skryabin, “Heat conduction in nanostructures,” *High Temperature*, vol. 55, no. 3, pp. 434–456, 2017.

[17] J. Callaway, “Model for lattice thermal conductivity at low temperatures,” *Physical Review*, vol. 113, no. 4, p. 1046, 1959.

[18] M. Holland, “Analysis of lattice thermal conductivity,” *Physical Review*, vol. 132, no. 6, p. 2461, 1963.

[19] J. B. J. Fourier, G. Darboux, *et al.*, *Théorie analytique de la chaleur*, vol. 504. Didot Paris, 1822.

[20] L. Geppert, “Solid state [semiconductors. 1999 technology analysis and forecast],” *IEEE Spectrum*, vol. 36, no. 1, pp. 52–56, 1999.

[21] G. Zeng, X. Fan, C. LaBounty, E. Croke, Y. Zhang, J. Christofferson, D. Vashaee, A. Shakouri, and J. E. Bowers, “Cooling power density of sige/si superlattice micro refrigerators,” *MRS Online Proceedings Library (OPL)*, vol. 793, 2003.

[22] A. Majumdar, “Microscale heat conduction in dielectric thin films,” 1993.

[23] C. Kittel and P. McEuen, *Introduction to Solid State Physics.* John Wiley & Sons, 2018.

[24] K. Raleva, A. R. Shaik, D. Vasileska, and S. M. Goodnick, *Modeling Self-Heating Effects in Nanoscale Devices.* Morgan & Claypool Publishers California, 2017.

[25] M. Asheghi, M. Touzelbaev, K. Goodson, Y. Leung, and S. Wong, “Temperature-dependent thermal conductivity of single-crystal silicon layers in soi substrates,” 1998.

[26] G. Dolling, "Lattice vibrations in crystals with the diamond structure," in *Inelastic Scattering of Neutrons in Solids and Liquids. V. II. Proceedings of the Symposium on Inelastic Scattering of Neurons in Solids and Liquids*, 1963.

[27] V. Jean, S. Fumeron, K. Termentzidis, S. Tutashkonko, and D. Lacroix, "Monte Carlo simulations of phonon transport in nanoporous silicon and germanium," *Journal of Applied Physics*, vol. 115, no. 2, p. 024304, 2014.

[28] E. Pop, R. W. Dutton, and K. E. Goodson, "Analytic band Monte Carlo model for electron transport in si including acoustic and optical phonon dispersion," *Journal of Applied Physics*, vol. 96, no. 9, pp. 4998–5005, 2004.

[29] M. F. Modest and D. C. Haworth, *Radiative heat transfer in turbulent combustion systems: theory and applications*. Springer, 2016.

[30] W. De Haas and T. Biermasz, "The thermal conductivity of quartz at low temperatures," *Physica*, vol. 2, no. 1-12, pp. 673–682, 1935.

[31] S. Barman and G. Srivastava, "Quantitative estimate of phonon scattering rates in different forms of diamond," *Physical Review B*, vol. 73, no. 7, p. 073301, 2006.

[32] M. Asheghi, K. Kurabayashi, R. Kasnavi, and K. Goodson, "Thermal conduction in doped single-crystal silicon films," *Journal of applied physics*, vol. 91, no. 8, pp. 5079–5088, 2002.

[33] S. V. Narumanchi, J. Y. Murthy, and C. H. Amon, "Submicron heat transport model in silicon accounting for phonon dispersion and polarization," *J. Heat Transfer*, vol. 126, no. 6, pp. 946–955, 2004.

[34] S. Barman and G. Srivastava, "Lifetime of nonequilibrium zone-center longitudinal optical phonons in zinc-blende materials," *Applied Physics Letters*, vol. 81, no. 18, pp. 3395–3397, 2002.

[35] C. Herring, "Role of low-energy phonons in thermal conduction," *Physical Review*, vol. 95, no. 4, p. 954, 1954.

[36] A. Ward and D. Broido, "Intrinsic phonon relaxation times from first-principles studies of the thermal conductivities of si and ge," *Physical Review B*, vol. 81, no. 8, p. 085205, 2010.

[37] M. Maldovan, "Micro to nano scale thermal energy conduction in semiconductor thin films," *Journal of Applied Physics*, vol. 110, no. 3, p. 034308, 2011.

[38] C. Dames and G. Chen, "Theoretical phonon thermal conductivity of si/ge superlattice nanowires," *Journal of Applied Physics*, vol. 95, no. 2, pp. 682–693, 2004.

[39] G. Srivastava, "The physics of phonons. 1990," *Bristol, UK: Adam Hilger*.

[40] J. A. Pascual-Gutiérrez, J. Y. Murthy, and R. Viskanta, "Thermal conductivity and phonon transport properties of silicon using perturbation theory and the environment-dependent interatomic potential," *Journal of Applied Physics*, vol. 106, no. 6, p. 063532, 2009.

[41] P. Carruthers, "Theory of thermal conductivity of solids at low temperatures," *Reviews of Modern Physics*, vol. 33, no. 1, p. 92, 1961.

[42] L. Desvillettes, C. Mouhot, and C. Villani, "Celebrating Cercignani's conjecture for the Boltzmann equation," *arXiv preprint arXiv:1009.4006*, 2010.

[43] C. Moglestue, "Monte Carlo particle modelling of small semiconductor devices," *Computer Methods in Applied Mechanics and Engineering*, vol. 30, no. 2, pp. 173–208, 1982.

[44] C. Jacoboni and L. Reggiani, "The Monte Carlo method for the solution of charge transport in semiconductors with applications to covalent materials," *Reviews of modern Physics*, vol. 55, no. 3, p. 645, 1983.

[45] P. Lugli, P. Bordone, L. Reggiani, M. Rieger, P. Kocevar, and S. Goodnick, "Monte Carlo studies of nonequilibrium phonon effects in polar semiconductors and quantum wells. i. laser photoexcitation," *Physical Review B*, vol. 39, no. 11, p. 7852, 1989.

[46] M. V. Fischetti and S. E. Laux, "Monte Carlo analysis of electron transport in small semiconductor devices including band-structure and space-charge effects," *Physical Review B*, vol. 38, no. 14, p. 9721, 1988.

[47] M. V. Fischetti and S. E. Laux, "Monte Carlo study of electron transport in silicon inversion layers," *Physical Review B*, vol. 48, no. 4, p. 2244, 1993.

[48] G. Chen and C. Tien, "Thermal conductivities of quantum well structures," *Journal of thermophysics and heat transfer*, vol. 7, no. 2, pp. 311–318, 1993.

[49] K. E. Goodson, "Thermal conduction in nonhomogeneous cvd diamond layers in electronic microstructures," 1996.

[50] G. Chen, "Thermal conductivity and ballistic-phonon transport in the cross-plane direction of superlattices," *Physical Review B*, vol. 57, no. 23, p. 14958, 1998.

[51] R. Peterson, "Direct simulation of phonon-mediated heat transfer in a debye crystal," 1994.

[52] Y. Chen, D. Li, J. R. Lukes, and A. Majumdar, "Monte Carlo simulation of silicon nanowire thermal conductivity," 2005.

[53] T. Wang, *Sub-micron thermal transport in ultra-scaled metal oxide semiconductor (MOS) devices.* PhD thesis, Purdue University, 2007.

[54] D. Terris, K. Joulain, D. Lemonnier, and D. Lacroix, "Modeling semiconductor nanostructures thermal properties: The dispersion role," *Journal of Applied Physics*, vol. 105, no. 7, p. 073516, 2009.

[55] A. J. Minnich, G. Chen, S. Mansoor, and B. Yilbas, "Quasiballistic heat transfer studied using the frequency-dependent Boltzmann transport equation," *Physical Review B*, vol. 84, no. 23, p. 235207, 2011.

[56] J. M. Hammersley and D. C. Handscomb, "General principles of the Monte Carlo method," in *Monte Carlo Methods*, pp. 50–75, Springer, 1964.

[57] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[58] A. Z. Panagiotopoulos, "Direct determination of fluid phase equilibria by simulation in the gibbs ensemble: a review," *Molecular simulation*, vol. 9, no. 1, pp. 1–23, 1992.

[59] G. Chen, "Ballistic-diffusive equations for transient heat conduction from nano to macroscales.," *J. Heat Transfer*, vol. 124, pp. 320–328, 2002.

[60] D. Ceperley, M. Mascagni, and A. Srinivasans, "A scalable library for pseudorandom number generation.," *ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE*, vol. 26, pp. 436–461, 2000.

[61] C. Bera, "Monte Carlo simulation of thermal conductivity of si nanowire: An investigation on the phonon confinement effect on the thermal transport," *Journal of Applied Physics*, vol. 112, no. 7, p. 074323, 2012.

[62] Q. Hao, G. Chen, and M.-S. Jeng, "Frequency-dependent Monte Carlo simulations of phonon transport in two-dimensional porous silicon with aligned pores," *Journal of Applied Physics*, vol. 106, no. 11, p. 114321, 2009.

[63] O. Muscato, "Relaxation-time approximations to the Boltzmann equation for electron transport in bulk silicon," *Physica A: Statistical Mechanics and its Applications*, vol. 317, no. 1-2, pp. 113–128, 2003.

[64] K. Kukita and Y. Kamakura, "Monte Carlo simulation of phonon transport in silicon including a realistic dispersion relation," *Journal of Applied Physics*, vol. 114, no. 15, p. 154312, 2013.

[65] A. J. McGaughey and A. Jain, "Nanostructure thermal conductivity prediction by Monte Carlo sampling of phonon free paths," *Applied Physics Letters*, vol. 100, no. 6, p. 061911, 2012.

[66] L. Chaput, "Direct solution to the linearized phonon Boltzmann equation," *Physical Review letters*, vol. 110, no. 26, p. 265506, 2013.

[67] K. Esfarjani, G. Chen, and H. T. Stokes, "Heat transport in silicon from first-principles calculations," *Physical Review B*, vol. 84, no. 8, p. 085204, 2011.

[68] K. Termentzidis, P. Chantrenne, and P. Keblinski, "Nonequilibrium molecular dynamics simulation of the in-plane thermal conductivity of superlattices with rough interfaces," *Physical Review B*, vol. 79, no. 21, p. 214307, 2009.

[69] S. Hepplestone and G. Srivastava, "Lattice dynamics and thermal properties of phononic semiconductors," *Physical Review B*, vol. 84, no. 11, p. 115326, 2011.

[70] J.-P. M. Péraud and N. G. Hadjiconstantinou, "An alternative approach to efficient simulation of micro/nanoscale phonon transport," *Applied Physics Letters*, vol. 101, no. 15, p. 153114, 2012.

[71] Z. Aksamija and I. Knezevic, "Anisotropy and boundary scattering in the lattice thermal conductivity of silicon nanomembranes," *Physical Review B*, vol. 82, no. 4, p. 045319, 2010.

[72] N. Roberts and D. G. Walker, "Computational study of thermal rectification from nanostructured interfaces," *Journal of Heat Transfer*, vol. 133, no. 9, 2011.

[73] Y. Chalopin, K. Esfarjani, A. Henry, S. Volz, and G. Chen, "Thermal interface conductance in si/ge superlattices by equilibrium molecular dynamics," *Physical Review B*, vol. 85, no. 19, p. 195302, 2012.

[74] Y. Chalopin, A. Rajabpour, H. Han, Y. Ni, and S. Volz, "Equilibrium molecular dynamics simulations on interfacial phonon transport," *Annual Review of Heat Transfer*, vol. 17, 2014.

[75] S. Sadasivam, Y. Che, Z. Huang, L. Chen, S. Kumar, and T. S. Fisher, "The atomistic green's function method for interfacial phonon transport," *Annual Review of Heat Transfer*, vol. 17, 2014.

[76] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape distributions," *ACM Transactions on Graphics (TOG)*, vol. 21, no. 4, pp. 807–832, 2002.

[77] J.-P. M. Péraud, C. D. Landon, and N. G. Hadjiconstantinou, "Monte Carlo methods for solving the Boltzmann transport equation," *Annual Review of Heat Transfer*, vol. 17, 2014.

[78] Y.-C. Hua and B.-Y. Cao, "Study of phononic thermal transport across nanostructured interfaces using phonon Monte Carlo method," *International Journal of Heat and Mass Transfer*, vol. 154, p. 119762, 2020.

Appendix A

Simulation Code

Code Listing A.1: simulatePhonon

```
void ModelSimulator::simulatePhonon(Phonon&& p, std::size_t measurement_steps) const {
    bool phonon_alive = true;
    double phonon_age = p.getLifetime();
    auto step = static_cast<std::size_t>(phonon_age / step_time_);
    p.setLifeStep(step); // For transient simulations
    Phonon::RelaxRates relax_rates{};
    double time_to_scatter = 0.;
    double time_to_measurement = 0.;

    auto get_scatter_info = [&step](const Phonon& p) {
        const auto relax_rates = p.getRelaxRates(step);
        return std::make_pair(relax_rates,
                              SCALING_FACTOR * -log(Utils::urand()) /
                                 std::accumulate(std::cbegin(relax_rates),
                                                                          std::cend(relax_rates), 0.));
    };

    while (phonon_alive) {
        // If the phonon has scattered on the previous iteration recalculate new scattering
           rates and
        // find the time to the next scattering event
        if (time_to_scatter <= 0.) {
            std::tie(relax_rates, time_to_scatter) = get_scatter_info(p);
        }
        // If a measurement event occurred -> find the time to the next measurement event
        if (time_to_measurement <= 0.) {
            time_to_measurement = step_times_[step] - phonon_age;
        }
        auto drift_time = std::min(time_to_scatter, time_to_measurement); // Drift time until
           next non-impact event
        const auto sensor_id = p.getCellSensorID();
        // drifted_time is how long the phonon drifts before an impact event
        // Will be equal to drift_time if there is no impact
        // Will be false/null if the phonon impacts an emitting surface - signals it should be
            removed from system
        const std::optional<double> drifted_time = handleImpacts(p, drift_time, sensor_id);

        if (drifted_time) { // If the phonon had a transition/boundary surface collision
            // If the phonon has transitioned to a new sensor area (impact with transition
                surface)
            // Adjust drift_time to reflect there may be additional impacts but, first we need
                to find
            // a new scattering time before continuing
            if (p.getCellSensorID() != sensor_id) {
                // reduce drift_time to the amount of time the phonon has drifted, so we can
                    start
                // the process over with a fresh scattering time as we have entered a different
                    sensor area
                // i.e. old time_to_scatter is no longer valid
                drift_time = *drifted_time;
            }
            p.drift(drift_time-*drifted_time);
            phonon_age += drift_time;
            time_to_measurement -= drift_time;
            time_to_scatter -= drift_time;
            if (time_to_measurement == 0.) { // Take a measurement
                if (++step < measurement_steps) { // Simulation time has not been exceeded
                    p.setLifeStep(step);
                    if (step >= step_adjustment_) {
                        p.updateCellHeatParams(step - step_adjustment_);
                    }
                } else { // Exceeds simulation time
                    phonon_alive = false;
                }
            } else if (!phasor_sim_ && time_to_scatter == 0.) {
                scatter(p, relax_rates);
            } else { // This is the condition when the phonon transitions to a new sensor area
                time_to_scatter = 0.; // Reset scattering time based on new sensor area
                    properties
            }
        } else { // Phonon made impact with an emitting surface (left system)
            phonon_alive = false;
        }
    }
}
```

Code Listing A.2: handleImpacts

```
// returning 0 means the calling function will drift the phonon for drift_time (it does not
   drift here)
// returning a number will reduce the amount of time the calling function drifts the phonon
   by that amount
// The next impact method places the phonon on the poi of the impacted surface effectively
   drifting it for impact_time
// The higher drifted time is here, the less amount of time the calling function drifts the
   phonon
// returning std::nullopt will kill the phonon
std::optional<double> ModelSimulator::handleImpacts(Phonon& p, double drift_time, std::size_t
   sensor_id) const {
    auto impact_time = nextImpact(p, drift_time);
    double drifted_time = 0.;
    std::size_t collision_counter = 0;
    // Impact with an emitting surface will set the phonon cell to nullptr
    while (impact_time) {
        if (p.outsideCell()) { return std::nullopt; }
        drifted_time += *impact_time;
        // If phonon is stuck, move it to a random location in the cell - primarily used to
           handle FP issues
        if (++collision_counter > MAX_COLLISIONS) {
            p.setRandPoint(Utils::urand(), Utils::urand());
            return std::make_optional<double>(drift_time); // calling function will not further
               drift the phonon
        }
        // If the phonon has changed sensor areas - return immediately as scatter time must be
           reset
        const auto cur_sensor_id = p.getCellSensorID();
        if (sensor_id != cur_sensor_id ) {
            return std::make_optional<double>(drifted_time);
        }
        impact_time = nextImpact(p, drift_time - drifted_time);
    }
    return (p.outsideCell()) ? std::nullopt : std::make_optional<double>(drifted_time);
}
```

Code Listing A.3: nextImpact

```
std::optional<double> ModelSimulator::nextImpact(Phonon& p, double time) const noexcept {
    // Get some necessary information
    const auto& [px, py] = p.getPosition();
    const auto& [vx, vy] = p.getVelVector();
    const Point start_point{px, py};
    const Point end_point{px+time*vx, py+time*vy};
    const auto boundaryLines = p.getCellBoundaryLines();
    if (start_point == end_point) {
        return std::nullopt;
    }
    const Line phonon_path{start_point, end_point};

    auto getTime = [](double start_coord, double end_coord, double velocity, double max_time) {
        return (velocity > VELOCITY_EPS || velocity < -VELOCITY_EPS) ? (end_coord -
            start_coord) / velocity : max_time;
    };

    // Find the nearest impact time and corresponding impact point
    std::optional<Point> impact_point = std::nullopt;
    for (const auto& line : boundaryLines) {
        // If there is a point of intersection that is not the start point
        if (const auto poi = line.getIntersection(phonon_path); poi && (*poi != start_point)) {
            // If the time taken to hit this POI is <= previous shortest time -> store POI and
                time
            const auto impact_time_x = getTime(start_point.x, (*poi).x, vx, time);
            const auto impact_time_y = getTime(start_point.y, (*poi).y, vy, time);
            const auto impact_time = (impact_time_x <= impact_time_y) ? impact_time_x :
                impact_time_y;
            if (impact_time <= time) {
                time = impact_time;
                impact_point = poi;
            }
        }
    }
    if (impact_point) {
        p.setPosition((*impact_point).x, (*impact_point).y);
        p.handleSurfaceCollision(*impact_point, step_time_);
        return std::make_optional(time);
    }
    return std::nullopt;
}
```

Code Listing A.4: scatter

```
void ModelSimulator::scatter(Phonon& p, const Phonon::RelaxRates& relax_rates) noexcept {
    const auto [tau_N_inv, tau_U_inv, tau_I_inv] = relax_rates;
    const double tau_inv = std::accumulate(std::cbegin(relax_rates), std::cend(relax_rates),
        0.);
    const double rand = Utils::urand();
    if (rand <= (tau_N_inv + tau_U_inv) / tau_inv) { // Not an impurity scatter
        // Resample the new phonon (freq, vel & polarization)
        p.scatterUpdate();
        if (rand > tau_N_inv / tau_inv) { // Umklapp scatter -> change direction vector
            p.setRandDirection(Utils::urand(), Utils::urand());
        }
    } else if (tau_I_inv > 0.) { // Impurity scatter
```

```
        p.setRandDirection(Utils::urand(), Utils::urand());
    }
}
```

Code Listing A.5: updateHeatParams

```
void Sensor::updateHeatParams(const Phonon& p, std::size_t step) noexcept {
    const auto sign = p.getSign();
    const auto& [vx, vy] = p.getVelVector();
    std::scoped_lock lg(*updateMutex_);
    inc_energy_[step] += sign;
    // Track net velocities in each cell for flux calculations
    auto& v = inc_flux_[step];
    v[0] += vx * sign;
    v[1] += vy * sign;
}
```

Code Listing A.6: findTemperature

```
std::vector<double> SensorInterpreter::findTemperature(const Sensor& sensor, std::size_t
   start_step) const noexcept {
    const auto& energies = sensor.getEnergies();
    auto start = std::cbegin(energies)+start_step; auto end = std::cend(energies);
    std::vector<double> temps(std::distance(start, end));
    const auto& material = sensor.getMaterial();
    const auto area = sensor.getArea();

    auto inversion = [&](double current_energy, bool pseudo=false) {
        double temp = 0., de, ub = ub_, lb = lb_;
        std::size_t iter = 0;
        while ( (ub - lb >= EPS) && (++iter != MAX_ITERS) ) {
            temp = (ub + lb) / 2.;
            de = (material.theoreticalEnergy(temp, pseudo) * area) - current_energy;
            (de < 0.) ? lb = temp : ub = temp;
        }
        return temp;
    };

    std::transform(std::execution::seq, start, end, std::begin(temps), [&, index=0](const
       auto& energy_units) mutable {
        // Multiply the number of energy units by the phonon effective energy to get the total
           energy at each measurement step
        const double energy = eff_energy_ * energy_units;
        if (t_eq_ != 0.) { // Do approximation to find the temperature
            // If it is a steady-state simulation, the index will be disregarded when finding
               the heat capacity
            return energy / (area * sensor.getHeatCapacity(index++)) + t_eq_;
        }
        else { // Do numerical inversion
            return inversion(energy);
        }
    });
    return temps;
}
```

Code Listing A.7: pre_builts.py

```
from .builder_tools import ModelBuilder
from .builder_tools import DispersionData
from .builder_tools import RelaxationData
from .builder_tools import Material

# Data from Jean2014 AIP Appendix
# Silicon data - B_i = 1.2e-45
sd_data = DispersionData((-2.22e-7, 9.26e3, 0.), 7.63916048e13,
                         (-2.28e-7, 5.24e3, 0.), 3.0100793072e13)
sr_data = RelaxationData(1.3e-24, 9.0e-13, 1.9e-18, 0., 2.42e13)
silicon = Material("Silicon", sd_data, sr_data)

def simple_linear(num_cells: int, t_high: float, t_low: float, t_init: float, t_eq: float,
                  x_base: float, y_base: float, spec: int=1, sim_type: int=0,
                  step_interval: int=0) -> ModelBuilder:
    b = ModelBuilder()
    # Specify general model settings
    b.setSimType(sim_type)
    b.step_interval = step_interval
    b.t_eq = t_eq
    # The model will be comprised of a single material
    b.addMaterial(silicon)

    avg_temp = (t_high + t_low) / 2
    # Create the cell component
    for i in range(num_cells):
        # Sensor must be added first
        s_id = b.addSensor(silicon.name, avg_temp)
        lower_left_point = (i*x_base, 0.)
        top_right_point = ((i+1)*x_base, y_base)
        # The cell must be attached to a sensor
        b.addRectangularCell(lower_left_point, top_right_point, s_id, spec)
    # Add left side surface
    b.addSurface((0., 0.), (0., y_base), t_high)
    # Add right side surface
    b.addSurface((num_cells*x_base, 0.), (num_cells*x_base, y_base), t_low)

    return b
```

Code Listing A.8: linear_demo.py

```
from psim import pre_builts

# Simple linear prebuilt construction
num_cells = 40
t_high = 310
t_low = 290
t_init = 300
t_eq = 300 # 0 here indicates a 'full' simulation
cell_x_len = 50
cell_y_len = 50

b = pre_builts.simple_linear(num_cells, t_high, t_low, t_init, t_eq, cell_x_len, cell_y_len)
b.setMeasurements(1000)
b.setSimTime(20)
b.setNumPhonons(1000000)

filename = 'linear_demo.json'
b.export(filename)
```

Appendix B

Sample JSON Input File

The following code produces a 100 nm $\times$ 50 nm nanowire. A 50 nm high, 310 K emitting surface is placed at $x = 0$, and a 50 nm high, 290 K emitting surface is placed at the $x = 100$. The length of the z-dimension is arbitrary. See Section 5.3 for more details. All surfaces are perfectly specular. There are four triangular cells linked to two sensors making each measurement area a square. The 'length' characteristic for the emitting surfaces is used to sort the surface. The actual length is the square root of the displayed value.

The system is comprised of silicon, and the dispersion relation data is given by the 'd_data' dictionary, with the LA and TA branches being the 'la_data' and 'ta_data' arrays, respectively. The relaxation rates are given in the 'r_data' dictionary. See Table 2.1 for more details.

The 'sim_type,' 'step_interval,' and 'phasor_sim' fields are not relevant to the work described in this thesis. This is also the case for the 'duration' and 'start_time' fields in the 'emit_surfaces' dictionary. For the simulation types described in this work, the 'duration' field will always be the same as the 'sim_time' field. All the other fields mentioned in this paragraph should be set to 0 or false in the case of the 'phasor_sim' field.

No formal work has been done on the optimal inputs for the 'num_measurements,' 'sim_time,' and 'num_phonons' fields. Due to the speed of the simulations, the parameters were often set to be much larger than necessary based on results from previous simulations. A potential upper limit for the number of phonons would be that number calculated from Eq. (2.14). A starting point for the simulation time could be the time it takes for the slowest moving phonon to traverse the characteristic dimension of the system. For more complex geometries, it may be necessary to use trial and error here or improve the algorithm so the simulation detects when it has reached the steady state. The number of measurements is complete guesswork at this point. In most cases, the numbers used in this work are generally much higher than necessary to produce results with minimal variance.

This configuration is for demonstration purposes. It should be subdivided to contain several more cells and sensors to obtain useful measurements. A Python program generates these files.

```
{
    "settings": {
        "num_measurements": 1000,
        "sim_time": 2,
        "num_phonons": 100000,
        "t_eq": 300,
        "sim_type": 0,
        "step_interval": 0,
        "phasor_sim": false
    },
    "materials": [
        {
            "name": "Silicon",
            "d_data": {
                "la_data": [
                    -2.22e-07,
                    9260.0,
                    0.0
                ],
                "max_freq_la": 76391604800000.0,
                "ta_data": [
                    -2.28e-07,
                    5240.0,
                    0.0
                ],
                "max_freq_ta": 30100793072000.0
            },
            "r_data": {
                "b_l": 1.3e-24,
                "b_tn": 9e-13,
                "b_tu": 1.9e-18,
                "b_i": 0.0,
                "w": 24200000000000.0
            }
        }
    ],
    "sensors": [
        {
            "id": 0,
            "material": "Silicon",
            "t_init": 300.0
        },
        {
            "id": 1,
            "material": "Silicon",
            "t_init": 300.0
        }
    ],
    "cells": [
        {
            "triangle": {
                "p1": {
                    "x": 0,
                    "y": 0.0
                },
                "p2": {
                    "x": 0,
                    "y": 50
                },
                "p3": {
                    "x": 50,
                    "y": 0.0
                }
            },
            "sensorID": 0,
            "specularity": 1
        },
        {
            "triangle": {
                "p1": {
                    "x": 50,
                    "y": 50
                },
                "p2": {
                    "x": 50,
                    "y": 0.0
                },
                "p3": {
                    "x": 0,
                    "y": 50
                }
            },
            "sensorID": 0,
            "specularity": 1
        },
        {
            "triangle": {
                "p1": {
                    "x": 50,
                    "y": 0.0
                },
                "p2": {
                    "x": 50,
                    "y": 50
                },
                "p3": {
                    "x": 100,
                    "y": 0.0
                }
            },
            "sensorID": 1,
            "specularity": 1
        },
        {
            "triangle": {
                "p1": {
                    "x": 100,
                    "y": 50
                },
                "p2": {
                    "x": 100,
                    "y": 0.0
                },
                "p3": {
                    "x": 50,
                    "y": 50
                }
            },
            "sensorID": 1,
            "specularity": 1
        }
    ],
    "emit_surfaces": [
        {
            "p1": {
                "x": 0.0,
                "y": 0.0
            },
            "p2": {
                "x": 0.0,
                "y": 50
            },
            "temp": 310,
            "duration": 2,
            "start_time": 0.0,
            "length": 2500.0
        },
        {
            "p1": {
                "x": 100,
                "y": 0.0
            },
            "p2": {
                "x": 100,
                "y": 50
            },
            "temp": 290,
            "duration": 2,
            "start_time": 0.0,
            "length": 2500.0
        }
    ]
}
```