Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

Mats Estensen

Simulation of a Mercury Cadmium Telluride Avalanche Photodiode

A Monte Carlo and Finite Element Method Approach

Master's thesis in Nanotechnology
Supervisor: Jon Andreas Støvneng (IFY), Trond Brudevoll (FFI)
March 2019

NTNU
Norwegian University of
Science and Technology

Mats Estensen

Simulation of a Mercury Cadmium Telluride Avalanche Photodiode

A Monte Carlo and Finite Element Method Approach

Master's thesis in Nanotechnology
Supervisor: Jon Andreas Støvneng (IFY), Trond Brudevoll (FFI)
March 2019

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

NTNU
Norwegian University of
Science and Technology

Abstract

An existing simulator software developed at the Norwegian Defense Research Establishment for simulating charge carrier transport phenomena in semiconductor materials has been further tested and developed in this thesis work. With applications to midwave-infrared sensors used in the defense industry, a mercury cadmium telluride avalanche photodiode has been used as an example device. The 3D simulation model is based on the Monte Carlo Method for carrier transport simulation and the Finite Element Method for discretization of the device. Issues and errors in the simulation model and its implementation have been investigated to improve the simulator further with regards to stability and efficiency of simulations.

Sammendrag

En eksisterende simulatorprogramvare utviklet ved Forsvarets Forskningsinstitutt for simulering av ladningsbærertransportfenomen i halvledermaterialer har blitt testet og videreutviklet i arbeidet med denne oppgaven. Med anvendelser for midtinfrarødsensorer i forsvarsindustrien har en kvikksølv-kadmium-tellurid avalanche fotodiode blitt brukt som halvlederkomponent i simuleringen. 3D-simuleringsmodellen er basert på Monte Carlo-metoden for ladningsbærertransport og Finite Element-metoden for diskretisering av komponenten. Feil og utfordringer i simuleringsmodellen og dens implementasjon har blitt påvist og utforsket for å forbedre simulatoren med hensyn på stabilitet og effektivitet i simuleringer.

"Our imagination is stretched to the utmost, not, as in fiction, to imagine things which are not really there, but just to comprehend those things which are there."

Richard P. Feynman, *The Character of Physical Law* (1965)

Acknowledgements

This thesis marks the conclusion of a Master of Science degree in Nanotechnology at the Norwegian University of Science and Technology (NTNU) which started in 2012. The work in this thesis was done during October 2018 – February 2019, without any prior specialization project ("project thesis").

The work was mainly done at the Norwegian Defense Research Establishment (FFI) at Kjeller under supervision of Researcher Trond Brudevoll together with Researcher Asta-Katrine Storebø. A sincere acknowledgment goes out to them for welcoming me to FFI and supervising my work along the way. Despite my challenges during this work their encouragement, inputs and discussions are greatly appreciated. Thank you to Jon Andreas Støvneng at NTNU for accepting to be NTNU's supervisor for this thesis.

The software used in this work was further developed from several previous master students from NTNU and other universities who collaborated with FFI, most recently Siri N. Fatnes in 2017. I would like to thank her for personally introducing me to the simulator, as well as providing extensive documentation in her work. This thesis is a direct continuation of her excellent work.

I would like to thank my current employer who granted me some months of academic leave from my full-time job and encouraged me in the period of working with this thesis. Thank you to Brit Wenche Meland and the other study counselors at NTNU for making it possible for me to complete this thesis, despite working remotely from Oslo/Kjeller, unfortunately being delayed with my studies after a one-year leave from the university.

Last, but not least, a big thank you goes out to my friends, family and girlfriend for their continuing support and encouragement throughout the years.

I hereby state that I have acknowledged all my sources of help and where I have used work of others which is not mentioned here, this is clearly stated throughout the text.

Mats Estensen
Oslo, March 25, 2019

Contents

List of Figures

List of Abbreviations

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
APD	avalanche photodiode
CPU	central processing unit
FD	Finite Difference
FEM	Finite Element Method
FFI	Norwegian Defence Research Establishment
HgCdTe	mercury cadmium telluride
LPE	liquid phase epitaxy
MBE	molecular beam epitaxy
MC	Monte Carlo
MCFEM	Monte Carlo software with finite element Poisson solver
NTNU	The Norwegian University of Science and Technology
PDE	partial differential equation
PLA	point location algorithm

Chapter 1

Introduction

Since the mid-19th century when the laws of electromagnetism were first described, the development of electrical components quickly gained wide traction as a promising technology for engineering advanced and useful devices. Throughout the 20th century further progress led to the use of different semiconductor materials which quickly were utilized for detectors and receivers [1]. This development coupled with a more fundamental understanding of the natural phenomenas of electromagnetism and light has later given us all the digital tools and components we use today – from the simplest single resistor to the most advanced supercomputers in the world.

In a the field of study where electricity and light are of interest, the main characters are the current-bearing electron and the light-bearing photon. Only through the comprehension of their nature to the atomic level can we harness their energy and produce useful currents and signals utilized in advanced devices. Among these devices, the semiconductor detector – more specifically the photodetector – for infrared detection is of interest in this thesis.

Semiconductor materials are widely used in imaging devices due to the photoelectric effect. The transferred energy from incident photons generate a electrical carrier in the semiconductor which can be transported in the material and extracted as current and read out as an output signal [2].

Various forms of image sensors can be found everywhere; in modern cell phones, advanced digital cameras, surveillance cameras, hyper-spectral apparatuses and much more [3]. They consist of diodes produced by using semiconductor materials that absorb the photons the sufficient energy for the desired imaging purpose. These diodes are often assembled together to form an one-dimensional (1D) or two-dimensional (2D) arrays to process signal from several similar or unique diodes together, where the signal can be read out as an image consisting of pixels which correlates to the number of diodes on the chip.

Simulation of Devices

Prior to manufacturing new devices it is necessary to have a design based on the desired physical properties which corresponds to the imaging object(s) of interest. As prototyping actual devices in a lab can be both expensive in time and cost, computer simulation allows to do this more rapidly and save development and production costs [4]. Computer simulations of electronic devices has since the late 60s [5] been used to model and understand the properties of various designs, allowing for greater understanding of the underlying physics. This offers significant insight into device physics beyond that which is possible to calculate analytically and observe in experiments, which can be used to create new – as well as further develop – device designs.

As users of some of the most advanced photodetectors, the defense industry have significantly contributed to the development of this technology through their research contributions over several decades [6]. They continue to utilize this technology through specialized applications including LIDARs, laser range-finders and other detectors in their military equipment.

Previous work at FFI

The Norwegian Defence Research Establishment (FFI), Norway's national research organization for the defense sector, have since end of 1962 researched and developed electronic devices for defense and military applications [7]. Their semiconductor lab, Epitek laboratory, has since 1990 developed and characterized semiconductor materials used for infrared detector applications [8]. One of the applications is the detector used in the infrared homing system in the sea-target missile *Naval Strike Missile*, produced by *Kongsberg Defence & Aerospace* [9].

A prime research effort at Epitek is the processing and characterization of molecular beam epitaxy (MBE)- and liquid phase epitaxy (LPE)-grown mercury cadmium telluride (HgCdTe) devices [10–12]. The experimental work is carried out in close cooperation with physicists working on modeling and simulation of such materials [13–16], where the majority of the work has been related to avalanche photodiode (APD).

FFI Monte Carlo Software

To simulate the devices the researchers at FFI have developed a state-of-the-art Monte Carlo (MC) simulator for charge transport and electro-optic applications [17]. The simulator can run in a *full-band* mode where it reads external band data or in an *analytical* mode where it calculates the bands for the holes and electrons. It contains simulation of several scattering and carrier interaction mechanisms, a 2D Finite Difference (FD) and Finite Element Method (FEM) Poisson equation solver and is currently adapted to a APD simulation scenario.

The software is programmed from scratch component-wise, such that the the program can be modified to use different modules on their own (e.g. only Poisson solver) or coupled together (e.g. Monte Carlo simulation without Poisson solver). This is a deliberate choice to create a program that is compatible with extensions that may be developed in the future (e.g. a new equation solver). This does not limit the program to be only used for simulating HgCdTe APDs, but also for other materials and devices which may be of interest in the future. An example is III-V antimonide-based photodetectors and other low-dimensional structures, e.g. quantum well and superlattice infrared photodetectors, which are being researched as possible successors to HgCdTe devices to reduce cost and increase detector sensitivity [18, 19].

Since 2007 there has been an ongoing simulator project in parallel initiated by summer interns and other graduate students who have worked with FFI in connection with their Master's thesis projects. Over these years, these two simulators have been developed separated, but some parts of the two simulators have been implemented in each other. By developing new implementations the student simulator has grown in both size and complexity to its current form. As different students have investigated different topics, their individual contributions have often been on specific parts of the software [20–26].

In 2017-2018, Siri N. Fatnes' work resulting in her master's thesis [26] led to a major refactoring of the software, introducing a more modular approach to the software as well as utilizing more of Fortran's more modern built-in functions and types, as well as improved the building flow for the software. The physical and mathematical improvements included integration of a new three-dimensional (3D) Poisson equation solver and a new implementation of a 2D and 3D point location algorithm (PLA) with a new triangulation scheme, resulting in an improved handling of the carrier injections in the simulations. Due to limited time, few and short simulations were carried out but the resulting work laid the foundation for further development and testing of the simulator.

Objectives and Approach

Based on the background given in the previous sections, the aim for this thesis is to continue the work initiated by the students on the simulator software – mainly Fatnes' recent work, as mentioned in the previous paragraph.

Despite many new implementations and improvements to the simulator several issues needs to be investigated and corrected, with a focus on carrier transport and device physics. The aforementioned lack of longer simulations are mainly obstructed by a unphysical behavior of the charged particles briefly mentioned in her thesis [26]. The high memory and central processing unit (CPU) requirements, as well as the use of a highly discretized mesh with quadratic elements also makes it costly compute-wise to perform longer simulations.

As the work in this thesis was not preceded by a project report/specialization subject, the scope will be limited to investigate and mitigate current bugs and improving current

implementation, as well as carry out new simulations of HgCdTe APDs. Larger efforts that requires significant changes across several modules in the code like introducing parallelization of the code and applying consistent scaling for all equations will therefore not be prioritized.

The main priorities are the following:

- Investigate existing errors and bugs in the simulator and continue cleaning up sections of the code to allow for further development of new modules
- Optimize simulator to further reduce memory and CPU requirements to allow for easier and faster simulations runs
- Generate optimized meshes with increased accuracy in desired regions to allow for finer output, faster solving and keeping noise levels low
- Carry out new simulations based on the new improvements and compare results with most recent achievements from the previous author

Thesis Structure

This thesis follows a classical scientific format, to give the reader a self-contained and logical presentation of the work that was done with a citation list found at the very end the thesis, for further reference.

Chapter 2 gives the reader theoretical background on the topics relevant to the simulator model used in the simulator. The model and the implementation of the simulator is presented in Chapter 3. The development and results are located in Chapter 4 along with discussion and an outline for further work. The work is finally concluded in Chapter 5.

As the work contained in this thesis builds upon development and usage of a simulation software developed at FFI the source code, compiled binaries and its full documentation is not extensively referenced or documented in this text. If the reader wishes to review the source code or have questions regarding the software, the author can be contacted by e-mail[1]. An existing helpful and rather up-to-date reference for the code written by Fatnes can be found is *Appendix B. Overview of Source Code* in [26].

[1] matsest [a] mxe.no

Chapter 2

Theory

This chapter gives a general introduction to the underlying physics relevant for semiconductor modeling, without being specific to the particular software at hand. It introduces some basics of device modeling and computer simulations, which leads up to the following chapter.

2.1 Semiconductor Physics

2.1.1 Semiconductor Materials

The basic properties of a semiconductor material is important for its ability to conduct electricity and provide as the basic building block for electrical devices. As its name suggest, a semiconductor is not fully conducting – it's *semi*-conducting. In comparison with the two other types of solid-state materials, *insulators* and *conductors*, semiconductors have an electrical conductivity in between, as shown in Figure 2.1. The electrical conductivity, σ, is defined as

$$\sigma = 1/\rho \text{ (S/cm)}, \tag{2.1}$$

where typical semiconductor materials have a conductivity between 10^{-8}–10^{2} S/cm [2].

Examples of a conductors where electrical charges flow freely are metals like copper (Cu) and aluminum (Al), while examples of insulators where charge flow is highly restricted are diamond (C) and glass (SiO_2). Common element semiconductor materials are group-IV materials like silicon (Si) and germanium (Ge), but also III-V compounds like gallium arsenide (GaAs) and II-VI compounds like cadmium telluride (CdTe). Ternary compounds like aluminum gallium arsenide (AlGaAs) and HgCdTe are also used in devices, where as the latter is the main material of interest in this thesis.

The conductivity in a material can vary with temperature, light or external magnetic fields, but it is most common to alter it by introducing *impurity* atoms into the material

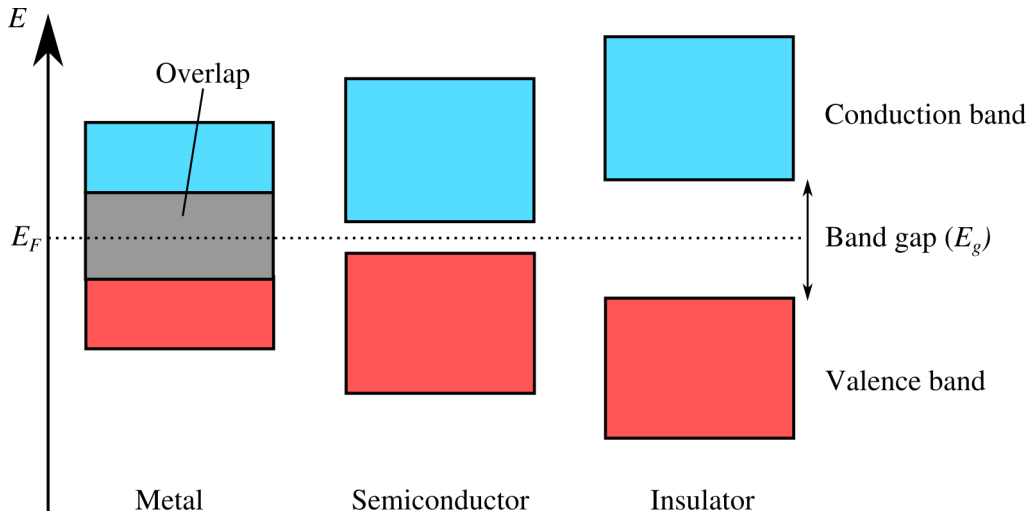
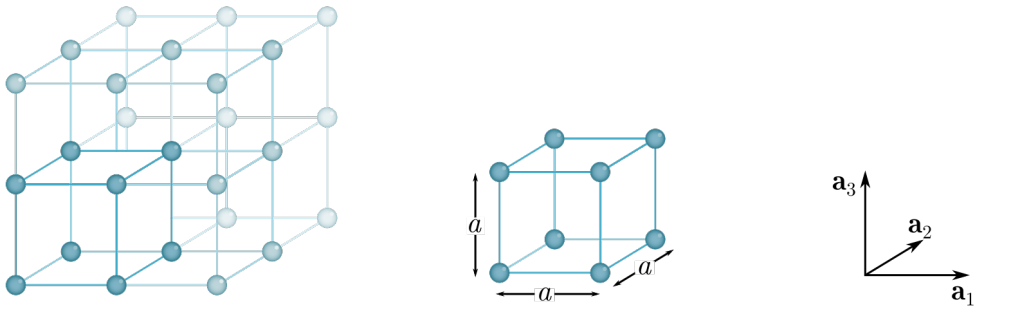


Figure 2.1: Comparison of energy band gaps between metals, semiconductors and insulators. Materials that conducts electrical current are classified as metals, while those who do not conduct are classified as insulators. Semiconductors are materials that falls between these two cases.
(Adapted from inductiveload under the CC-BY-SA 2.5 license)

structure – the process known as *doping*. To understand how the impurities affect the material properties, we need to introduce the concept of crystal structure in semiconductors.

2.1.2 Electrons in a crystal structure

With the exception of *amorphous* and *polycrystalline* solids, most semiconductors are single crystals, where the atoms are organized periodically in three dimensions. The fundamental building block of this three-dimensional periodic lattice is called a *unit cell*, as shown in Figure 2.2.



a) Simple cubic lattice b) Simple cubic unit cell c) Primitive vectors

Figure 2.2: A simple cubic lattice in three dimensions (a) can be considered a repetition of the unit cell (b). The unit cell is defined by three primitive vectors (c).

A generalized primitive unit cell consists of three primitive vectors and repeats throughout the lattice. From this unit cell we can construct the full lattice and define every point in the lattice by the translational vector

$$\mathbf{R} = n_1\mathbf{a_1} + n_2\mathbf{a_2} + n_3\mathbf{a_3}, \tag{2.2}$$

where n_i are integers while a_i are primitive vectors in different directions, as shown in Figure 2.2c for a simple cubic unit cell. The lattice points corresponds to the placements of the different atoms in the lattice. For specific values of the vector lengths and angles between them, we can reduce the numbers of possible lattices to 14 distinct *Bravais lattices* which are either considered simple, base-centered, body-centered or face-centered based on the location of the lattice points in the unit cell. In the case of a *primitive* cell, as shown in Figure 2.2b, the lattice points exist only on the corners of the unit cell. The information contained within the unit cell – distances, angles and atoms – contain all the information we need to describe the full lattice. As well as describing the physical structure of the crystal, the unit cell and lattice can also give information about the electrical properties of the crystal, by moving from the *real space* to the *reciprocal space*.

Reciprocal Lattices and k-space

Due to the periodicity of the lattice we can use *Fourier analysis* to convert the *real lattice* to a *reciprocal lattice*. This lattice is defined in the *reciprocal space*, also known as *k-space* or *momentum space*, where the unit is inverse length (1/m). In this formulation, the periodic lattice planes in real space corresponds to a periodic set of points which constitutes the reciprocal lattice. The periodicity of the lattice also leads to a periodicity of the physical and electrical properties, making Fourier analysis ideal for studying these properties [27].

From the vector $\mathbf{R}$ in Eq. 2.2 we can define the reciprocal counterpart as

$$\mathbf{G} = v_1\mathbf{b_1} + v_2\mathbf{b_2} + v_3\mathbf{b_3}, \tag{2.3}$$

where v_i are integers and $\mathbf{b_i}$ are the primitive axis vectors of the reciprocal lattice. These vectors are defined by the corresponding vectors $\mathbf{a_i}$ in the real lattice by an orthogonal relationship

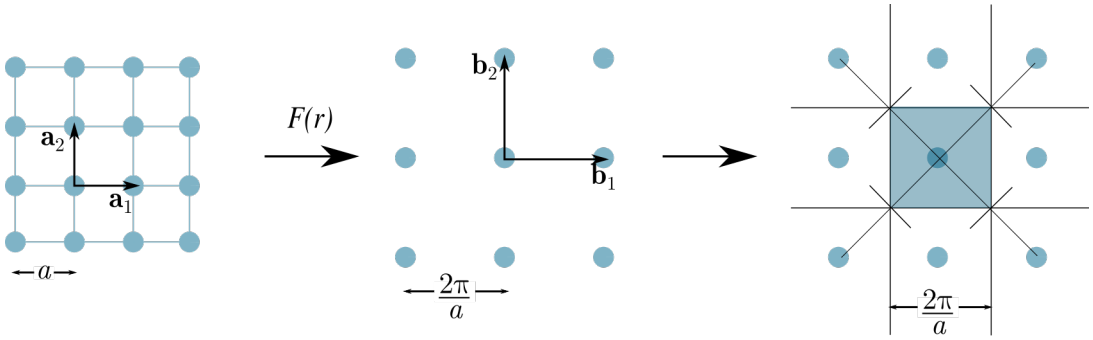
$$\mathbf{b_i} \cdot \mathbf{a_j} = 2\pi\delta_{ij}, \tag{2.4}$$

where δ_{ij} is the Kronecker delta. This relationship is visualized in two dimensions in Figure 2.3. By using $\mathbf{G}$ we can – similarly as with $\mathbf{R}$ for the real lattice – define every point in the reciprocal lattice. Since $\mathbf{G}$ defines all allowed points in the reciprocal lattice it also forces a condition for allowed wavevectors in the reciprocal space. More precisely, if a particle

with a corresponding wave in the reciprocal lattice with wavevector $\mathbf{k}$ is scattered in the crystal, it has to obey the condition

$$\Delta\mathbf{k} = \mathbf{k}' - \mathbf{k} = \mathbf{G} \tag{2.5}$$

where $\mathbf{k}'$ is the outgoing wavevector. $\Delta\mathbf{k}$ represents the change in wave vector at scattering, also called the *scattering amplitude*.



a) Simple cubic real lattice

b) Simple cubic reciprocal lattice

c) 1st Brillouin Zone

Figure 2.3: The real lattice (a) undergoes Fourier transformation to reciprocal space (b). The lattice plane spacing in the real lattice is a while the spacing between points in the reciprocal lattice is $\frac{a}{2\pi}$. $\mathbf{a}_i$ and $\mathbf{b}_i$ are the primitive vectors of the real and reciprocal lattice, respectively. The Brillouin zone (c) is constructed by intersecting lines at the midpoints between points in the reciprocal lattice. For the sake of visualization, the lattice is shown as two-dimensional.
(Adapted from Gang65 under the CC-BY-SA 3.0 license)

Similarly to a unit cell in the real lattice we reduce the reciprocal lattice to a periodic and uniquely defined primitive cell. This is commonly defined by a *Wigner-Seitz* cell rather than using the primitive axis vectors directly. By doing this the Brillouin zone limited by its *zone boundary* represents a minimal volume that contains all unique wavevectors that satisfy the condition in Eq. 2.5. Since it contains all unique wavevectors, the value of properties that depend on k will repeat periodically by crossing the zone boundary (*Bragg reflection*). As we will see this proves useful when studying the energy bands which determine the conductivity as described in 2.1.1.

2.1.3 Energy Bands and Carriers

A key characteristic of a semiconductor is the energy band gap between the valence and conduction band, as shown in Figure 2.1, in comparison with metals and insulators. For a single atom, the surrounding electrons occupy different discrete energy levels and for a large lattice these levels overlap into bands - allowed energy levels where electrons can reside. The valence band is defined as the fully occupied outermost band in a semiconductor

at absolute zero temperature, where as the conduction band is the first completely unoccupied band. Due to the thermal energy at temperatures higher than absolute zero electrons can transition from the valence band to the conduction band if the electron energy is higher than the bandgap. This creates a *hole* (electron deficit) in the valence band where the electron resided and allows for conduction of electrical current in the bands. This process is also referred to as the generation of an electron-hole pair. [27]

Due to the shape of the bands and the gap between them, they can explain the varying conductivity between different semiconductors, as we can see in Figure 2.4 which shows a generic band model in momentum space - an *energy-band diagram*. In reality, the band diagrams are 3D and more complex and will vary with different material but is shown in 1D for the sake of simplicity. The band gap is defined as the gap between the maximum of the valence band and minimum of the conduction band. Near this region, most energy bands are parabolic. [27]

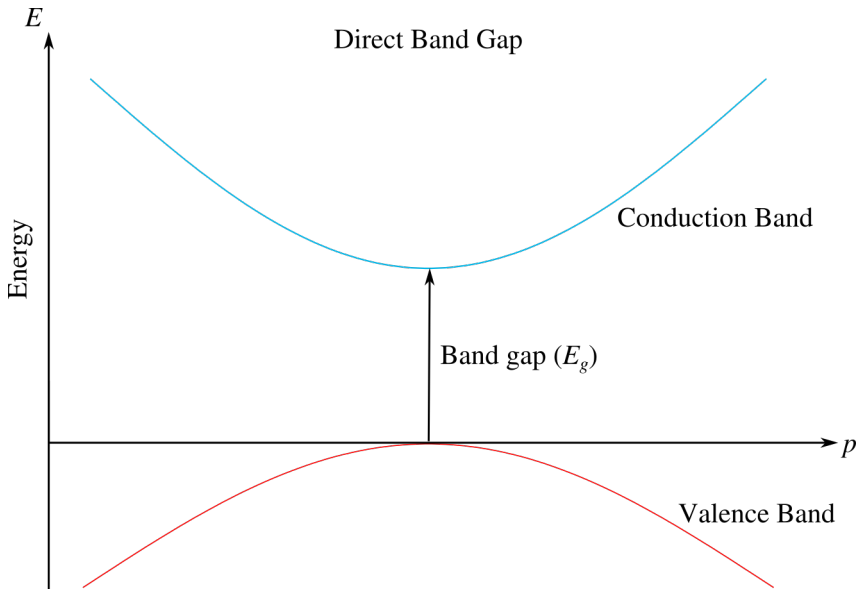


Figure 2.4: A generic energy-band diagram for a direct bandgap semiconductor. In the case of an indirect semiconductor, maximum and minimum values of the bands would not be aligned in p.

Carriers

The concentration of carriers (electrons and holes) will affect the electrical properties of the semiconductor. A semiconductor with the same concentration of electrons and holes is classified as an *intrinsic* semiconductor. For many applications it is desirable to alter this balance by introducing impurities. These impurities are either *donors* or *acceptors* – atoms with a surplus or deficit of a valence electron compared to those in the crystal. As these valence electrons are not covalently bound to the crystal atoms, they can more easily move through the material as mobile carriers. This leads to an *extrinsic* semiconductor, which is

classified as either *n*-type (larger electron than hole concentration) or *p*-type (larger hole than electron concentration) based on the impurity introduced into the crystal. In these cases the carrier with highest concentration is referred to as the *majority* carrier and consequently *minority* carrier for the opposite. This is shown in Figure 2.5 for both acceptors and donors for a generic semiconductor.

The most important effect this has with regards to the energy-band relationship is that it will introduce new energy levels in the band gap near the conduction (*n*-type) or valence (*p*-type) band. This allows for creating semiconductor devices consisting of interfaces between *n* and *p*-type materials.

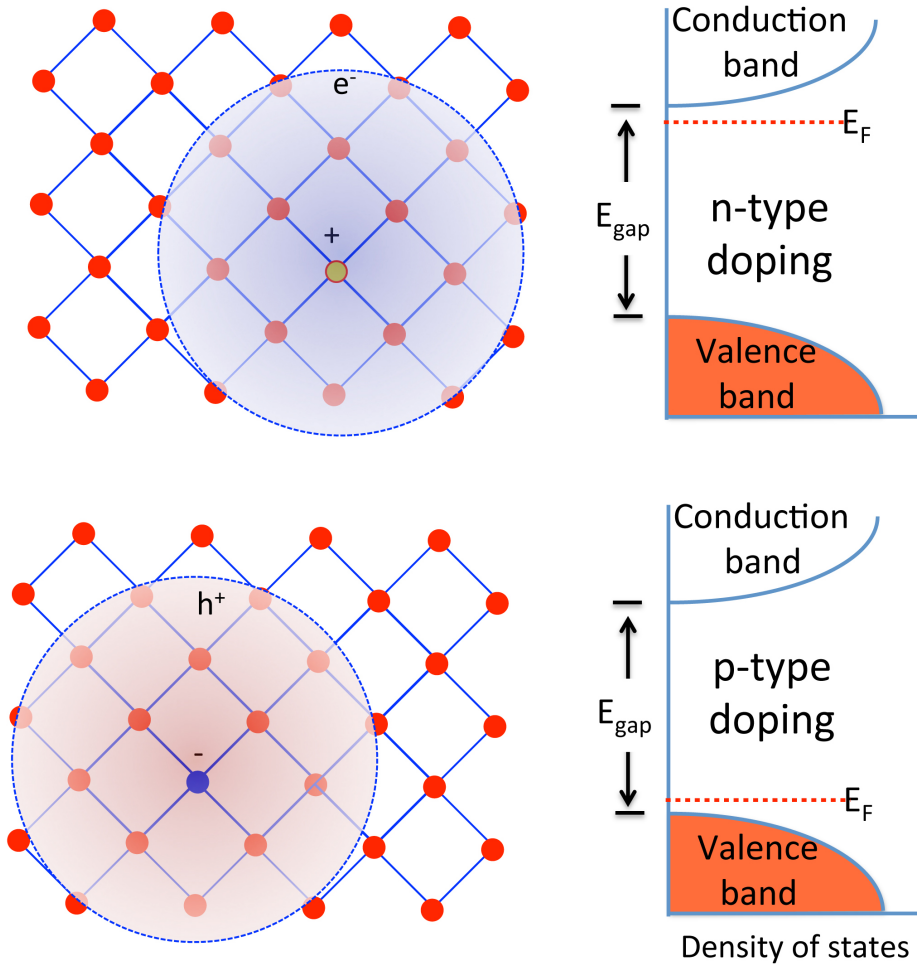


Figure 2.5: Illustration of *n* (top) and *p*-type (bottom) doping in semiconductors where an electron donor or acceptor is introduced in the material. The right side shows how this affect the energy-density of states diagram where the Fermi-level is closer to the conduction or valence band, respectively. In an intrinsic semiconductor this level is in the middle of the band gap. (Illustration from Tem5psu on Wikipedia under the CC-BY-SA 4.0 license)

2.2 Semiconductor Devices

To use semiconductor materials for useful applications we need to engineer a device. To achieve this it is necessary to have materials with different electrical properties together. The basic building blocks of semiconductor devices is the $p-n$-junction, together with metal-semiconductor junction and the junction between different semiconductor materials (*heterojunctions*) [27].

A special case of a $p-n$-junction is the photodiode, which can create electrical current from optical signals. It is operated under reversed bias, where the p-type end is connected to a negative terminal and oppositely for the n-type. This creates a large voltage barrier and a large electrical field around the junction, allowing separation of the current bias-generated current and the read-out photogenerated current.

2.2.1 Avalanche Photodiode

In the 50s and 60s much research were put into developing semiconductor devices for photo-detection in the wide infrared spectrum ($\sim$1–12 µm). The development in these and following years was largely driven by the defense and space industry [18].

An APD is type of photodiode which has a built-in gain to increase the read-out photo-generated current. APDs generates carriers by converting incident photons by the photoelectric effect in a cascading manner, such that a single photon can generate multiple carriers. The strong reverse-bias creates a strong electric field at the $p-n$-junction, where carriers are accelerated such that they again can generate new carriers. This process is known as *impact ionization* where bound carriers break loose. As sufficient energies, the newly generated carriers can repeat the process of impact ionization and the process is such known as *avalanche multiplication* and leads to a multiplication gain of the internal current. [2]

Due to the nature of the avalanche multiplication process which excites carriers in a random manner, minimizing the generated noise from primary and secondary excitations is an important consideration. This is often mitigated by having regions with different levels of doping to separate the photo-generated carriers and avalanche-generated carriers. Another important consideration with APDs is the *quantum efficiency*, which describes how efficiently the device convert the number of incident photons to generate primary charge carriers. [2]

Various materials are used for APDs depending on the application and requirements to wavelength, noise and operating conditions. As previously mentioned, the material in this thesis is HgCdTe and a example APD device based on this material will be presented in the following chapter.

HgCdTe

$Hg_{1-x}Cd_xTe$ is a ternary alloy semiconductor commonly used in detector devices due to its tunable bandgap and high optical absorption. It has a zinc-blende crystal structure, which consists of two interpenetrating face-centered cubic lattices. When $x = 0$ it is the semimetal HgTe with zero bandgap while at $x = 1$ we have the semiconductor HgTe with a bandgap of 1.5 eV. The bandgap is direct and also changes with temperature.

It is at low temperatures, down to 77-120 K the material exhibits the highest quantum efficiency and lowest dark current [28]. Due to the band structure, there is an asymmetry between the effective masses of the electrons in the conduction band and heavy holes, which gives a highly favorable electron to hole impact ionization ratio for compositions with $0.1 < x < 0.7$. This makes the multiplication process mainly initiated by one carrier, which yields a lower gain noise than if both carriers had the same impact ionization ratio [29].

2.3 Semiconductor Device Simulation

The era of computers makes it possible to solve large analytical problems by using numerical models and computers. This can contribute to experimental work, both to verify empirical results but also to more precisely describe properties that proves hard to quantify experimentally. As shown in Figure 2.6 the interplay between experimental work and simulations is of importance to achieve physical results. To achieve this, we need a suitable model and well-determined realistic input parameters. According to Krc and Topic [30] a general approach to this can be listed as

1. Defining the equations that describe the problem (device properties)
2. Choose a numerical approach to solve the problem (system of equations)
3. Implement the model in a computer code (simulator).

With regards to the list above and the simulator software used in this thesis, an overview of the model and implementation will be given in the following chapter. The following sections in the present chapter serves as a introduction to the theory behind the most important aspects of the implementations.

2.3.1 Monte Carlo Method

Most physical systems are in principle deterministic, assuming that they are well enough understood and explained. MC is a simulation method which is based on using stochastic variables to solve a deterministic problem. The method has been around since early 1900s,

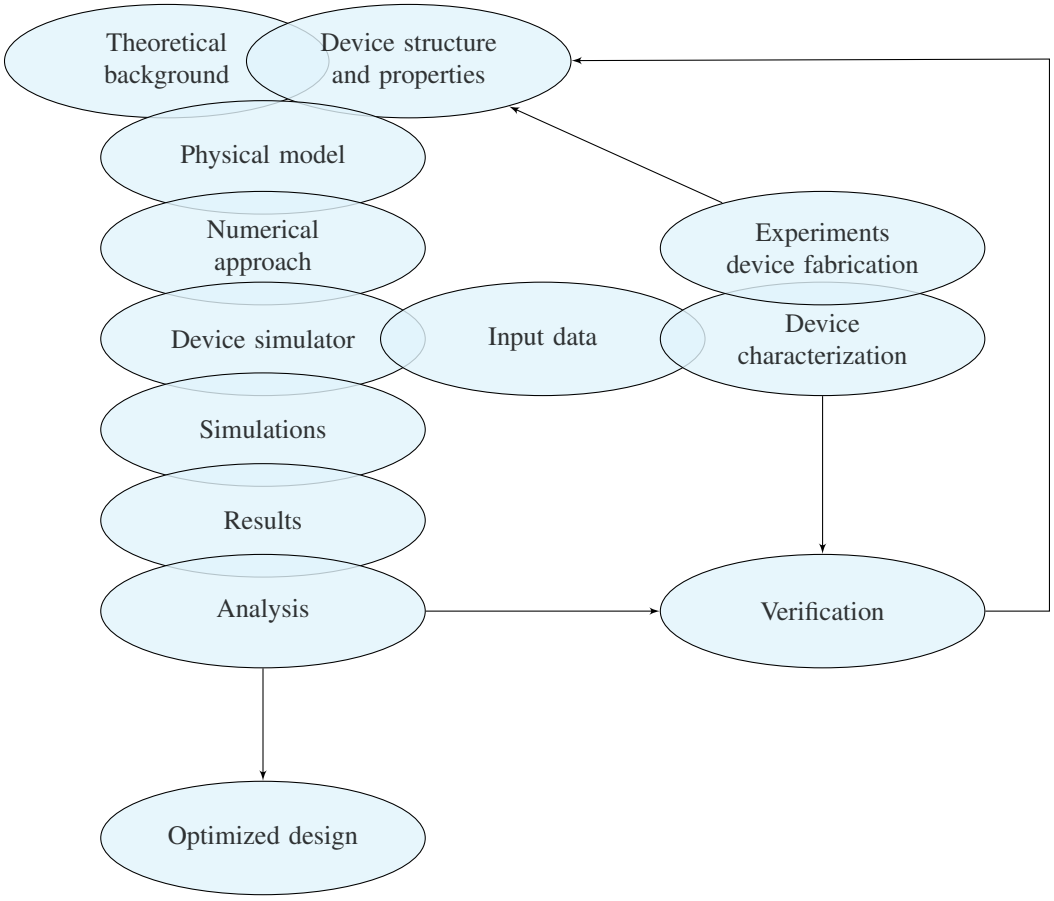


Figure 2.6: A flow chart which shows the main steps in simulation and its interplay with the experimental work. Adapted from Figure 1.12 in [30].

but won territory in the 40s and 50s as one of the most used simulation methods across a variety of fields where it is difficult to approach a problem analytically [31].

The method is based on proposing random changes to a system, and evaluating whether the change was a favorable one. This can be generalized into three main steps, which are repeated a large number of times. First, the range of possible values (*domain*) for the stochastic variable is defined. A random number is then found within that domain, based on a probability distribution function. Finally, the value is evaluated based on a set of predetermined rules. A generic example is a simulation of tossing a dice where the discrete domain will be $x \in \{1, 2, 3, 4, 5, 6\}$ with a uniform probability distribution function for the stochastic variable X. [32]

A strength of the MC method is the generic nature of the model. As such, MC can be used to simulate a range of quantitative problems in various fields. It can be applied in physical sciences, engineering, statistics, finance, compute and more [31]. A description of the use of the method in the simulator in this thesis is given in the following chapter.

2.3.2 Finite Element Method

Many space and time dependent physical processes are expressed through partial differential equations (PDEs) - functions which are dependent on multiple variables and coupled together through derivatives of one or more of these variables. However, analytical methods can only give solutions to very few PDEs. This drives us to solve the equations numerically, giving an approximation of the exact solution.

FEM is one way of solving a PDEs numerically. It approximates the PDE by discretization into equations that are more easily solved within a subdomain of the system - a *finite element* - which represents the spatial discretization. To obtain a suitable form of the equations methods like Ritz Variational or Galerkin's method can be applied [33]. The equations for the elements are combined to a global set which can yield an approximation to the solution of the PDE [34]. According to Jin [33] the basic steps of FEM can be summarized as

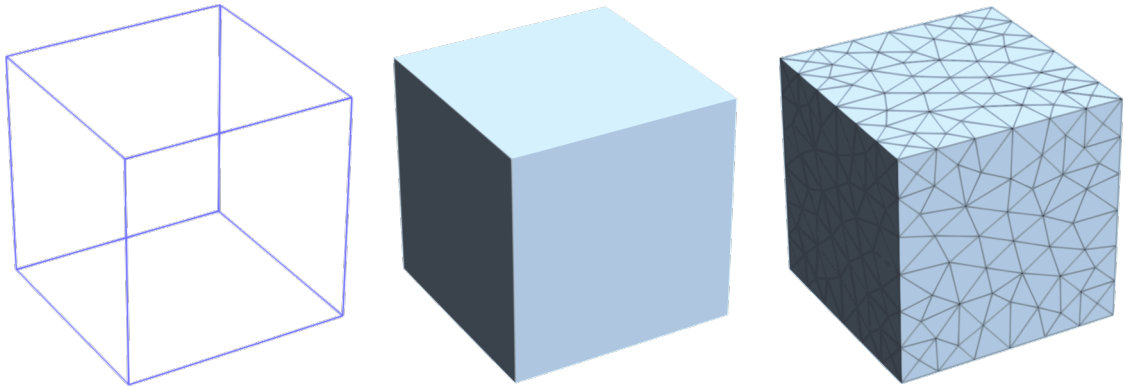
1. Discretization or subdivision of the domain
2. Selection of the interpolation functions
3. Formulation of the system of equations
4. Solution of the system of equations

The system of equations will quickly be large of the discretization is dense and it will require special methods for solving them efficiently [30]. A way of doing this which is implemented in the simulator software is the preconditioned conjugate gradient method [26, Ch.3].

FEM was initially used for structural mechanics but is today used among many fields of engineering, including electromagnetism and fluid mechanics [33]. In this thesis it is used for modeling the 3D APD to solve the electrical field updates through Poisson's equation as described by Fatnes in [26, Ch. 2].

Meshing

When using the FEM, an important step in the process is how the discretization of the domain into elements is carried out. This is done by creating a mesh of the domain, as shown in Figure 2.7 for a simple cube.
In general, the finer the mesh (more nodes and elements), the better an approximation one is able to obtain at the cost of higher computational cost. To keep computional cost low it is common practice to use a finer mesh in regions with high variation and larger elements in more steady regions [33]. Furthermore, the shape of the elements can be of large importance, and computational methods exist in which the element sizes and shapes



a) Points and lines b) Volume c) Meshed volume

Figure 2.7: In meshing software gmsh a cube is defined through the points at the 8 corners. The points are connected by lines (a) and shown as a cubic volume (b). Finally, an optimized mesh of the volume consisting of 358 nodes and 1809 tetrahedral elements is created.

are adapted in regions of the domain where large error estimates are found. Elements can also be triangles or tetrahedrons (2D/3D) instead of squares/cubic elements which allows for more complex geometries and fewer nodes.

As further described in the following chapter, we use computer software with graphical interfaces to create meshes for use with the simulator software. The meshing software used in this thesis is *gmsh*. It uses its own 3-D *Delauney* algorithm to create and optimize meshes and its implementations are described in [35].

To create a mesh with the software we need to define the overall geometry including points, curves, surfaces and volumes. For an APD, this involves generating an input file with the overall width, size and length as well as the doped regions of the device - the procedure is described in Appendix A in [26]. As the meshing is carried out automatically, the meshing algorithms will not be discussed in-depth. Some important statistics from the mesh that directly affects the simulation will be discussed, including np (number of points/nodes) and ne (number of elements).

2.3.3 Numerics and Limitations

As numerical approximations are never exact, they will always differ from the analytical solutions. When implementing numerical algorithms in a computer program, additional sources of errors can also be present. This refers both to how computers represents numbers, rounding errors, how local errors might propagate throughout the simulation and how the numerical method itself might introduce discrepancies (e.g. convergence issues).

The precision of floating-point numbers is a well-known potential problem for computer programs. Calculations including long floating-point numbers can lead to rounding errors due to the limited number of bits in a defined variable [36]. To mitigate this most

floating-points are defined with a larger bit length – in the simulator software the working precision is explicitly set to up to 15 digits (double precision) with an exponent range of up to 307 digits ($10^{\pm 307}$). At the cost of computational time and memory requirements, this yields fewer rounding errors than for built-in parameter types. It is desirable to keep numbers within a reasonable magnitude and as such parts of the simulator also involves scaling to keep numbers consistent. Some optimization is also performed at compilation level [37].

The implementation and complexity in the FEM and MC models will highly affect computing time and resource requirements. With regards to the Monte Carlo implementation the number of particles simulated will highly affect the time. With regards to FEM the discretization and type of elements (e.g. linear vs. quadratic) is of importance, especially in 3D. As the size of the elements decrease, the number of nodes and effectively the system of equations will increase. This demands more computational power and memory, but this can be somewhat mitigated by introducing decomposition methods of limiting the use of full-sized sparse (mostly zero-elements) matrices [33]. This is implemented in the simulator, as described in [26, Ch. 3.1].

As computational and memory resources are limited, this also limits the efficiency of the simulator. If we have access to a high-performing computing cluster, there is a possibility of running a program with shared memory (running jobs in parallel on a single node), distributed memory (runnings jobs in parallel on multiple nodes) or a combination. However, this requires significant changes to implementations, e.g. splitting up coupled equations, and is not easily done. Commonly used standards for this includes OpenMPI (distributed memory) and OpenMP (shared memory). [38]

Chapter 3

Simulator Model and Implementation

This chapter gives a overview the simulator program and the work carried out to further test and develop the program. Section 3.1 introduces the simulation model while 3.2 describes the program flow of the simulation.

As the overall structure of the program has not changed from the prior version in 2018, the reader is recommended to review *Appendix B. Overview of Source Code* written by S. Fatnes in in [26]. For further reference on mathematical and physical implementations, the reader is referred to their theses of previous students [20–25].

3.1 Simulation Model

In general terms, the simulator it is a particle-based bias simulator for a semiconductor device. It deals with charge transport of particles and attempts to simulate a model of a real-life semiconductor device similar to a realistic bias simulation. Since it is not feasible to track particle by particle in a real-life device, the major challenge with modeling the simulator is the behavior of these particles.

The simulator software is in its current version called *Monte Carlo software with finite element Poisson solver* (MCFEM). The full model in the simulation spans a range of physical and numerical problems, where the implemented solver utilizes the MC method and FEM as the most important components. The FEM is used for the discretization of the device to locally solve Poisson's equation and calculate the electric field during simulation. The MC method deals with the movement of particles in the device, utilizing randomness to achieve realistic movement for a large number of particles over time. For the random number generator the computer time upon simulation is used as a seed.

To use the simulator we define a set of input parameters and a device model to simulate and let it run for the desired time. From this we generate data about the potential at all

different locations in the device and track the movement and behavior of particles in the device. Based on our input we are able to see how proposed geometries, material compositions and parameters affect the physical behavior of a device. When the simulator reaches a sufficient level where the implementation of models works according to observable physical laws and outputs realistic data, the data from the simulations can thus be useful to support the research of devices prior to prototyping and making costly investments in lab. The simulator program is implemented in Fortran, which is described in 3.3.

The simulator is designed to be general in terms of choice of materials and device, such that it can be reprogrammed to simulate another device geometry or another semiconductor material.

3.1.1 Device Parameters

The device used for simulation is an HgCdTe APD similar to that which is researched by FFI both through simulations and physical experiments [13–16]. A simple model is shown in Fig 3.1 with geometries, doping types and doping densities. These are common among all simulations.

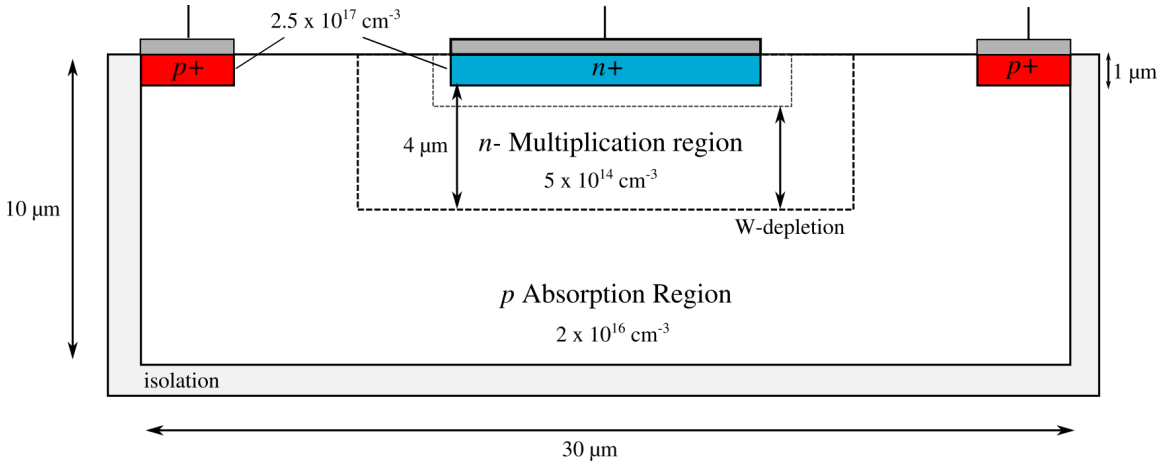


Figure 3.1: The APD model which is simulated with geometrical parameters given. The model is shown in 2D but is 10 µm in the third dimension as well. The doping type and doping densities are given for the different regions. The isolation layer is not part of the simulated domain and only included to satisfy boundary conditions if an alloy gradient is applied. The top contacts are connected to a read-out circuit which is not shown.

The 1 µm thick isolation layer is included in the model due to the possibility of adding a alloy gradient of the material in the absorption region without without violating the boundary conditions. This is not done in current simulations, and it does not changes the nature of the simulation, as shown by Fatnes [26].

The simulation is carried out at room temperature and the reverse bias is set to -7 V applied to the *p* contacts, while keeping the *n* contact at 0 V.

The material used is $Hg_{0.72}Cd_{0.28}Te$. By using Vegard's law the lattice parameter is $x \cdot a_{CdTe} + (1-x) \cdot a_{Hg}$ which corresponds to $\sim 6.486 \times 10^{-10}$ m when using lattice constants 6.477×10^{-10} m for CdTe and 6.490×10^{-10} m for HgTe. This composition corresponds to a band gap of approximately 0.28 eV at room temperature [39]. All other material specific parameters used in the simulation is given in `matpar` module.

The band model used for the material was created mainly by Olsen [20] Skåring [22] in their thesis work with the model. It makes an approximation of the Brillouin zone as spherical with a radius of $\sim 9 \times 10^7$ m^{-1}. The model consists of a isotropic nonparabolic conduction band and a more complex heavy hole and light hole band. In the latter bands the band model consist of an analytic part for low k-values while it uses another model for the high k-values, with a ad-hoc solution for creating a continuous second derivative. This is well described in Ch. 3 of Skåring's thesis [22].

3.2 Program Structure

The structure of the simulator is based around a main program file, `MCFEM`, which governs the overall run of the simulation. It calls the various modules located in Fortran module files. The modules are collections of variables, subroutines and functions which are connected through a physical aspect in the implementation, for example FEM routines collected in a module while carrier dynamics is collected in another. Several modules consists also of components that are common in several subroutines, such that they are contained in a separate module file. An automatic documentation system called FORD (Fortran Documenter) is used to generate an interactive HTML-based documentation which contains all modules, subroutines, functions and variables with comments from developers.

An overview of the simulation is shown in the flowchart in Figure 3.2. This shows the logical flow of the program - from initialization through the time loop and to the completion of the simulation.

The input for the simulation is one initialization file consisting of all device and material parameters, and one generated mesh file for the device geometry. The mesh file is created with *gmsh* as described in 2.3.2 in the previous chapter. The initialization file also contains some options for the simulation, like which mechanisms should be simulated, time step resolution and how long the simulation will run, at what times data should be collected and the number of particles that should be simulated. The most important device and material parameters that are given is the doping densities and geometry (mush match the generated mesh file).

The first action is the initialization of variables and types used in the simulation. This allocates the necessary memory for the simulation to run successfully. The device triangulation from the mesh file is used to assemble the stiffness matrix for the FEM. The device is also initialized within the program based on the input file parameters and mesh file.

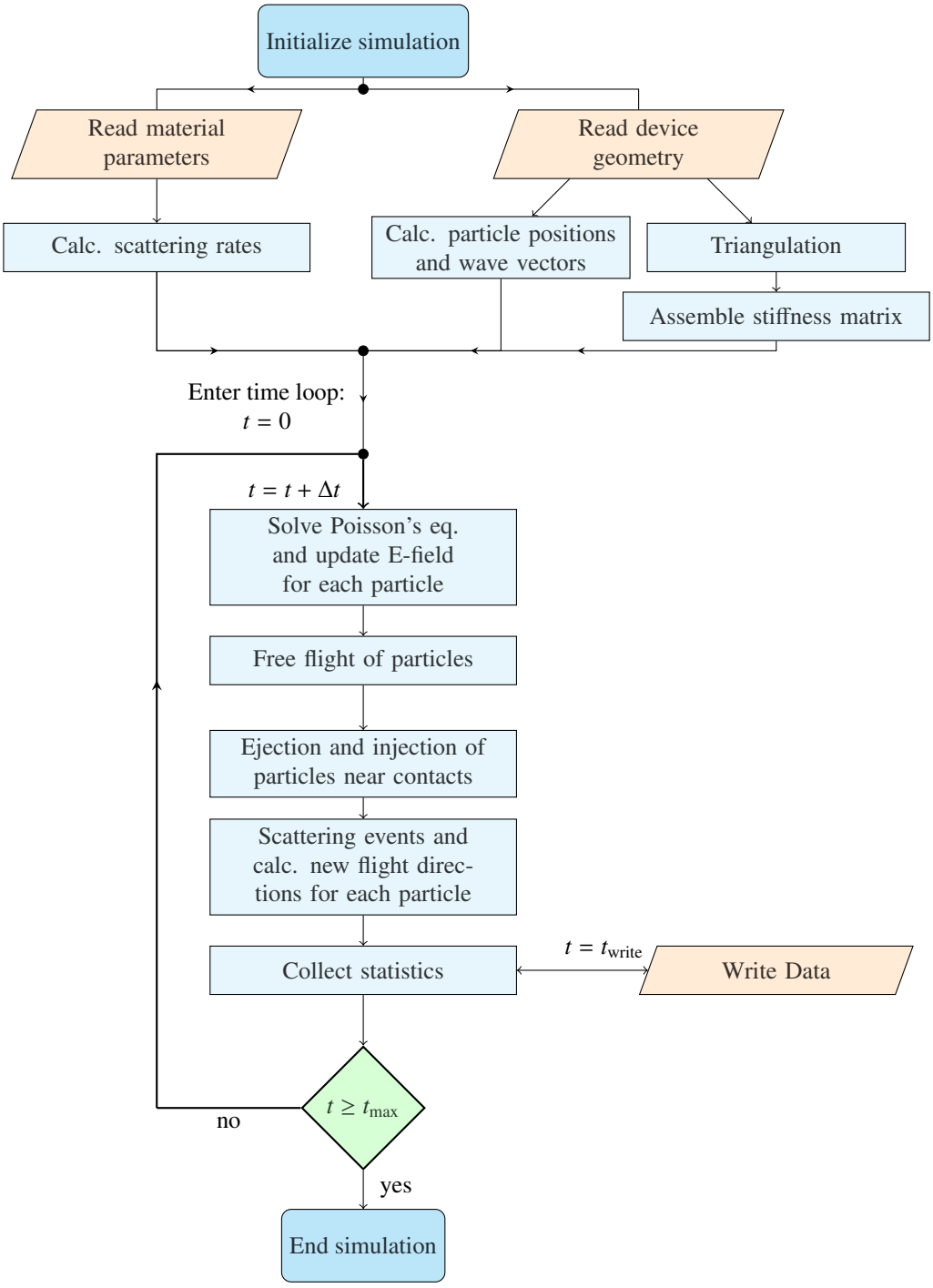


Figure 3.2: The overall logical flow of the MCFEM simulator program is shown. All the steps prior to the time loop can be considered as initialization steps for the simulation to start while the time loop is repeated for a defined time range. The output data is collected at predefined times during the time loop until the time t reaches the defined simulation length and the simulation ends.

The steps in the time loop are the main parts of the MC simulation. The time step between each in the simulation is typically 1–2 ps. Here the Poisson equation is solved and the electric field is calculated for each of the particles before the free flight of particles is carried out. According to boundary conditions near the contacts carriers are ejected or injected and carriers that reach the edges of the device are reflected. Scattering mechanisms are then simulated for each particle and new flight is calculated. Positions and k-vectors for the particles are updated several times during these steps to their respective variables. If the timestep is in the set of defined extraction timesteps statistics from the simulation are outputted to data files. If the the time has not yet reached the defined simulation time the loop is entered again using the particle state from the last loop iteration.

The simulator produces several output data files, which is defined in the module for collecting statistics, and some output files that are only generated at initialization and used throughout the simulation. The most important data files that are written to during the time loop (only at defined extraction timesteps) is the electron and hole wave k-vectors and positions. The potential at each node in the geometry is also written during these timesteps.

At the end of the simulation all used files are closed and variables are deallocated from the memory.

3.3 Coding, Build and Runtime Optimization

The simulator is implemented in Fortran. The choice to use Fortran as a language is justified by choosing a language that is numerically fast and simple to use for typical numerical problems (its name is derived from *FORmula TRANslation*). It is widely used in the scientific community and has excellent support for legacy code (e.g. available optimized numerical algorithms) as well as modern standards which are regularly introduced in newer version of the language [40]. Its interfaces to other widely used programming languages like C/C++ and Python is also an advantage, which is utilized in the WIEN2k program, which FFI uses in conjunction with their device simulations. All changes to the code are tracked through a git repository.

In this work, the program has been built using the open source compiler *gfortran* 7.3.0 using optimization level 3 [37]. While debugging and testing several helper flags have been utilized to get more verbose runtime information. All the full commands necessary to build, debug and run the simulator is documented in the source code README files which is available upon request to the author. With changes to the build files an alternative compiler like Intel's commercial compiler *ifort* can also be used.

The simulations were run on nodes in the *Idun* cluster at the courtesy of the HPC group at The Norwegian University of Science and Technology (NTNU) [1]. The simulator was also compiled on these nodes with the similar version of *gfortran* as the code is optimized

[1] `https://www.hpc.ntnu.no/display/hpc/NTNU+HPC+GROUP`

to the platform they are compiled on. The nodes are mostly Dell PExxx series with 2 x Intel Xeon E5-26xx processors with 20-36 cores and 128-192 GB RAM. The avg. workload for simulations required ~ 40 GBs of RAM and run an one core, allowing several simulations to be run simultaneously at one node or distributed through several nodes. Actual wall-time for simulations are described in the following chapter.

Chapter 4

Simulator Development and Results

This chapter gives an overview of the issues that were identified to investigate in this thesis and what was done to mitigate them. It also includes some results from simulation runs.

A total of 12 full simulations of the APD device was run with simulation times ranging from 11–40 picoseconds. Besides fixing smaller errors in the program, the simulation runs have mainly been focused on investigating the different problems and comparing with previous runs from Fatnes. As described in the following sections, the most important parameters that were varied are related to device geometry and length of the simulation. All the simulations were done with the optimized mesh as described in 4.3.

Despite numerous simulations, the data from these are not as well represented and visualized in this chapter as they deserve. Yet, all the data is exported and stored for analysis and comparison for future developers of the simulator.

4.1 Program Development

There are several issues with varying degrees of complexity and severity in the current version of the simulator. Due to the modular development of the code by multiple authors, some of these have occurred when implementing new features while some have been well-known and considered acceptable. The ones that are mentioned in this section are specifically following up the issues that arose from new implementations and changes by the previous author [26]. The main motivation is to increase stability of the simulator and investigate essential physical issues with the model.

4.1.1 Compilation Changes

In the beginning of this project, a lot of effort was put into being able to compile the program with *ifort*. Since *ifort* has been the preferred compiler for the developers of the

non-student simulator, as well as for several other previous student developers, it was desirable to change the compiler. As the simulations also would run on Intel processors, using their compiler should yield the most optimized performance.

As all implementations in the previous version was carried out while compiling with *gfortran*, the task of being able to compile with *ifort* and run successfully proved to be non-trivial. After translating all the build files to be compliant with *ifort* with similar compilation options as in *gfortran*, simulation runs were prone to crash after running 5-10 minutes, reaching only a simulation of only 12 fs (of 36 ps) with a segmentation fault.

Despite being compiled with the `-heap-arrays` to put the arrays and temporary arrays on the heap instead of the stack, the error was persistent. The error occurred in the the flight routine the holes, when the error with unusually large k-values (and corresponding zero velocity-holes) was persistent. As the simulation was not able to run more than a few picoseconds, it was difficult to debug the error over a larger number of timesteps. The choice was therefore made to revert the compiling approach to using *gfortran*.

The changes in the build files was reverted to again be compliant with *gfortran* and to run simulations in the same manner as the previous developer. Since the changes are tracked in a git repository, it is possible to recover these changes to continue the effort to change to *ifort*. The latest version that was tested was *ifort* 2018.3.222.

4.2 Unphysical Behavior of Holes

4.2.1 Background

An issue with an apparently unphysical behavior of particles were one of the first issues investigated. Based on the potential plots of the devices, as presented briefly by Fatnes in her thesis, and the trajectories of particles over time it was clear that the movement of holes was inconsistent with the simulation model. Holes were attracted to surfaces and displayed severe noise in the potential plots.

At first it was difficult to understand which part of the simulation could cause this behavior as the high-level flight routine responsible for movement of particles calls a large number of subroutines. It was also considered that the the physical model of the APD device and material could come to play in causing irregular behavior in parts of the device.

Neutral Region Height

Based on previous experience with a early version of the MCFEM-software from a previous master student [23], stability issues occurred when the height of the neutral region was relatively large ($\sim$1 µm). With a large neutral region, the injection algorithm was prone to inject too many particles compared to the number of particles that were ejected – effectively causing an oscillating number of particles in the device. This was later remediated by

researchers at FFI by reducing the neutral region height and stable behavior was observed when the height was reduced to 0.03 um.

Based on this, an initial hypothesis was that the previously used heights of 0.1–0.2 µm was too large and that reducing this could cause a more stable behavior.

Holes With Zero Velocity

As a continuation of looking into the hole trajectories, the data files were further analyzed. By visualizing the positions over time and following the behavior of particles it was clear that certain holes suddenly stopped moving in the simulation. This occurred even after few picoseconds and did persist throughout the simulation. By generating logs in the simulation runs, this was further confirmed with a definite zero velocity. Since the charge of static particles would severely affect the local potentials in the device, this was quickly identified as the most important issue with the model.

4.2.2 Results

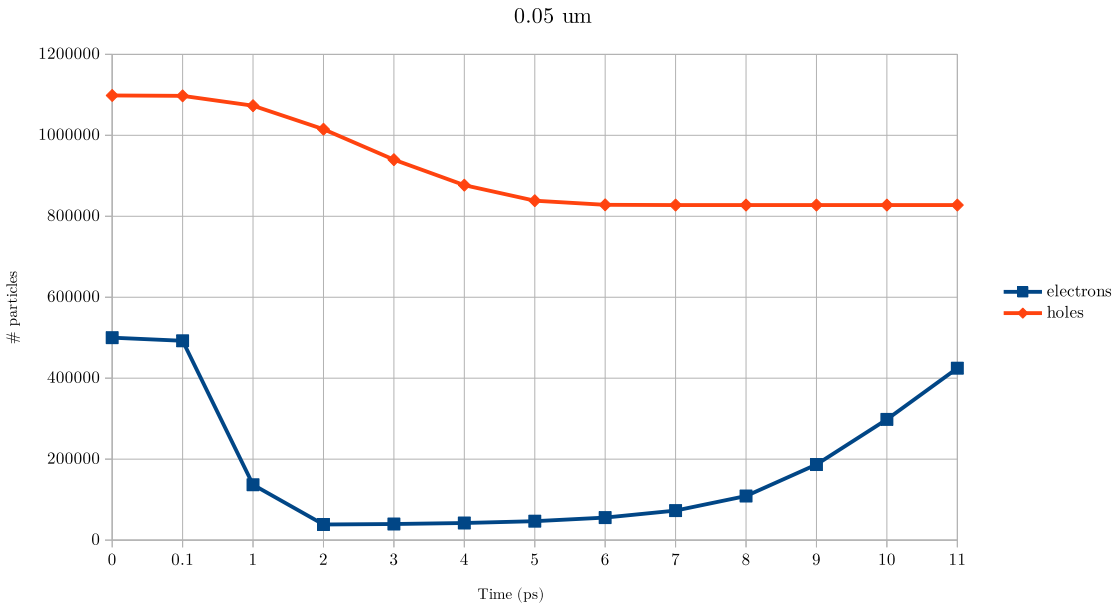


Figure 4.1: A plot total number of particles in the device through a simulation of 11 picoseconds. As seen, the response in both electron and hole count is visible, but after approximately 5 ps the hole count is not changing. The device has a neutral region height of 5 µm.

The results from the analysis of counting the total particles in the device, can be seen in Figure 4.1. The plot shows the number total number of holes and electrons in the device throughout a simulation of 11 picoseconds. At first the response of the device is clearly

visible as the number of particles changes. After 5 picoseconds the number of holes approaches a level where it does not further change. I n this case, the majority of the holes does move within the device leaving few holes to reach the contacts to be ejected. Thus, no new particles are injected to the device as well. As the simulation continues we can see that the number of electrons still change. Presumably, the opposite effect occurs for electrons where they reach the contacts and are ejected. Over time, the lack of holes causes a larger number electrons to be injected.

A similar response also was observed for simulation runs with varying neutral region heights near the contacts. Simulations were carried out with 0.02, 0.03, 0.04, 0.05, 0.07, 0.1 and 0.13 μm. As the trajectories of the holes did not change significantly with these variations, the hypothesis of reducing the neutral height to get more stable behavior was rejected. In further simulations the neutral region height was set to 0.1 μm.

In addition, for simulations with a neutral region height of 0.02-0.03 μm, the simulations was not able to run successfully. In these simulations the hole simulation seemed to freeze when reaching 4-5 picoseconds with no apparent error thrown by the simulator. It is likely that this lower limit of neutral region height correlates to the mesh size and should be possible to avoid if the mesh size is made even smaller near the contacts. This hypothesis needs further review to verify.

Large k-values

After a number of simulation runs with varying neutral height without any resolution of the unphysical behavior, the implementation of the flight routines was further investigated.

By analyzing all all the routines called in indirectly form the `MCFEM` main program, it was possible to identify which parts of the simulator model that could be the root cause of persistent zero velocity of holes. The movement of holes is calculated in a step fashion where the velocities at the last and current timestep is averaged and multiplied by the timestep length. This model is fundamentally different from the calculation of movement of electrons which utilizes a model based on the strength of the electric field as described in 3.4.3 in [41].

To investigate the calls made the calculate the movement of holes we identify the calls as:

1. `MCFEM` (main program) calls `flight` function from the `carrierDyns` module
2. The `flight` function calls the `holevgi` from the `measurements` module
3. This `holevgi` function calls the `dEdk` from the `scatteringrates` module.

The `holevgi` function calculates the velocity for holes by using the group velocity multiplied by the size of the wavevector in each direction x, y and z. The group velocity is calculated by the formula

$$v_g = \frac{1}{\hbar}\frac{\mathrm{d}E}{\mathrm{d}k}. \tag{4.1}$$

In this function the only changing variable is the derivative of the energy-band with respect to k, which is calculated according to the band model implemented by Skåring [22]. This model contains the analytic part and the so-called "ad-hoc" part which calculates the derivatives differently based on the magnitude of k. When k reaches a value of the predefined variable that corresponds to the boundary of the first Brillouin zone, the derivative is not defined. According to the symmetry in k-space and the band model which is only defined in the first Brillouin zone, the hole should reflect at the boundary of this zone and have a valid value.

The reflection of holes is in fact only done when the position vectors surpasses the physical boundaries of the device, but surprisingly the holes are not reflected in k-space when the wave vectors surpasses that of the first Brillouin zone. This leads to a magnitude of the k-value which continues to increase. As such, there is no defined derivative of the band for these values. This causes the output of the `dEdk` function to be zero and thus zero movement of holes.

Proposed Solution

Since this is closely related to Skåring's band model implementation, it proves difficult to solve it elegantly as the derivative is calculated and used in many other functions throughout the model. To change or even replace this model requires a significant development effort out of the scope of this thesis.

It is worth mentioning that there is an comment in the `dEdk` function, in the case of k-values out of boundaries that mention that this is not dealt with. This shows that at some time, this has been an issue but this has not been addressed in the most recent theses using the simulator. This could either be due to short simulation times where it has not yet been clearly observed or it has not occurred to such an extent prior to the latest implementations. Regardless of this, it is obvious that this is a major weak point in the implemented band model and thus fundamental to solve for the further development of the simulator.

Besides redefining the majority of the band model, including the calculation of the derivatives, a proposed solution is to add a Bragg reflection condition for k-values exceeding the first Brillouin zone.

This should done prior to calling the `dEdk` function. This could act as an "ad-hoc" solution which only affects the calculation of hole velocities without altering the `dEdk`

function which is used by other routines in the simulator. This was attempted at the very end of the work in this thesis, but not without throwing fatal errors during compilation.

4.3 Mesh Optimizations

4.3.1 Background

In the work by Fatnes, several improved meshes for the simulator were created. A representative selection of these meshes are shown in Figure 4.2. The meshes displayed a variation in the mesh density, either overall in the device or selectively in the region around the *pn*-junction.

Based on the simulation cases presented it her thesis further optimization was possible. It was shown that an overall high density mesh produced too much noise in the simulation. Especially in the absorption region with lower doping densities, a small element size produced noise.

The comparison of quadratic and linear elements also showed that the use of quadratic elements did not yield much improvement in the simulations based on the accuracy of the potential in the device. Combined with the cost of increased computational requirements in the simulator for working with quadratic elements, the use of linear elements seemed most promising.

4.3.2 Results

Based on the meshes shown in Figure 4.2 it was sought to generate a new and improve mesh with linear elements and a denser mesh in regions of interest without causing noise in the simulation. Such a mesh would improve the time and resources needed for simulation by reducing the number of nodes while keeping a sufficiently high level of detail.

The mesh created in this work is shown in Figure 4.3.The mesh was refined to increase the accuracy in regions where the potential changes quickly, i.e. where the field gradient is large. Thus, the regions of interest are the regions around the junctions of the doped regions, including the doped regions under the contacts. With the previous geometry files, such a level of mesh selectively around all doped regions was not possible.

To achieve this, the geometry was refined to include "dummy" regions with no physical meaning for the device parameters. These regions were implemented such that it was possible to more precisely define a characteristic length in the regions of interest (similar to Fig. 4.2b). This was done while keeping the mesh in the absorption region less dense (similar to Fig. 4.2a).

A comparison of the meshes presented in Fig. 4.2 and the improved mesh in Fig. 4.3 is shown in Table 4.1. This shows the total number of nodes and elements in the meshes.

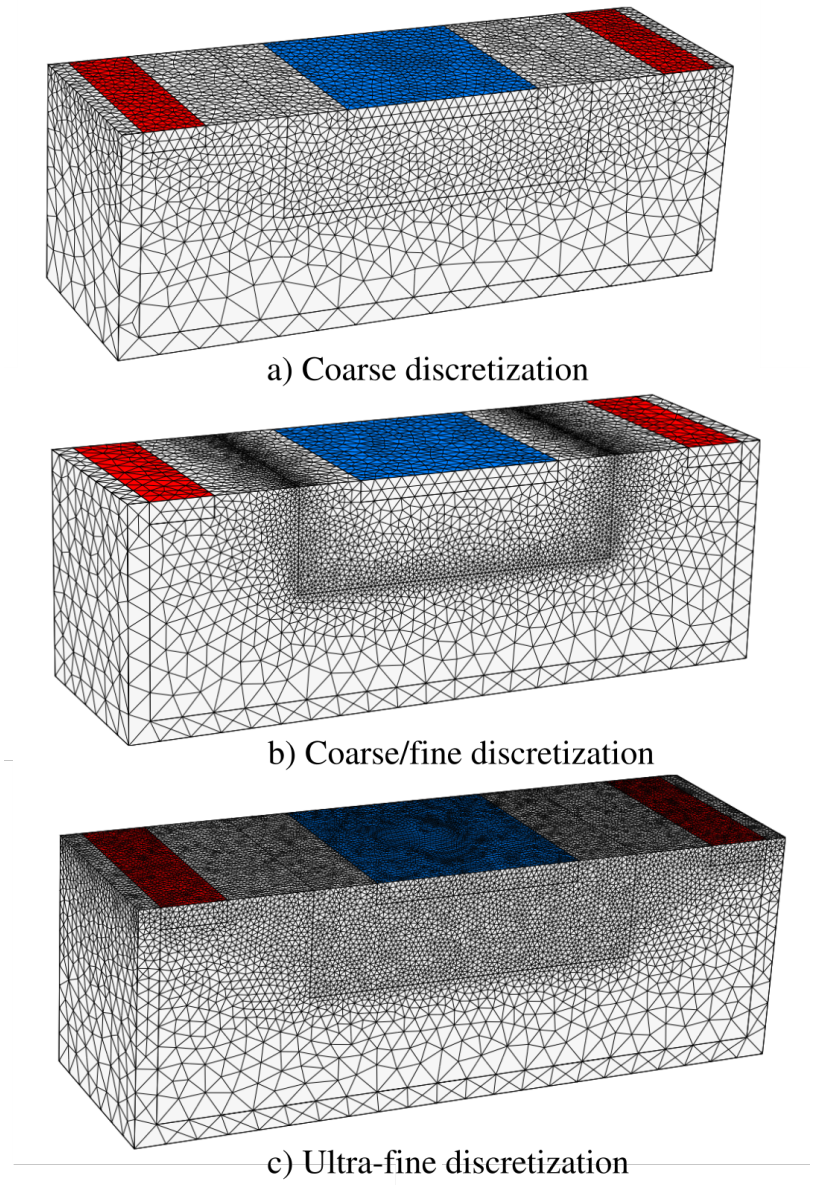
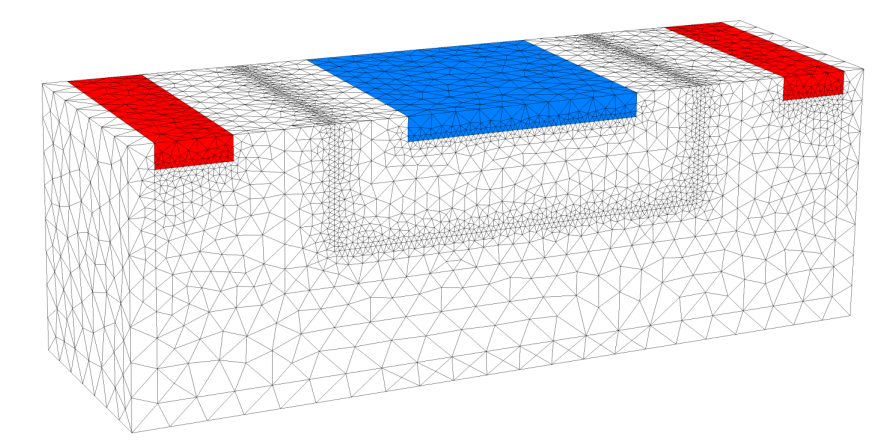


Figure 4.2: Previous iterations of the mesh for the APD device as created by Fatnes [26]. The meshes are classified after their discretization density from coarse (a) to (c) ultra-fine. The mesh in (b) is selectively fine in the region around the *pn*-junction. but coarse in the other regions.

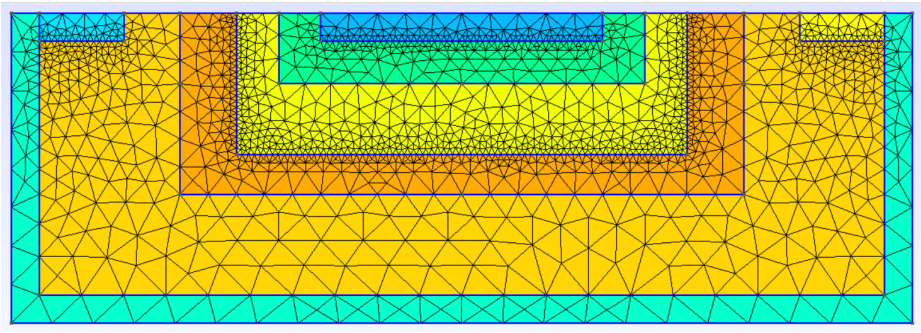
Not surprisingly, the improved mesh contains less nodes due to the linear element type. Compared to the most coarse old mesh (4.2a) it has a factor of one third the number of nodes. It also reduces the number of elements compared to the previously selectively refined mesh (4.2b).

	4.2a	4.2b	4.2c	Improved mesh
# Nodes	77,529	240,914	455,931	24,950
# Elements	52,574	172,599	323,866	134,828
Element type	quadratic	quadratic	quadratic	linear

Table 4.1: A comparison of the old meshes with the new mesh. The total number of nodes and elements are listed as well as the element type.



a) 3D view of improved mesh



b) 2D view of improved mesh

Figure 4.3: The optimized mesh of the APD is shown in 3D (a) and with increased contrast in 2D (b). The colors signify the different regions defined in the geometry.

4.4 Increased simulation time

4.4.1 Background

One of the caveats of simulation runs in the prior version of the simulator was the lack of simulation runs exceeding $\sim$ 10 picoseconds. Previously this have often corresponded with developers who have used more time on developing large and new implementations in the simulator and re-factored significant pieces of the code. As new implementations have been needed, the computational cost also increases with additional functionality making optimization a lower priority.

Based on simulations by researchers at FFI we have seen that the first 100 picoseconds of simulation of a similar device is still in a transient state [15]. To observe the response and reach a more steady state, longer simulations are necessary. However, due to the time frame of this thesis and the current issues in the simulator, such long simulations will likely not be useful.

But to observe the response in greater detail and to verify if or if not the erroneous behavior of particles persist or if new behavior occurs, it is desirable to attempt to run longer simulations for more than 10 picoseconds.

4.4.2 Results

As demonstrated in the previous sections, the many issues with the particle behavior in the device makes it difficult to run longer simulations that can output meaningful physical data. As these issues was not sufficiently resolved in this work, the longer simulations did prove as useful as initially thought.

Still, to test the new mesh created during this work, longer simulations was possible to carry out, despite the persistent issues of zero-velocity holes. This proved viable with simulations that were able to run more efficiently than previously. The major part of this spike in efficiency is due to refinement and optimization of the mesh, as described in 4.3. Another factor is the access to a high-performing computing cluster at NTNU, which allows for more memory, faster storage, high uptime and computing resources than for previously used computers. Details regarding this cluster was given in 3.3 in Chapter 3.

In this work, the simulation time has been increased to 40 ps with timesteps of 1 fs – a total of 40,000 timesteps. This simulation was done with the improved mesh shown in 4.3. The simulation ran for approximately 53 hours which leads to a runtime of 80 minutes per simulated ps. A visualization of the average potential along the z-direction is shown in Figure 4.4. It exhibits similar behavior as those included in Fatnes' simulations at early stages, but it is seems like a potential drop is established along the *pn*-junction in 4.4c at 16 ps. However, the same behavior of maximum potential near the $p+$ regions is observed yet again.

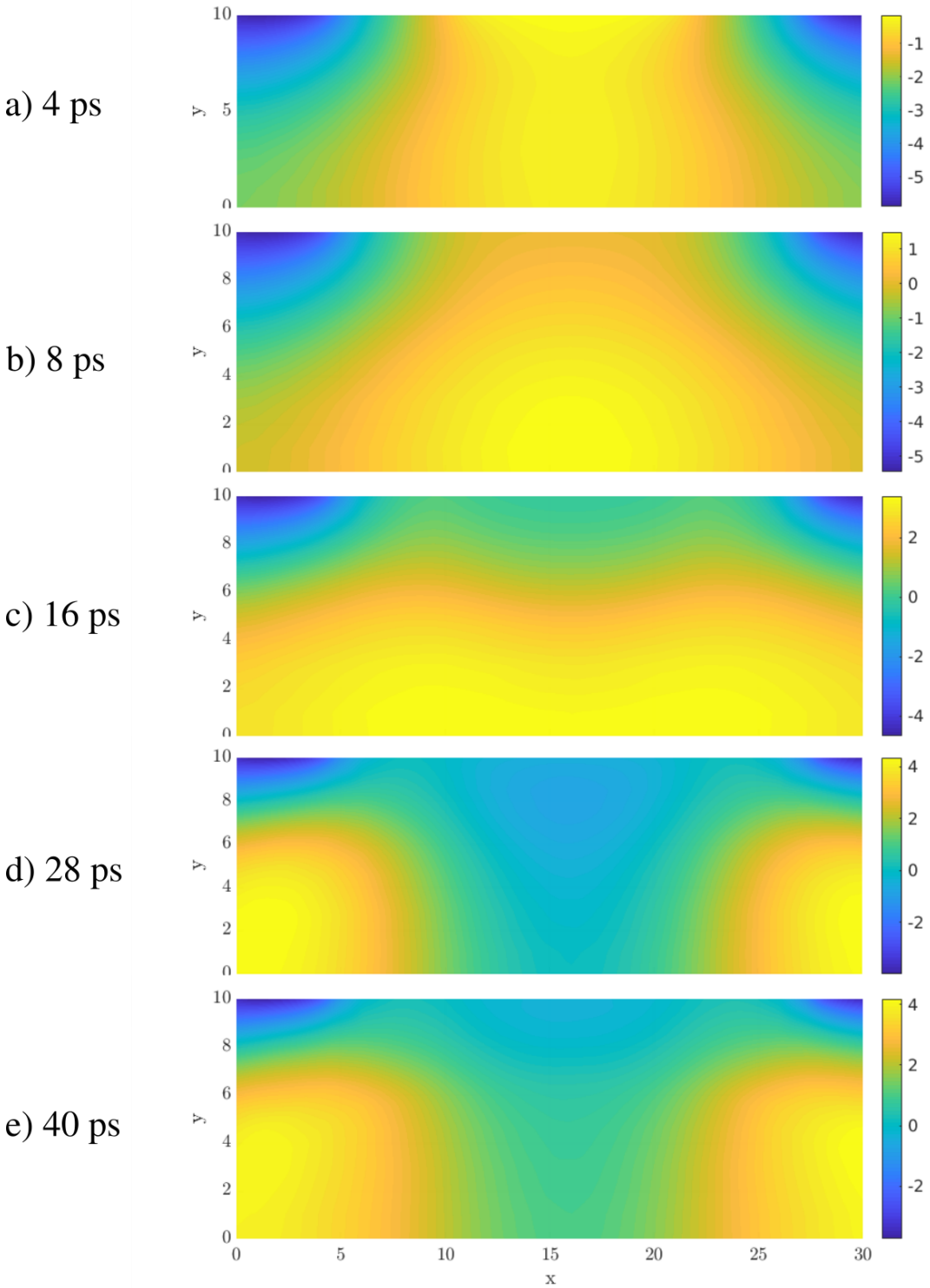


Figure 4.4: The average potential in z-direction for the 40 ps long simulation with the optimized mesh is shown for various times during the simulation.

4.5 Further Work

To continue improving the simulator, the most urgent issue is the one of zero-velocity holes during simulations. The main focus with further development must thus be to continue the effort to resolve this.

When this is resolved the other parts still need further testing and verification, before new implementations should be introduced in the simulator. A considerable list of cases for further testing based on Fatnes' implementations is outlined in the section for further work in her thesis [26]. This should make out for several possible projects for students with a background in physics and numerics. Possible approaches could be focused towards mathematics (improving the implemented algorithms used in calculations) or physics (improving various aspects of the physical model).

Based on the work in this thesis, the following points are suggested as the next steps in further work with the simulator:

Resolving Zero-Velocity Hole Behavior

As mentioned in 4.2.1, a simple solution without changing the band model is suggested. However, for coming developers who seek to refine or develop a new band model, this could be an idea for a combined specialization and master's project.

Continue A Simulation From A Previous Run

Since specialization and master's projects often is limited in time and a good portion of the time is needed to get familiar with the simulator it is desirable to optimize how simulations are run with longer simulation times. An approach to this is to implement an alternative initialization routine where the last output from a previous simulation run is used as input. If one have already simulated 40 ps and the data looks promising for an even longer simulation run, a new simulation could simply continue from 40 ps and further by using the last state of previous run as input. This is implemented in the non-student version of the simulator used at FFI and would be a big improvement for this simulator as well.

Parallelization

As the access to a powerful computing cluster, there are several techniques which can be utilize to parallelize the program which can both speed up simulations run on one computer or distribute the workload across nodes in a cluster. This could also decrease the time needed to run simulations on less powerful hardware as well. This however is not something that can be carried out easily without changing important parts of the algorithm implementations, but should be considered. Work done with this topic could also be of use for the non-student MC simulator.

Migrate To ifort Compilation

To align with the development of the non-student MC simulator at FFI and make use of Intel's latest optimizations for its processors (which the simulator uses), it is proposed to continue the effort to migrate to using Intel's *ifort* as the compiler of the simulator.

Despite the effort to compile with *ifort* as described in 4.1.1 did not prove successful, it seems likely that this would be more viable when the problem of zero velocity-holes is resolved. As this should limit the ever increasing k-values for the holes, a segmentation fault should be able to be avoided more easily. If the error should be introduced again, using debugging compiling options within *ifort* could be considered to identify possible other sources of the issue.

Another feat of moving to this compiler would be to further investigate if there are some of the algorithms used in the simulator which could be optimized by using LAPACK and BLAS routines which are included in *ifort*. These are highly optimized linear algebra routines implemented in machine-optimized code which outperforms the native equivalents in standard Fortran.

Chapter 5

Summary

Monte Carlo software with finite element Poisson solver (MCFEM), th simulator software developed over several years by students at the Norwegian Defence Research Establishment (FFI), has in this 5 month-long Master's project been further tested and developed, with a model of a mercury cadmium telluride (HgCdTe) avalanche photodiode (APD) serving as the simulated device. As the simulator received many significant mathematical implementations by the previous developer, the focus has been on further testing the simulator and increasing stability.

Building on the most recent efforts by Fatnes in 2017 – 2018, several issues have been identified and further investigated. It has been demonstrated that the simulator is capable of running simulations up to 40 ps, only limited by time and computing resources to continue longer. By utilizing a nodes at a high-performance compute cluster, we have achieved to run the simulator at faster speed and with more memory available than before allowing for these longer runs.

By introducing a more detailed meshing geometry it has been shown that we can increase the mesh density around smaller areas of interest while keeping the mesh less dense in larger regions. This improvement has reduced the total number of nodes and elements while keeping the detail level sufficiently high.

Stability issues of the solver has been investigated by simulating devices with varying neutral height of the n-contact region, and the issue of zero velocity-holes have been thoroughly investigated in the code. A proposed solution to this has also been presented.

There is regrettably data from many of the simulation runs not included in this thesis, which deserved to be included and thoroughly discussed. With more time this could have been done, but hopefully this data could be of help to the next developer.

Despite that the implementation of the fix for the zero velocity-hole problem is not completed, the proposed solution serves as a further work for the next developer. When that issue is fully resolved, the physical model would be greatly improved and allow for reaching a steady state in the simulation with longer simulation runs. The many implementations in the simulator still need further development with focus on stability and efficiency,

but at the current time it shows a good potential for further improvements.

Bibliography

[1] Wikipedia. *Timeline of electrical and electronic engineering*. Website. https://en.wikipedia.org/wiki/Timeline_of_electrical_and_electronic_engineering Accessed: 2019-01-28. 2011.

[2] S. M; Sze and K. NG Kwok. *Physics of Semiconductor Devices*. 3rd. Wiley, 2007. ISBN: 9780471143239.

[3] Wikipedia. *Photodiode*. Website. https://en.wikipedia.org/wiki/Photodiode Accessed: 2019-03-07. 2019.

[4] C Snowden. *Semiconductor Device Modelling*. London: Springer, 1989. ISBN: 978-1-4471-1033-0. DOI: 10.1007/978-1-4471-1033-0.

[5] Wikipedia. *Semiconductor device modeling*. Website. https://en.wikipedia.org/wiki/Semiconductor_device_modeling Accessed: 2019-03-07. 2017.

[6] Ravinder Pal. "Infrared technologies for defence systems". In: *Defence Science Journal* 67.2 (2017), pp. 133–134. ISSN: 0976464X. DOI: 10.14429/dsj.67.11223.

[7] FFI. *FFIs Historie nr 11*. PDF. https://www.ffi.no/no/Publikasjoner/Documents/FFIs-historie-nr11.pdf Accessed: 2019-01-31. 2003.

[8] FFI. *FFIs Historie nr 1*. PDF. https://www.ffi.no/no/Publikasjoner/Documents/FFIs-historie-nr1.pdf Accessed: 2019-01-31. 2003.

[9] FFI. *FFIs Årsrapport 2012*. PDF. https://www.ffi.no/sites/aarsrapport2013/dette-er-ffi/kalender/Sider/De-intelligente-missilene.aspx Accessed: 2019-01-31. 2013.

[10] R. Haakenaasen et al. "Imaging one-dimensional and two-dimensional planar photodiode detectors fabricated by ion milling molecular beam epitaxy CdHgTe". In: *Journal of Electronic Materials* 34.6 (2005), pp. 922–927. ISSN: 03615235. DOI: 10.1007/s11664-005-0043-3. arXiv: 1211.5586.

[11] R. Haakenaasen et al. "Imaging photovoltaic infrared CdHgTe detectors". In: *Physica Scripta T* T126 (2006), pp. 31–36. ISSN: 02811847. DOI: 10.1088/0031-8949/2006/T126/007.

[12] R. Haakenaasen et al. “HgCdTe research at FFI: Molecular beam epitaxy growth and characterization”. In: *Journal of Electronic Materials*. 2010. ISBN: 0361-5235\r1543-186X. DOI: 10.1007/s11664-010-1211-7.

[13] Asta Katrine Storebø, Trond Brudevoll, and Knut Stenersen. “Calculated temperature rise in midinfrared laser irradiated Hg0.72 Cd0.28 Te”. In: *Journal of Applied Physics* 103.5 (2008), p. 9. ISSN: 00218979. DOI: 10.1063/1.2890751.

[14] Asta Katrine Storebø, Trond Brudevoll, and Knut Stenersen. “Numerical modeling of IR-laser-irradiated HgCdTe”. In: *Journal of Electronic Materials* 39.10 (2010), pp. 2220–2232. ISSN: 03615235. DOI: 10.1007/s11664-010-1321-2.

[15] Asta Katrine Storebø and T Brudevoll. “Modeling of a Back-Illuminated HgCdTe MWIR Avalanche Photodiode with Alloy Gradients”. In: *Journal of Physics: Conference Series*. IOP Publishing, 2015. DOI: 10.1088/1742-6596/647/1/012051.

[16] Asta Katrine Storebø, Dara Goldar, and Trond Brudevoll. “Simulation of infrared avalanche photodiodes from first principles”. In: *PROCEEDINGS OF SPIE* (2017). DOI: 10.1117/12.2262473.

[17] Trond Brudevoll. *Monte Carlo Software for Charge Transport and Electro-optic Applications*. 2018.

[18] Antoni Rogalski, Małgorzata Kopytko, and Piotr Martyniuk. “Chapter 1 - Infrared Detector Characterization”. In: *Antimonide-based Infrared Detectors: A New Perspective* April (2018). ISSN: 02613069. DOI: 10.1016/j.matdes.2009.12.003.

[19] K. K. Choi et al. “Optics research at the U . S . Army Research Laboratory”. In: *Applied Optics* 56.3 (2017), B103–B115. DOI: 10.1364/AO.56.00B103.

[20] Øyvind Olsen. “Construction of a Transport Kernel for an Ensemble Monte Carlo Simulator”. Master’s Thesis. NTNU, 2009.

[21] Ole Christian Norum. “Monte Carlo Simulation of Semiconductors – Program Structure and Physical Phenomena”. Master’s Thesis. NTNU, 2009.

[22] Øyvind Skåring. “Ultrashort Relaxation Dynamics in Laser Excited Semiconductors”. Master’s Thesis. NTNU, 2010.

[23] Camilla N. Kirkemo. “Monte Carlo Simulation of PN-Junctions”. Master’s Thesis. University of Oslo, 2011.

[24] Jonas Julius Harang. “Utregning av det elektromagnetiske feltet i Monte Carlo transport simulering”. Master’s Thesis. NTNU, 2017.

[25] David Kristian Åsen. “Self-Force Reduced Finite Element Poisson Solvers for Monte Carlo Particle Transport Simulators”. Master’s Thesis. NTNU, 2016.

[26] Siri Narvestad Fatnes. “A Three-Dimensional Finite Element Poisson Solver for Monte Carlo Particle Simulators”. Masters’s Thesis. NTNU, 2018.

[27] Charles Kittel. *Introduction to solid state physics*. 8th ed. 2005, p. 703. ISBN: 0-471-11181-3.

[28] Xiaoli Sun et al. "HgCdTe avalanche photodiode detectors for airborne and spaceborne lidar at infrared wavelengths". In: *Optics Express*. Vol. 25. 14. Optical Society of America, July 2017, p. 16589. DOI: 10.1364/OE.25.016589.

[29] Anand Singh, Vanya Srivastav, and Ravinder Pal. "HgCdTe avalanche photodiodes: A review". In: *Optics and Laser Technology* (2011). ISSN: 00303992. DOI: 10.1016/j.optlastec.2011.03.009.

[30] Janez Krc and Marko Topic. *Optical Modeling and Simulation of Thin-Film Photovoltaic Devices*. 2013. DOI: 10.1201/b14551.

[31] Art B Owen. *Monte Carlo theory, methods and examples*. 2013.

[32] Morten Hjorth-Jensen. *Computational Physics, Lecture Notes Fall 2015*. Oslo, 2015.

[33] Jianming Jin. *The Finite Element Method in Electromagnetics*. 3rd. Wiley-IEEE Press, 2015. ISBN: 1118571363, 9781118571361.

[34] COMSOL. *The Finite Element Method (FEM)*. Website. https://www.comsol.com/multiphysics/finite-element-method Accessed: 2019-03-07. 2016.

[35] Christophe Geuzaine and Jean-François Remacle. "A three-dimensional finite element mesh generator with built-in pre- and post-processing facilities". In: *Int. J. Numer. Meth. Engng.* 79.11 (2017), pp. 1309–1331. ISSN: 03770273. DOI: 10.1002/nme.2579. arXiv: 1010.1724.

[36] Sun Microsystems. *What Every Computer Scientist Should Know About Floating-Point Arithmetic*. Article. https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html Accessed: 2019-03-18. 1991.

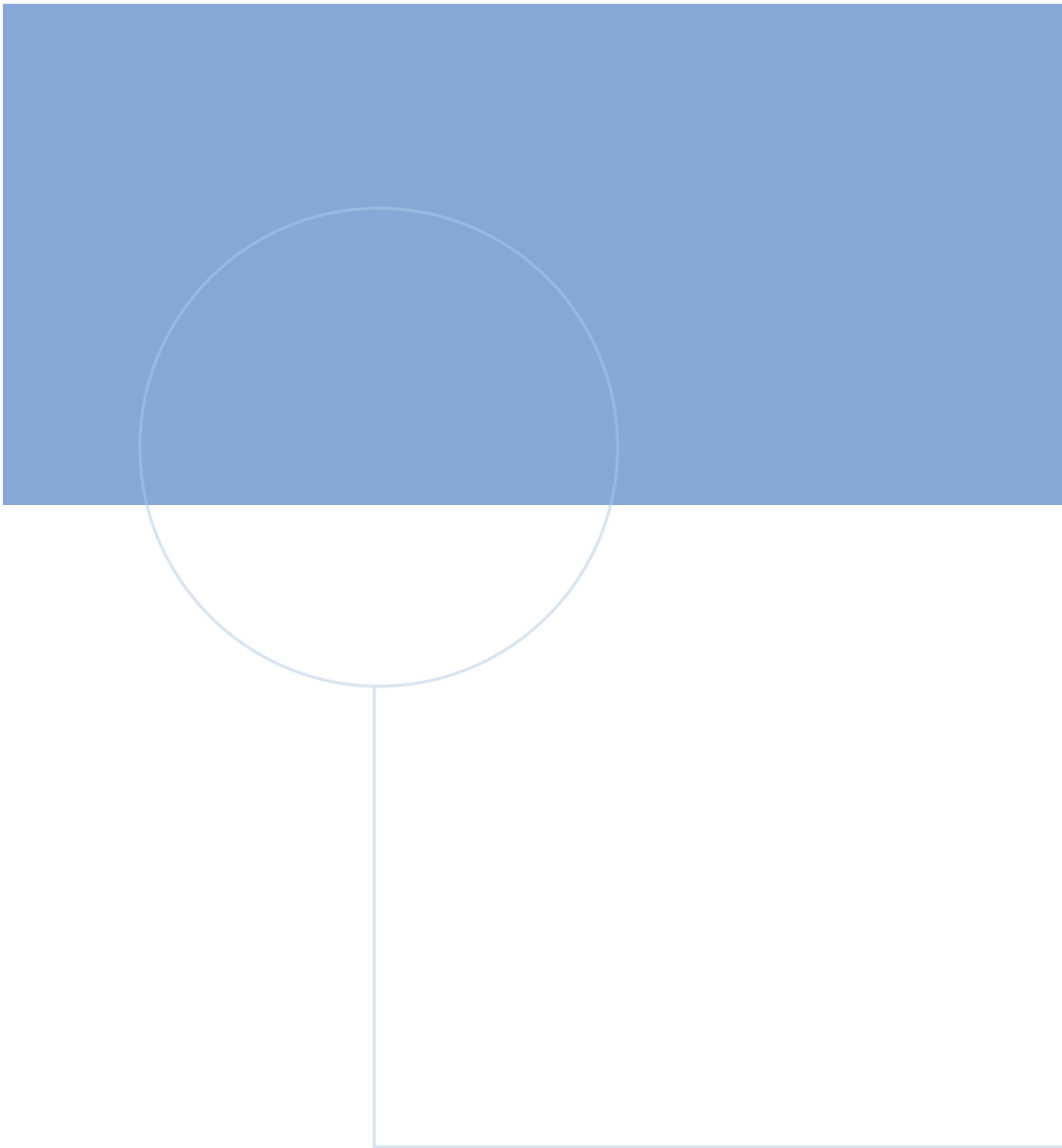
[37] gfortran. *gfortran — the GNU Fortran compiler, part of GCC*. Website. https://gcc.gnu.org/wiki/GFortran Accessed: 2019-03-18. 2019.

[38] Brandon Parker. *Introduction to MPI and OpenMP. Workshop: High Performance Computing on Stampede 2, Jan. 23, 2017*. 2017.

[39] J. P. Laurenti et al. "Temperature dependence of the fundamental absorption edge of mercury cadmium telluride". In: *Journal of Applied Physics* 67.10 (1990), pp. 6454–6460. ISSN: 00218979. DOI: 10.1063/1.345119.

[40] Lars Koesterke. *Modern Programming Languages: Fortran 90/95/2003/2008*. Website. https://www.tacc.utexas.edu/documents/13601/162125/fortran_class.pdf Accessed: 2019-01-28. 2011.

[41] Geir Uri Jensen. "Monte Carlo Simulation of III-V Semiconductor Devices". Doctorate's Thesis. NTNU, 1989.



NTNU
Norwegian University of
Science and Technology