NTNU – Trondheim
Norwegian University of
Science and Technology

Implementing a Full-Band Monte Carlo Model for Zincblende Structure Semiconductors

Tore Sivertsen Bergslid

Physics
Submission date: August 2013
Supervisor: Jon Andreas Støvneng, IFY
Co-supervisor: Trond Brudevoll, Forsvarets Forskningsinstitutt
Asta-Katrine Storebø, Forsvarets Forskningsinstitutt

Norwegian University of Science and Technology
Department of Physics

ABSTRACT

English:

During the work with this master's thesis a number of improvements have been made to the Monte Carlo program being developed at FFI. Algorithms for handling numerical band and scattering rate data have been constructed and integrated with the program. Of all the changes made in this work, most important is the fact that the program has been made capable of running with band structures and scattering rates calculated by the $\mathbf{k} \cdot \mathbf{p}$-method, leaving the less accurate analytical approximations behind. The program is now capable of running bulk Monte Carlo simulations using a full-band model for the valence bands. All important infrastructure is also set up for adding full-band versions of other bands.

Norsk:

Under arbeidet med masteroppgaven har det blitt gjort flere forbedringer til Monte Carlo programmet som er under utvikling ved FFI. Det har blitt konstruert algoritmer for å håndtere numerisk bånd- og spredningsrate-data og disse har blitt integrert i programmet. Av alle endringene som har blitt gjort er den viktigste at programmet har blitt gjort i stand til å kjøre med båndstruktur og spredningsrater beregnet av $\mathbf{k} \cdot \mathbf{p}$-metoden, og de analytiske approksimasjonene har blitt erstattet av disse. Programmet er nå i stand til å kjøre bulk Monte Carlo simuleringer ved å bruke en fullbåndsmodell for valensbåndene. All nødvendig infrastruktur er også tilstede for å legge til fullbåndsmodeller for andre bånd.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

There are two common ways of simulating charge carrier transport in semiconductors: Solving the transport equations explicitly, or by Monte Carlo (MC) simulations. The first option relies on direct equation solving techniques, while the latter is based on simulating each individual particle through a series of free flights terminated by scattering events, thus solving the Boltzmann transport equation indirectly.

This work is a continuation of an MC simulator project started at FFI in 2007. Since the beginning, several students have worked on bringing the MC particle simulator into its current state [1, 2, 3, 4, 5, 6, 7]. The main goal of the project is to develop a state of the art MC simulation tool for both bulk semiconductors as well as devices, in order to help progress the photovoltaic infrared detector development happening at FFI. Being able to simulate and learn how various devices work without actually building them is an invaluable resource to have for such purposes.

$Hg_{1-x}Cd_xTe$, mercury cadmium telluride (MCT), is an interesting semiconductor. It is commonly used as a detector material for infrared photon detectors, which is why it is of interest for this project. The combination of the semimetal HgTe and the wide band gap semiconductor CdTe means that the band gap can be adjusted to anywhere from 0 to 1.5 eV by changing the temperature and alloy fraction x. These properties, along with its unique impact ionization properties, make it a very attractive material for use in high sensitivity avalanche photo diodes (APD). Using the older analytical model to study APDs was the focus of two earlier master's theses [4, 5].

A vast majority of the workload in producing this thesis was restructuring old code, implementing new algorithms and debugging. Preparatory work began

during a summer job in 2012. Initially, this meant reading old theses written on the subject and also reading through the entire program, some 16 000 lines at the time, in order to gain an understanding of how the program was put together. After learning how the MC program works, Halvorsen's programs were studied extensively and their structure updated to prepare them for implementation into the MC program. These programs, now subroutines, were originally written as standalone programs, necessitating a quite drastic restructuring in order to make them compatible. During this process a number of small test programs were written; some of which served as the basis for test programs still in use.

Functionality for running band and rate calculations from the main menu of the MC program was then implemented. This integrated the Halvorsen programs with the MC program, allowing the user to set all parameters from one place, instead of having to go via separate programs like before.

Work then began on finding a good interpolation routine for the bands and rates. Implementing this and other small changes necessary to make them work was a time consuming task.

The final big algorithm to be implemented was a way of finding a suitable wave vector for a given energy.

A decent chunk of time also went into fixing minor bugs in the program.

At the end of this work the program now sits at nearly 21 000 lines of code, plus a few thousand more in test programs.

Chapter 2 discusses the theory behind some of the most important routines in the MC program. In chapter 3 the algorithms implemented during this work are presented in great detail. There are also discussions regarding problems encountered during the implementation process and how they were solved, as well as a look at how well the routines perform. Chapter 4 contains some results from a full run of the MC program, in order to show that our initial goals have been achieved. Finally, chapter 5 sums up what has been accomplished and talks about ideas for future work.

CHAPTER 2

THE MONTE CARLO METHOD

Monte Carlo methods, as applied to semiclassical charge transport in semiconductors, consist of a simulation of carriers influenced by electric and magnetic fields, and by the scattering rates. Semiclassical transport models treat carrier scattering according to quantum mechanical laws, with the carrier movement between scattering events regarded as a classical process. Scattering rates tell us how many scattering events a given carrier will undergo per unit time. When the analytical band model is in use, the duration of the free flight and the new direction of the wave vector $\mathbf{k}$ after scattering are both selected stochastically according to some given probability for each mechanism. In the full-band model the new direction of the wave vector is not chosen at random. Instead of finding a new absolute value, $|\mathbf{k}|$, and choosing a direction stochastically, the routine determines each individual component of the vector.

In this chapter we shall only give an outline of the procedures for finding the new $\mathbf{k}$-state in cases where simple scattering mechanisms are involved, such as phonon mediated and impurity mediated carrier scattering, starting with a description of the technique for calculating the free flight duration in section 2.1. Section 2.2 talks about how to determine what type of scattering occurs. What happens after a particle has been scattered is discussed in section 2.3. Section 2.3.1 explains briefly what is done after scattering in the case of analytical band models in the MC program. This is included because analytical bands are what the program has relied on up until now. Finally, in section 2.3.2, there is a quite detailed look at what is done in the case where the full-band model is used. Many of these subjects are covered in more detail in Brudevoll's PhD thesis [8] and/or Halvorsen's master thesis [9].

2.1 Free flight duration

If we denote the total scattering rate by $S(\mathbf{k}(t))$ and the probability that the interval (0,t) does *not* contain a scattering by $Q(t)$, the following relation emerges:

$$Q(t+dt) = Q(t) \cdot [1 - S(\mathbf{k}(t))\, dt]\,, \tag{2.1}$$

or

$$Q'(t) + S(\mathbf{k}(t)) \cdot Q(t) = 0, \tag{2.2}$$

which has the solution

$$Q(t) = \exp\left[-\int_0^t S(\mathbf{k}(t'))\, dt'\right]. \tag{2.3}$$

We must now find the probability density $p(t)$ for the duration of the free flight. The probability that the flight ends in a time interval dt around t is

$$dP(t) = p(t)\, dt = Q(t)S(\mathbf{k}(t))\, dt, \tag{2.4}$$

where $P(t)$ is the cumulative probability. That is, the probability density that the flight will end at t is equal to the probability that it has not ended yet, $Q(t)$, times the probability that it will end in the small time interval around t, $S(\mathbf{k}(t))dt$. Using equation 2.3, the distribution we seek for the free flight time may be written

$$p(t) = S(\mathbf{k}(t)) \cdot \exp\left[-\int_0^t S(\mathbf{k}(t'))\, dt'\right]. \tag{2.5}$$

If we generate random numbers r evenly distributed on the interval (0,1) and solve the equation $r = P(t)$ for the corresponding t's, it can be shown that the distribution of the scattering times t will be in accord with $p(t)$. To see why, we merely note that the probability of r lying on an interval dr around a specific r' is equal to the probability that t lies on the interval dt around a specific t' (correspondence one to one between the function r and the variable t through the relation $r = P(t)$). Due to the uniform distribution of r, the probability of r lying in the interval dr is simply $dr = dP(t) = p(t')dt$. The number of occurrences of t's from a particular interval dt is proportional to the corresponding width dr and therefore proportional to the quantity $p(t')$. This method for generating distributions is called the direct technique.

Integration of $p(t)$ from 0 to t will give $P(t)$. However, $P(t)$ is the probability that the particle has been scattered within the interval (0,t), and therefore we may write down the result directly as

$$P(t) = 1 - Q(t) = 1 - \exp\left[-\int_0^t S(\mathbf{k}(t'))\, dt'\right], \tag{2.6}$$

and the procedure for using the direct technique is to let $r = P(t) = 1 - Q(t)$.

Since r is a random number between 0 and 1, it is completely equivalent with $1 - r$. Hence we replace $1 - r$ with r, and taking the natural logarithm on both sides of the new equation yields the final relation:

$$\ln r = \ln Q(t) = -\int_0^t S(\mathbf{k}(t'))\, dt' \,. \tag{2.7}$$

To solve this equation for t, we add a fictitious scattering mechanism with the rate S^0 to S so that the sum of these two rates is constant within each time interval.

The total rate $S + S^0 = \Gamma(t)$ then replaces S in the equation above. The integral is converted to a sum over fixed time increments, t_{inc}, with a constant value of Γ in each time interval ($\Gamma(t)$ is a staircase function in t), see Fig. 2.1. The value of Γ used within each time interval is equal to or greater than the maximum real scattering rate that the particular carrier in question will encounter during its flight over this time interval.

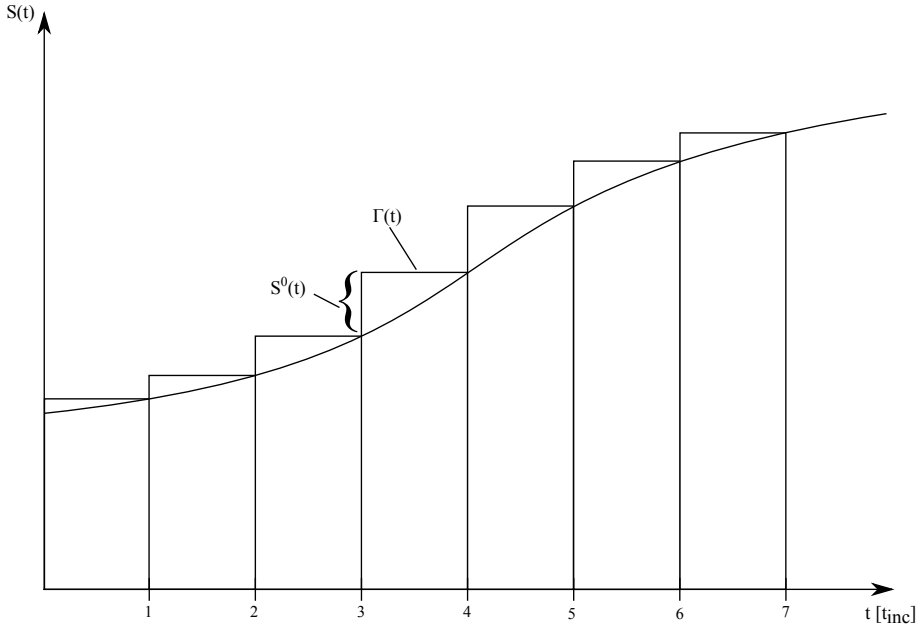


Figure 2.1: Self-scattering with the constant-time method of Yorston.

Once a new interval has been evaluated, the accumulated value on the right hand side is checked. If this value exceeds the value of the random number $\ln(r)$, the free flight time is found in this time interval; if not, another time step

is included. If the mechanism S^0 is the chosen one at the end of a free flight, we have a self-scattering, and the carrier will continue with its $\mathbf{k}$ unchanged. If a real scattering mechanism is chosen, $\mathbf{k}$ will be changed and the whole process of evaluating the integral starts all over again. This particular version of the self-scattering scheme is called the constant-time method, and it was given by Yorston [10].

2.2 Choosing the scattering mechanism

The specific scattering mechanism i is selected by generation of a random number r between 0 and 1 according to the relation

$$\sum_{i=0}^{j-1} S^i(\mathbf{k}_e) < r \cdot \Gamma(\mathbf{k}_e) < \sum_{i=0}^{j} S^i(\mathbf{k}_e), \tag{2.8}$$

where $\mathbf{k}_e$ is the wave vector at the end of the free flight and $\Gamma(\mathbf{k}_e)$ is the sum of scattering rates from all possible scattering mechanisms at $\mathbf{k}_e$. Scattering mechanism number j is chosen if the above relation holds, see Fig. 2.2. The scattering rates can sometimes be calculated before the start of the simulation, for discrete values of $\mathbf{k}$, as has been done in this work.

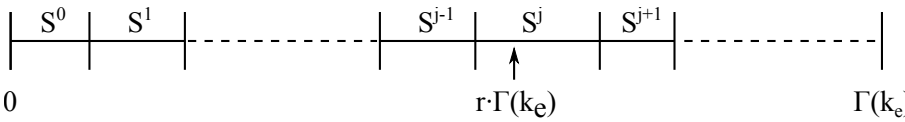


Figure 2.2: Choosing the type of scattering. Mechanism number j is chosen, S^0 is self-scattering.

2.3 The state after scattering

2.3.1 Analytical band models

When the specific scattering mechanism has been determined, the new energy is evaluated from the energy at the end of the free flight, the amount of energy to be absorbed or emitted, and the separation of energy bands where appropriate.

If we have an isotropic (but possibly non-parabolic) band model and a constant energy exchange (e.g. either 0 or $\hbar\omega_0$), the modulus of the new wave vector is given automatically, and only the scattering angles remain to be found. By generating random numbers, the polar and azimuthal scattering angles (θ, ϕ) with respect to the initial wave vector are determined.

The techniques used for generating these angles are fairly standard, and descriptions can be found in the literature. We have the direct technique, the rejection technique, and the combined technique. Other techniques for generating special distribution functions also exist, but we shall not encounter them here.

The azimuthal angle ϕ may be chosen at random with uniform probability, since the angular scattering distribution function in cases with isotropic band structures will in general be a uniform function in this variable. Therefore the polar angle θ is the only one to be found utilizing the techniques mentioned above, making the process of finding the new $\mathbf{k}$ a relatively simple, one-dimensional affair.

If a non-elastic model for phonon scattering is included, with $h\omega_q$ depending on the magnitude of the phonon wave vector $\mathbf{q}$ (isotropic dispersion relation), the modulus of the new $\mathbf{k}$ after scattering will not be known beforehand; it will be a function depending on $\cos\theta$. For example, in the case of acoustic phonon scattering this function cannot be found exactly unless a fourth order algebraic equation involving energy and momentum conservation is solved (linear disperion relation, parabolic bands). Even though solutions of fourth order algebraic equations are well known, they are difficult to deal with in practice, and can alternatively be solved by numerical techniques.

We introduce the angular distribution function $P^i_{\text{ang}}(\Omega)$ (where Ω is the space angle, and i is the scattering mechanism) obtained from the integrand when we integrate the transition probability $P^i_{\text{trans}}(\mathbf{k},\mathbf{k}')$ (for transitions from the initial state $\mathbf{k}$ to a specified final state $\mathbf{k}'$) over all possible final states. Thus, the scattering rate for mechanism i is given by

$$S^i(k) = \frac{V}{(2\pi)^3}\int_{k'} P^i_{\text{trans}}(\mathbf{k},\mathbf{k}')\,d\Omega\,k'^2\,dk', \tag{2.9}$$

where $V/(2\pi)^3$ is the density of states in $\mathbf{k}$-space when assuming periodic boundary conditions (not counting spin) and V is the semiconductor volume. Now, $P^i_{\text{trans}}(\mathbf{k},\mathbf{k}')$ contains an energy conserving δ-function that will vanish when we perform the integration over the angular variable Ω and the radial part of the final-state wave vector $\mathbf{k}'$. The most frequent (and also the most practical) thing to do is to first integrate over the radial part, that is, over the absolute value k' of the final-state wave vector. Then an un-normalized version of the space angular distribution function $P^i_{\text{ang}}(\Omega)$ appears:

$$P^i_{\text{ang}}(\Omega) = \frac{V}{(2\pi)^3}\int_{k'} P^i_{\text{trans}}(\mathbf{k},\mathbf{k}')k'^2\,dk'\,. \tag{2.10}$$

We choose the polar axis of a local coordinate system to lie along the direction of the initial wave vector $\mathbf{k}$. In $P^i_{\text{trans}}(\mathbf{k},\mathbf{k}')$, the angular dependence of both the band structure and the specific scattering mechanism in question are incorporated. Band structures, non-parabolic or not, do not depend on the angles in

the isotropic case, so the carrier-phonon and carrier impurity scattering mechanisms usually only depend on $\mathbf{q} = \mathbf{k}' - \mathbf{k}$, $\mathbf{k}'$ and $\mathbf{k}$. Key relations between these wave vectors can therefore be expressed without involving the azimuthal angle ϕ, only $\cos\theta$ is needed, and therefore the angular distribution function will factorize in the local frame of reference:

$$P^i_{\text{ang}}(\Omega)d\Omega = P^i_{\text{ang}}(\phi) \cdot P^i_{\text{ang}}(\cos\theta)\, d(\cos\theta)\, d\phi\,. \tag{2.11}$$

Note that here and in the following we use the term "factorize" in the meaning factorize in two functions; one depending on the polar angle and one depending on the azimuthal angle.

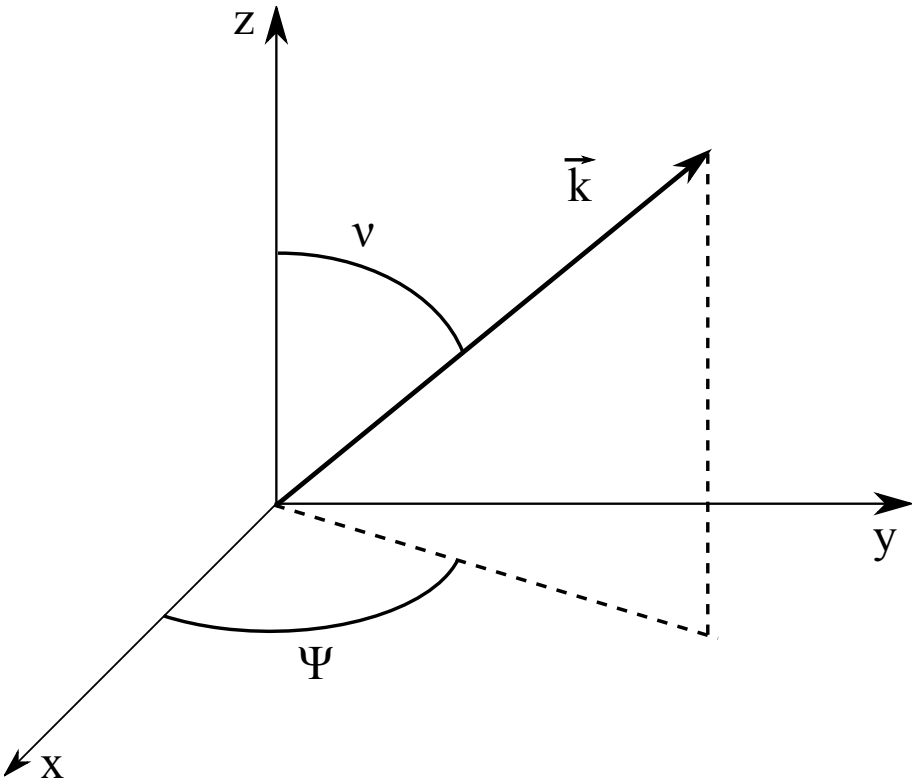


Figure 2.3: The global coordinate system of the simulation.

Accordingly, choices of ϕ and $\cos\theta$ can be made independently. In addition, $P^i_{\text{ang}}(\phi)$ will be uniform, making the choice of ϕ a trivial matter. Once the new angles relative to the initial wave vector $\mathbf{k}$ are found, the orientation of $\mathbf{k}$ relative to a fixed, global frame of reference for the simulation (Fig. 2.3) is taken into account, and the global orientation of $\mathbf{k}'$ is obtained through standard coordinate transformations as given below.

As we have seen above, ϕ and $\cos\theta$ are the natural random variables for the an-

gular distribution function. If the endpoints of the various $\mathbf{k}'$-vectors generated are uniformly distributed over the surface of a sphere, we have a uniform angular distribution, described by the uniform distributions $P^i_{\text{ang}}(\phi)$ and $P^i_{\text{ang}}(\cos\theta)$.

If we have a complex, warped band structure, such as in the case of holes, $P^i_{\text{ang}}(\Omega)$ will not factorize, regardless of the choice of reference frames for the angular variables (θ, ϕ), i.e. the random variables $(\cos\theta, \phi)$.

The situation is not at all like that of electrons in an ellipsoidal valley, where the Vogt-Herring transformation [11] takes us back to the spherical case with only a modest effort. Taking the local polar axis along the direction of the initial wave vector is not going to help us here, so we abandon that approach completely. Instead, we use as our angular variables the parameters ν and ψ, the polar and azimuthal angles with respect to the fixed global coordinate system of the simulation, Fig. 2.3. This reference frame is fixed with respect to the crystalline axes of the semiconductor.

Standard coordinate transforms as given below are needed to convert the polar and azimuthal variables of the wave vector $\mathbf{k}'$ (given with respect to a coordinate system in which $\mathbf{k}$ is directed along the z-axis, referred to as the local coordinate system) to direction cosines l' in the fixed global coordinate system of the simulation. As we saw in Fig. 2.3, the initial $\mathbf{k}$ has polar and azimuthal variables (ν, ψ) with respect to the fixed coordinate system. See also Fig. 2.4.

The standard coordinate transformations are

$$l'_x = \cos\theta\cos\gamma_1 - \sin\theta\cos\phi\cos\psi\cos\nu + \sin\theta\sin\phi\sin\psi, \quad (2.12)$$

$$l'_y = \cos\theta\cos\gamma_2 - \sin\theta\cos\phi\cos\nu\sin\psi - \sin\theta\sin\phi\sin\psi, \quad (2.13)$$

$$l'_z = \cos\theta\cos\nu + \sin\theta\cos\phi\sin\nu, \quad (2.14)$$

where $\cos\gamma_1 = \sin\nu\cos\psi$ and $\cos\gamma_2 = \sin\nu\sin\psi$.

We see from Fig. 2.4 that there are two possible choices for the direction of the local axis x_{local}; either as shown in Fig. 2.4 or in the opposite direction. Defining x_{local} in the opposite direction would also affect the definition of the local azimuth angle ϕ. The standard coordinate transformations given above are of course only valid if x_{local} and ϕ are defined as shown in Fig. 2.4.

In simulations of electrons, the equivalent valleys are often lumped together and treated as one spherical, non-parabolic valley. In that case, one avoids the problem of a non-factorizing $P^i_{\text{ang}}(\Omega)$. For holes, if a detailed model of the warped bands is desired, the new angles of $\mathbf{k}'$ must be found using straightforward two-dimensional versions of the well-known rejection and combined techniques.

For the unstrained valence bands it is possible to construct piecewise analytical models for the $E(\mathbf{k})$ relation, consisting of several different sections or branches. This has been applied into the present analytical version of the MC model for holes, in an isotropic band model omitting warping but including non-parabolicity.

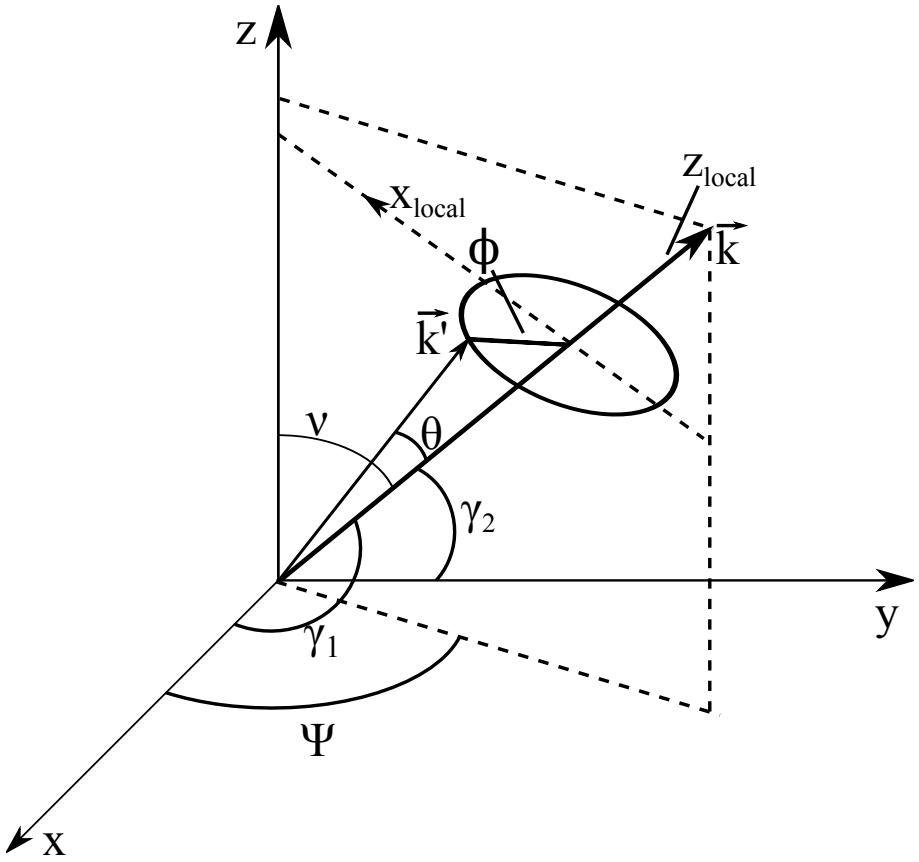


Figure 2.4: Directional cosines and angles of $\mathbf{k}'$ in the global coordinate system.

At high energies or if strain is applied, the warping of the bands becomes so severe that we do not find analytical band models sufficiently accurate and simple enough to be of any help in a MC program. The main problem is often to reproduce a rapid angular dependence of $E(\mathbf{k})$. Therefore, that line of approach must be abandoned, and bands must be entirely described by tables.

2.3.2 Bands described by tables - the full-band model

With no analytical band model present, the only option is to discretize the Brillouin zone. All information concerning the valence bands is stored in tables, calculated by the program written by Halvorsen [9], with the modifications made by Karlsen [12]. Scattering rates for a discrete number of $\mathbf{k}$-points have been calculated in advance, also by the work of Halvorsen. Since many elements of the method used to calculate rates is applied when looking for a final state after scattering, let us review how this is done.

Small cubes of **k**-space are the destination of a carrier that has been scattered, with no distinction in the transition probability to different $\mathbf{k}'$-vectors within such a cube. A search for cubes where energy and momentum conservation may be fulfilled points out the relevant final cubes. The further selection of these cubes is made by giving each of them an individual weight according to the magnitude of the scattering matrix element and the number of final states with the given final energy within each cube.

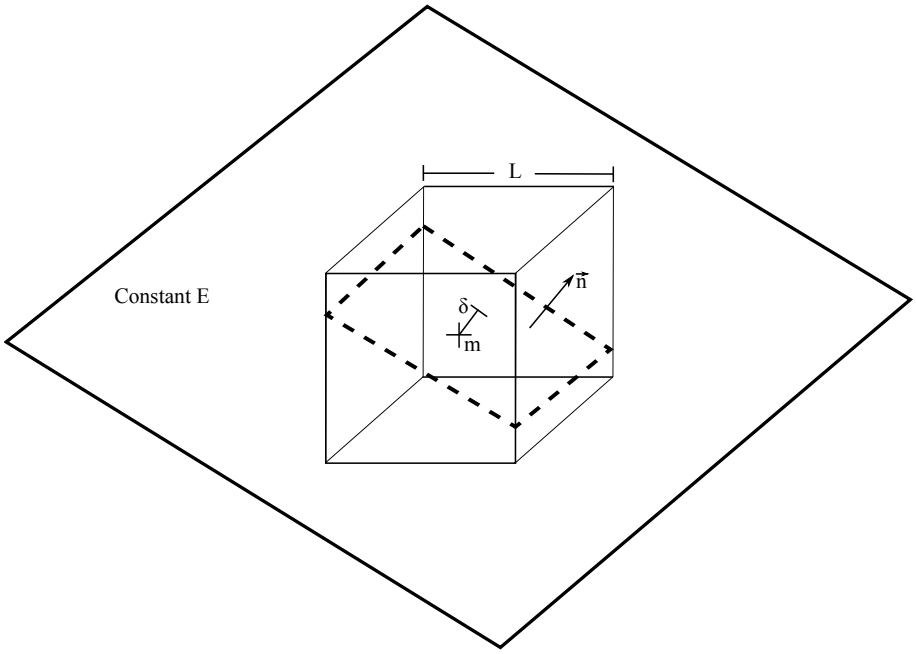


Figure 2.5: A small cube cuts off a portion of the constant-energy surface of the band structure in **k**-space.

Among the data stored in tables are the minimum and maximum energies within each cube, along with data on the gradients and second derivatives of $E(\mathbf{k})$. Derivatives have been calculated by analytical expressions which can be found in Halvorsen's thesis. Constant-energy surfaces are approximated by a plane within each cube, normal to the gradient, and displaced by an increment δ in the wave vector along the positive direction of the gradient at the midpoint m of the cube, Fig. 2.5. This δ is given by the formula

$$\delta = \frac{E(\mathbf{k}) - E(\mathbf{k}_m)}{|\nabla E(\mathbf{k}_m)|}. \tag{2.15}$$

When a plane with given normal sweeps across a cube, the cross-sectional area will be proportional to the number of states between neighbouring constant-energy surfaces inside the cube. For a scattered carrier, these states constitute

the new possible destinations within that particular cube. Other cubes can have more states available, indicated by their larger cross-sectional area. Of course, the shape of this plane of intersection will depend on the direction of its normal, and the area itself will diminish as the plane moves away from the center of the cube. In the case of a constant-energy exchange with the scattering mechanism, the energy conserving surface of the scattering process will coincide with one of the constant-energy surfaces. If the particle undergoes an energy exchange during the scattering process the location of the new energy conserving surface must be established and calculations must be performed with respect to this surface.

Equations for the cross-sectional area of this energy conserving surface, $A(\delta)$, were found by Gilat and Raubenheimer [13], with a minor error corrected by Halvorsen. Let us restate these equations here.

Assume that the components of the unit plane normal $\mathbf{n} = [l_1, l_2, l_3]$ satisfy $l_1 \geq l_2 \geq l_3 \geq 0$, for simplicity. This can be made true for any $\mathbf{n}$ by sorting the components by magnitude. The symmetry of a cube lets us do this without causing any trouble, as this is a purely geometric calculation. Another restriction made possible by the symmetry is to say $\delta > 0$. Let us designate $b = L/2$, with L defined in Fig. 2.5. The distances from the center of the cube to the four corners in the relevant half of the cube are then given by

$$\delta_1 = b|l_1 - l_2 - l_3|, \tag{2.16}$$
$$\delta_2 = b(l_1 - l_2 + l_3), \tag{2.17}$$
$$\delta_3 = b(l_1 + l_2 - l_3), \tag{2.18}$$
$$\delta_4 = b(l_1 + l_2 + l_3). \tag{2.19}$$

The equations for $A(\delta)$ will vary depending on the magnitude of δ. The equations are then

$$A(\delta) = 4b^2/l_1 \text{ when } 0 < \delta < \delta_1 \text{ and } l_1 \geq l_2 + l_3, \tag{2.20}$$
$$A(\delta) = [l_1 l_2 l_3]^{-1}[2b^2(l_1 l_2 + l_1 l_3 + l_2 l_3) - (\delta^2 + b^2)] \text{ when } 0 < \delta < \delta_1 \text{ and } l_1 < l_2 + l_3, \tag{2.21}$$
$$A(\delta) = [l_1 l_2 l_3]^{-1}[b^2(3l_2 l_3 + l_1 l_2 + l_1 l_3) + \delta b(l_1 - l_2 - l_3) - \frac{1}{2}(\delta^2 + b^2)] \text{ when } \delta_1 < \delta < \delta_2, \tag{2.22}$$
$$A(\delta) = 2[l_1 l_2]^{-1}[b^2(l_1 + l_2) - \delta b] \text{ when } \delta_2 < \delta < \delta_3, \tag{2.23}$$
$$A(\delta) = [2l_1 l_2 l_3]^{-1}[b(l_1 + l_2 + l_3) - \delta]^2 \text{ when } \delta_3 < \delta < \delta_4. \tag{2.24}$$

These equations will result in the cross-sectional area taking different shapes. In the order listed above, the shapes are: parallellogram, hexagon, pentagon, quadrangle and triangle.

We will also need to find the overlap factor. This can be found in one of three ways:

- Unity overlap factor.
- Analytical expressions given by Wiley [14].
- Overlap factors calculated from eigenvectors of the $\mathbf{k} \cdot \mathbf{p}$ Hamiltonian.

The first one is simply setting the overlap factor equal to one. The analytical expressions for the overlap factor are

$$G_{\text{intraband}}(\mathbf{k}, \mathbf{k}') = \frac{1}{4}(1 + 3\cos^2\theta), \tag{2.25}$$

$$G_{\text{interband}}(\mathbf{k}, \mathbf{k}') = \frac{3}{4}\sin^2\theta. \tag{2.26}$$

For the third, and most commonly used, way, the equation is

$$G_{nn'}(\mathbf{k}, \mathbf{k}') = \frac{1}{2}\sum_{\mu=1}^{2}\sum_{\mu'=1}^{2} |\mathcal{F}^{+}_{n'\mu'\mathbf{k}'}\mathcal{F}_{n\mu\mathbf{k}}|^2, \tag{2.27}$$

where μ denotes spin state and $\mathcal{F}$ is the eigenvector of the $\mathbf{k} \cdot \mathbf{p}$ Hamiltonian.

Matrix elements are specific to each of the scattering mechanisms, and are not important enough to list here. They can be found in the appendix of Halvorsen's thesis [9].

Once the final cube has been chosen, an adjustment of the final wave vector is necessary. Fischetti and Laux [15] did this by adjusting the $\mathbf{k}$-vector along one of the cube's principal directions (edges, side diagonals, cube diagonal). This is different from the method used by Halvorsen and in this work. Here, the adjustment happens from the center of the cube along the energy gradient until it intersects the energy conserving plane. In Halvorsen's work this is handled by a routine called `CORREC`, and this method has also been adapted and used in finding final states after scattering in the MC program.

We can now summarize the steps needed to find the scattering rates within the technique of tabulated bands, sometimes called the full-band MC technique:

First, all cubes with the correct final energy are found by a searching procedure. This method should be usable on arbitrary band structures. The maximum and minimum energies of each cube have been found earlier and are stored in tables. To find the candidate cubes we then simply run through the entire mesh and store the cubes where the given energy is contained within the maximum and minimum energy of that cube. For each of the cubes suggested by the rejection technique we have to evaluate the scattering matrix element, $M^i_{nn'}(\mathbf{k}, \mathbf{k}')$; the overlap factor, $G_{nn'}(\mathbf{k}, \mathbf{k}')$; the length of the gradient in the mesh point; the distance from $\mathbf{k}_m$ to the energy conserving plane, δ; the shape of the cross section and finally the area of the cross section, $A(\delta)$. Here n and n' are the initial and final bands, i is the scattering mechanism and $\mathbf{k}$ and $\mathbf{k}'$ are the initial and final $\mathbf{k}$-vectors.

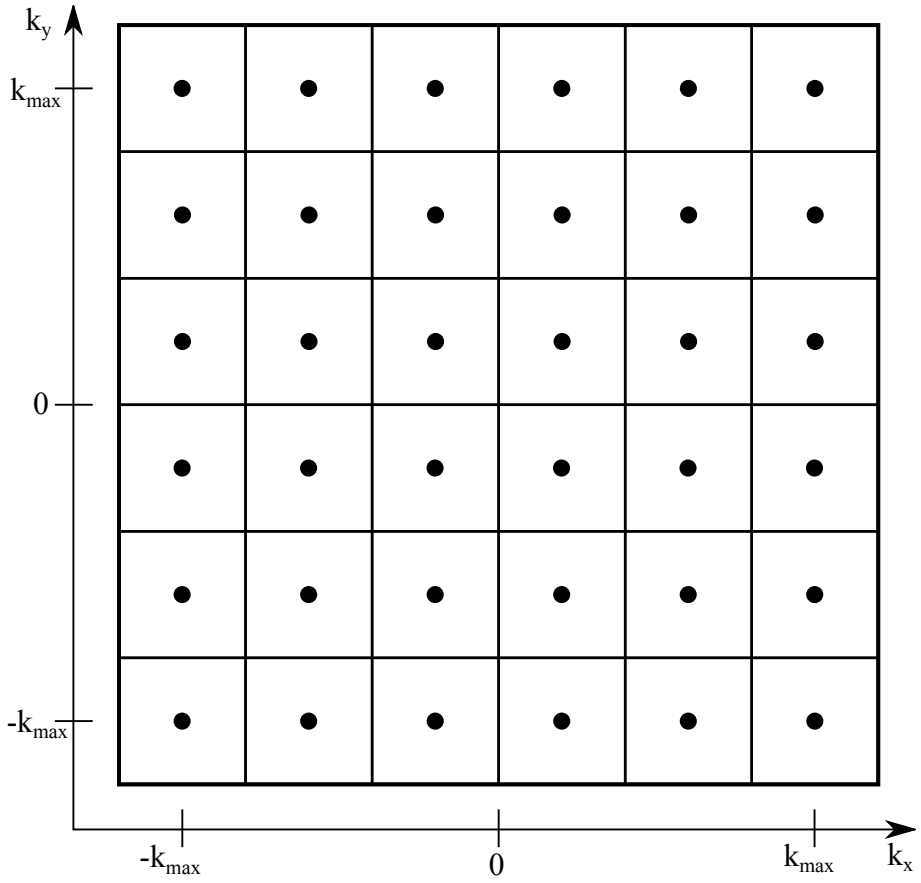


Figure 2.6: Organization of mesh with tabulated bands. Here shown in 2D.

The integration over the radial part $d\mathbf{k}'$ in equation 2.9 is omitted. Instead, integration along the direction normal to the energy conserving surface is performed. This gets rid of the energy conserving δ-function. The integrand $P^i_{\text{trans}}(\mathbf{k}, \mathbf{k}')$ for the scattering rate $S^i(\mathbf{k})$ is calculated together with an estimate of the number of energy conserving final states in each of the possible final cubes. In other words, the integrand is our cube weight, and the integral is converted to a sum over all possible final cubes.

The matrix element is multiplied by the area of the cross section, the $\mathbf{k}$-space density-of-states factor and the overlap factor. This product constitutes the weight W_j of cube number j:

$$W_j = M^i_{nn'}(\mathbf{k}, \mathbf{k}')G_{nn'}(\mathbf{k}, \mathbf{k}')\frac{A(\delta)}{(2\pi)^3|\nabla E_n(\mathbf{k})|}. \tag{2.28}$$

To obtain the scattering rate $S^i(\mathbf{k})$ from this, one can use the formula

$$S^i(\mathbf{k}) = \sum_{j=1}^{\text{maxbox}} W_j, \tag{2.29}$$

where maxbox is the number of cubes found to conserve energy and momentum. This has been done in advance and the rate is stored for $\mathbf{k}$-vectors corresponding to the black dots in Fig. 2.6.

The process of finding a suitable final state after scattering follows this same procedure, but naturally without doing the rate sum. After finding the cubes with energies containing the final value, one of them is chosen at random. If this cube is accepted by the rejection technique, as explained below, the final $\mathbf{k}'$-state is now to be found within this cube.

Clearly, the amount of calculations involved here is quite formidable. All of the final cubes must be supplied with data concerning the shape of the bands, taken from a table. Based on the data, additional evaluations are necessary for each cube.

It would have been a huge task to precalculate and store the weights W_j for all possible final cubes for each point in the mesh and a given scattering mechanism i. Not only would the size of the table have to be quite formidable, but there are also other objections. Such a table has to be based on the weight of a transition from the wave vector marked by black dots in Fig. 2.6 to an energy conserving wave vector $\mathbf{k}$ in the final cube. For an arbitrary $\mathbf{k}$ in the initial cube, however, with an energy deviating somewhat from the energy at the mesh point, there may exist no energy conserving final states for it to enter in the cube pointed out by the weight for a transition from the wave vector at the mesh point. But the neighbours to this final cube could contain all the more available states. Such a rigidly divided $\mathbf{k}$-space has its drawbacks; the weights become critically dependent on the position within the initial cell, and therefore they should not be precalculated and stored since they would not be useful without a careful total restructuring of the overall MC algorithm.

The scattering rate $S^i(\mathbf{k})$, which consists of a sum of weights, is a smoothly varying quantity. Therefore, it should be precalculated and stored if possible.

The technique used for choosing the scattering mechanism, Fig. 2.2, could also be used for choosing the final cube, but the discrete version of the rejection technique is preferred because it is faster in most situations, see Fig. 2.7.

In the discrete rejection technique, two random numbers are generated; r is a real number in the interval $[0, 1]$ and j is an integer in the interval $[1, \text{maxbox}]$. A fixed number C is chosen, exceeding all individual cube weights. If the point (j, rC) described by the two random numbers lies on a particular column area of the histogram, the corresponding cube is chosen. If the point (j, rC) lies above the histogram area, a new pair of random numbers is generated until the suggested pair again lies on the histogram column area corresponding to a

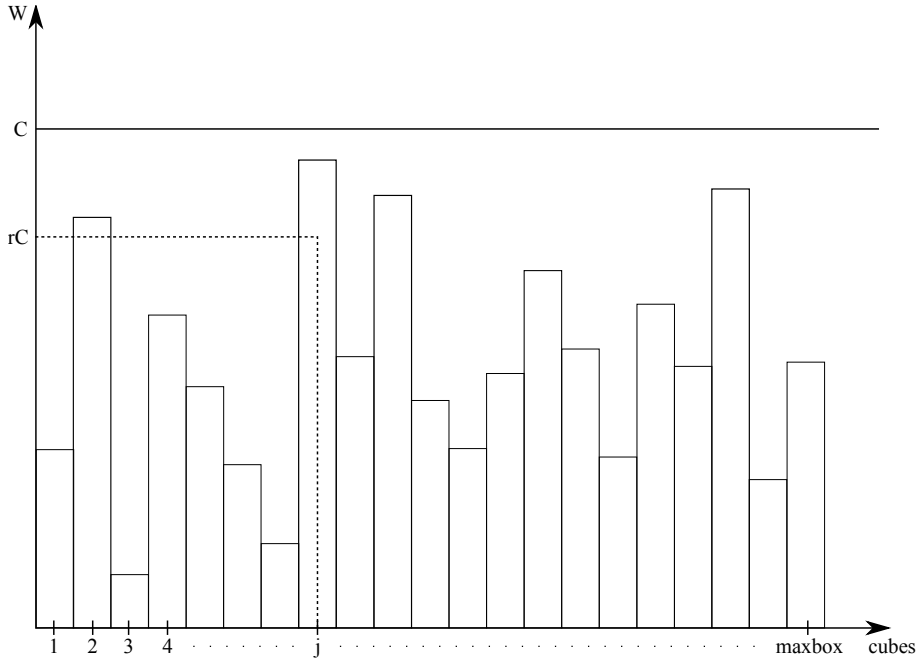


Figure 2.7: Discrete rejection technique. Cube number j will be chosen.

particular cube. Since all suggestions of random number pairs are uniformly distributed on the total rectangle area $([1, \text{maxbox}], [0, C])$, each cube has a chance of being selected which is in accordance with its column area in the histogram, and thereby with its weight. Figure 2.8 shows an example of what the distribution of weights might look like. In this case the weights are calculated for non-polar phonon emission, intraband in the light hole. The **k**-vector of the particle used in this calculation was set to be $\mathbf{k} = (5.0, 5.0, 5.0) \cdot 10^8\,\text{m}^{-1}$, which corresponds to an energy of approximately 0.12 eV. We will take a closer look at different weights in section 3.6, where we will also see that the distribution of weights does not always look like figure 2.8.

Calculation of individual weights during the simulation is a slow process, and even if the sum of the weights is known (equal to the rate), one would generally end up evaluating more weights if we used the same technique as for choosing the scattering mechanism.

If we consider carrier-carrier scattering, or evaluating the screening parameter for ionized impurity scattering self-consistently, it is not tempting to store the rates in a table. The amount of information provided would be unmanageable, since the carrier density would constitute one of the input parameters. Therefore, scattering rates for these special processes have to be evaluated during the simulation, by adding the individual weights for the given initial **k**. In principle, all final cubes must be located to find the rate, but the weights so obtained are

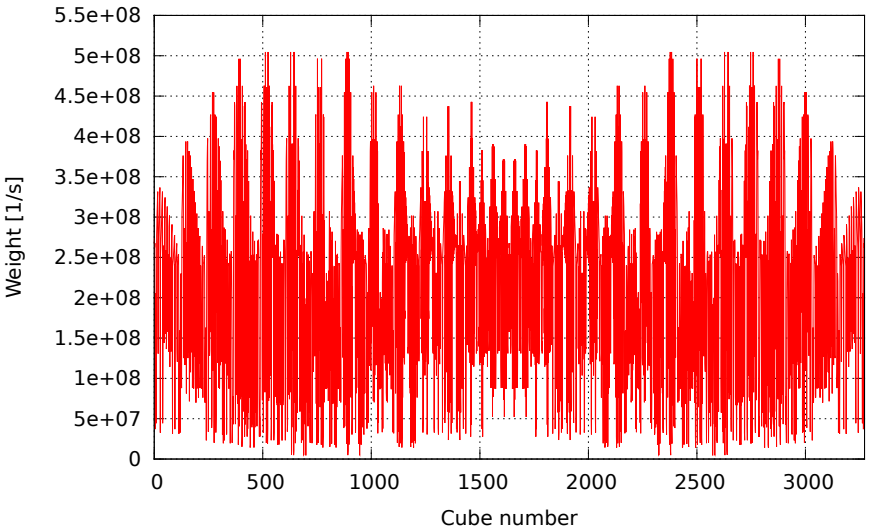


Figure 2.8: Weights for non-polar phonon emission, LH -> LH. Input energy of approximately 0.12 eV, with **k**-vector along the [111] direction.

immediately re-used to find the new cube after scattering. No interpolation is needed, because the rate has been evaluated for the actual **k** in the initial cube, not from the **k** at a mesh point as was the case for the precalculated rates.

For $E(\mathbf{k})$ and the group velocity v an interpolation scheme is used. In addition to these quantities, an MC program must be able to invert the $E(\mathbf{k})$ relation quickly and efficiently, because the location of $\mathbf{k}'$ in **k**-space is needed very often during the evaluation of the transition weights mentioned above. We have actually now seen how this is done when the band structure is given in a table. When the searching procedure for final cubes containing a given energy is performed, this corresponds to an inversion of the dispersion relation $E(\mathbf{k})$. In fact, all **k**'s on the constant energy surface display the $\mathbf{k}(E)$ relation.

To locate a final wave vector at an energy conserving or a constant energy surface, a small extension of the wave vector at the mesh point (black dots) next to the small final cube is made, until the desired final energy is reached. In the work of Fischetti and Laux, this extension was parallel to the principal directions of the final cube; edges, side diagonals, and the cube diagonal. Which one was chosen was determined by a random number with uniform probability. This reflects the fact that all states having the correct final energy within the final cube will have approximately the same probability of being chosen. As previously mentioned, this extension now instead occurs in the direction of the energy gradient.

The first MC programs that utilized a discrete description of the bands [16, 17,

18] did not make this extension at all. Instead, the central wave vector in the final cube was chosen. The energy associated with this wave vector was found by Fischetti and Laux at IBM to deviate from the correct one by more than 10 meV for electrons in the conduction band, depending on the shape of the bands in the neighbourhood.

Very often, in discrete versions of the MC technique, the energy region near the band edge is described analytically. This enhances speed and reduces the size of look-up tables. Also, the evaluation of the small energy exchanged in acoustic scattering events needs a high precision at the low energies and temperatures this mechanism is important, and such a high precision is difficult to achieve with a discrete description. However, at the moment there are no such analytical approximations in place for low energies. The use of a fine mesh near the Brillouin zone center provides adequate results for now.

In general, large look-up tables would enhance the execution speed of a program. But sooner or later the time consuming search in tables and the amount of interpolations needed would catch up with such an approach, eventually causing a slow down of the program. Fischetti and Laux claims there is a problem of accuracy even with 41 000 **k**-points precalculated in the first Brillouin zone (1BZ). They found that we cannot expect to reach an accuracy better than 4 meV on average under the inversion of E(k). Therefore, the stored tables must be larger than this for many purposes. In a cube mesh this corresponds to about 35 mesh points in each direction. Section 3.6 will discuss how well the current implementation works in more depth. As the full-band model must be considered experimental at this stage, actual details of the implementation are subject to change in future versions.

CHAPTER 3

PROGRAM DETAILS

A number of new functions have been added to the program. This chapter details the functionality and implementation of said functions, as well as a general overview of the entire MC program. The chapter opens with the general overview in section 3.1. Section 3.2 talks briefly about the integration of Halvorsen's band structure and rate calculation programs. Details on the interpolation routine used to interpolate the band structure and rates can be found in section 3.3. The following section, 3.4, discusses the necessary changes to make the interpolation routine work properly for the rates. Handling of the free particle flight is considered in section 3.5. Finally, section 3.6 discusses the implementation of the algorithms for finding a suitable **k**-vector after a scattering event.

The program is written in Fortran 90 and compiled using the Intel Composer XE 2011 compiler.

3.1 Program description

Descriptions of the earlier versions of the program can be found in the master's theses of Norum [1], Olsen [2], Skåring [3] and Kirkemo [4]. In depth explanations of routines not implemented in this work are not included here, but can be found in these earlier theses.

In figure 3.1 we see a simplified flowchart of the main tasks in a simulation run. Let us explain these tasks in order.

Upon running the program, you are presented with a menu that looks something like this:

```
Main Menu
=========
Make your selection
-------------------
1) Simulation parameters
2) Scattering mechanisms
3) Pauli, Hotphonons, Poisson & Halvorsen
4) Device simulation
5) Save
6) Load
7) KPBAND Initialization
8) Run KPBAND
9) Calculate scattering rate (time consuming)
0) Continue
```

The new additions to the menu in this version of the program are points 7, 8 and 9. With the integration of Halvorsen's routines in the MC program, all the necessary parameters for calculating band structures and scattering rates can now be set directly from this menu, rather than going via Halvorsen's standalone programs.

After setting all parameters, you can choose to calculate bands or scattering rates, or to start the main MC simulation. Bands and scattering rates are stored to files in the same directory as the program is run from. Scattering rates take quite some time to calculate, so they will typically not be calculated every time the program is run. Bands are relatively quick to calculate (for reasonable mesh sizes), but it is important that bands and rates have been calculated using the same parameters. Altering the band structure without recalculating the rates will of course lead to incorrect results.

When the MC simulation is started, the band and rate data is read from file and processed. The files containing the data are unformatted binary files, which is an efficient, but complicated, way of storing data. It uses the very old concept of record lengths to denote the memory needed to store different data types. This record length is the size of the data in bytes. What this means is that the file containing the energy mesh, only a single number for every mesh point, will have a record length equal to the byte size of a single number in the given precision of the program (typically 8). Gradients contain three numbers for every mesh point, so the record length for the gradient files is three times that of the energy files. For second derivatives it is six times, and for eigenvalues it is MS$\cdot$2$\cdot$2, where MS is the number of eigenvalues (6, 8 or 14; the $\mathbf{k}\cdot\mathbf{p}$ model size), times two because they are complex numbers, times two again because of two spin states per band. A very basic program for reading some of this data and printing it to plaintext which can be plotted by your plotting software of choice had already been written, and served as a starting point for the construction of the read routines. This routine was incrementally improved in the early days of this work. Falch then wrote an improved version which was later modified and

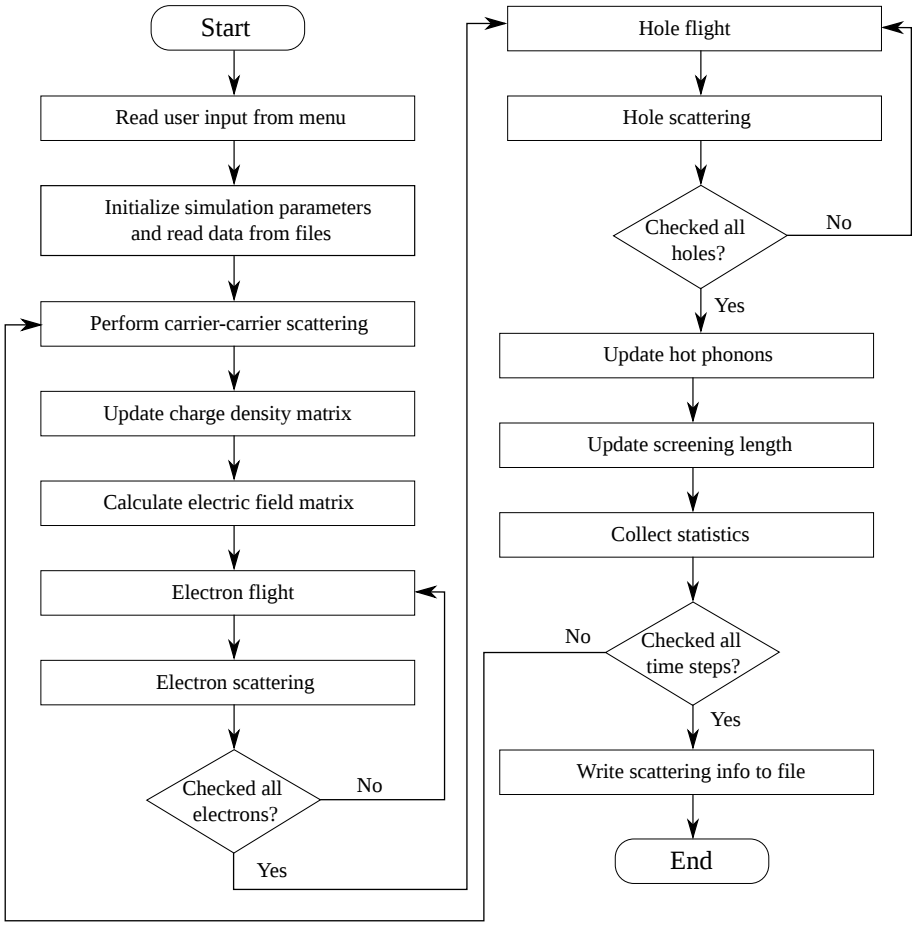


Figure 3.1: Simplified flowchart of the MC program.

integrated with the MC program. The maximum and minimum values of the energy gradient are also found while reading in the data.

Halvorsen stored the second derivatives in the order xx, yy, zz, xy, xz, yz. This suited the needs of his program, but not so much now. In order to write the interpolation routines in the most efficient way, the xz and yz second derivatives are swapped during the read routine.

Next up is the process of initializing the positions and momenta of the electrons and holes. Carrier position is only important for device simulations, so for bulk simulations we skip this step. However, initializing the momenta is of course important in all types of simulations. They can be initialized in either a Gaussian, optical or experimental distribution [1].

In earlier versions of the program, this is the point where the analytical scatter-

ing rates were calculated and stored in tables. This functionality is still somewhat present, as not all of the analytical rates are available from Halvorsen's program. The scattering mechanisms in the program at this time are carrier-carrier, coupled modes, polar optical, non-polar optical, acoustic, ionized impurity, plasmon, intervalley, alloy and impact ionization scattering. Of these, carrier-carrier, coupled modes, plasmon, intervalley, alloy and impact ionization are only available in analytical form. In addition, the Halvorsen ionized impurity rates are too erratic to use in the program at the moment, so if ionized impurity scattering is enabled the analytical rates will be used. When these rates are fixed, it is a simple matter to put them back into use.

With the 6×6 $\mathbf{k} \cdot \mathbf{p}$-method, only the heavy hole, light hole and split-off bands were found numerically, and interband hole scattering was only considered between the heavy and light hole bands during an MC simulation run. With the extension to the 14×14 $\mathbf{k} \cdot \mathbf{p}$-method, we could also find the first four conduction bands. However, only the first conduction band is to be used in simulations.

Then we get into the main simulation loop. The first thing to happen is the carrier-carrier scattering. The algorithm is limited to one scattering of each kind (electron-electron, hole-hole, electron-hole) per time step, requiring sufficiently short time steps. Carrier-carrier scattering is typically not enabled in bulk simulations.

Updating the charge density matrix is done using the cloud-in-cell algorithm. Calculating the electric field matrix is done by solving the Poisson equation using a successive overrelaxation method every n'th time step, where n is a freely chosen integer. Both of these methods are explained in detail by Kirkemo [4]. These options are also typically disabled for bulk simulations.

Electron flight and electron scattering are each handled in separate routines. In the `flight` routine, the $\mathbf{k}$-vector and position in space are updated according to the total electric field at that time. Equations of motion for particles in $\mathbf{k}$-space can be found in any introductory solid state book, e.g. Kittel [19]. The `scatter` routine determines whether or not a scattering event takes place in this time step and if so, it then decides which type of scattering occurs.

For holes the process is the same as electrons, but a number of changes have been made to both routines. In the flight routine it was necessary to add calculation of the sum of all scattering rates at the start of the free flight, in order to implement something closer to the Yorston method for self-scattering described earlier. Details of this will be discussed in section 3.5. The scattering routine which used to calculate scattering rates for the given $\mathbf{k}$-vector using the tabulated analytical rates now uses the discrete rates and the interpolation methods described in section 3.4. Like for electrons, it is now decided whether or not scattering occurs in this time step, and the specific mechanism is chosen. The details of how this is done are, however, not quite the same. This is where the pseudo-Yorston method is implemented, which reduces the number of self-scatterings for holes tremendously. The way in which the new $\mathbf{k}$-vector is found has then

also been changed to the new Halvorsen compatible routine. This is done via the method described in section 2.3.2, and the details of the implementation will be looked at in section 3.6.

After running through all electrons and holes and performing scattering on the chosen particles, it is time to collect statistics for this time step. Depending on the type of simulation, various information is stored so that it later can be printed to file.

Finally, it is time to update parameters for the next time step. This means updating the hot phonon information as well as the screening length, if either of them are enabled.

After finishing all time steps, the saved data from earlier is written to a number of files. For bulk simulations, the most interesting is the data on total number of scatterings of each type, as well as data on scattering of individual particles, energy consistency throughout the run (check for artificial energy increase/decrease) and momentum distribution.

3.2 Band structure and rate calculation

The band structure is calculated by the $\mathbf{k} \cdot \mathbf{p}$-method. The 6×6 version of this procedure is detailed by Halvorsen [9], who implemented a working version of this algorithm for his thesis. Initially, updated versions of his programs were integrated with the MC program, giving the program the capability to calculate and use the discrete band structure and scattering rates in the MC simulation.

Calculation of rates is done as explained in section 2.3.2. This method is also more thoroughly explained by Halvorsen.

Improvements to Halvorsen's work have been implemented by Bjørnar Karlsen [12]. He has extended the $\mathbf{k} \cdot \mathbf{p}$-model from 6×6 to 8×8 and 14×14, thus giving us the structure of the first four conduction bands as well as the valence bands. In this work his changes have been implemented in the main MC program, enabling the use of the updated models in band and rate calculations.

In the following sections there are figures of this band structure. They are merely meant to illustrate the accuracy of the interpolation routines, and as such the band parameters used in the calculations are not that important. For the 6×6 method, the parameters are the ones given by Halvorsen for GaAs, with a few minor changes. This is because the ultimate goal was always to incorporate the 14×14 method into the MCT MC program, with the 6×6 method simply being the first step on the way there. Parameters for the two methods are different, so time was not spent on finding correct MCT parameters for the 6×6 method when it was going to be replaced anyway.

3.3 Band structure interpolation

With the band structure now consisting of data given only in discrete mesh points, the need to interpolate between the mesh points efficiently and accurately arises. For that purpose, an interpolation scheme like the one used by Fischetti and Laux [15] was implemented.

Energies, derivatives and second derivatives are all stored for two given 3D-meshes for each band, ν; a fine mesh around $\mathbf{k} = 0$ and a coarse mesh that covers the first Brillouin zone. For a given $\mathbf{k}$-vector, a search is performed over the mesh in order to find the eight corners of the cube in which this vector belongs, $\{\mathbf{k}_\lambda\}(\lambda = 1, 2, ..., 8)$. Unlike the rate calculations and final cube selections, this algorithm considers the mesh points to make up corners of a cube, not the midpoints. The energy can then be quadratically expanded around each corner as follows:

$$E_{\nu,\lambda}(\mathbf{k}) = E_\nu(\mathbf{k}_\lambda) + \frac{\partial E_\nu(\mathbf{k}_\lambda)}{\partial k_i}(k_i - k_{i,\lambda}) + \frac{1}{2}\frac{\partial^2 E_\nu(\mathbf{k}_\lambda)}{\partial k_i \partial k_j}(k_i - k_{i,\lambda})(k_j - k_{j,\lambda}), \quad (3.1)$$

where sums over identical indices must be performed. In order to find the energy at the given $\mathbf{k}$, the weighted sum of the contributions from each corner must be calculated,

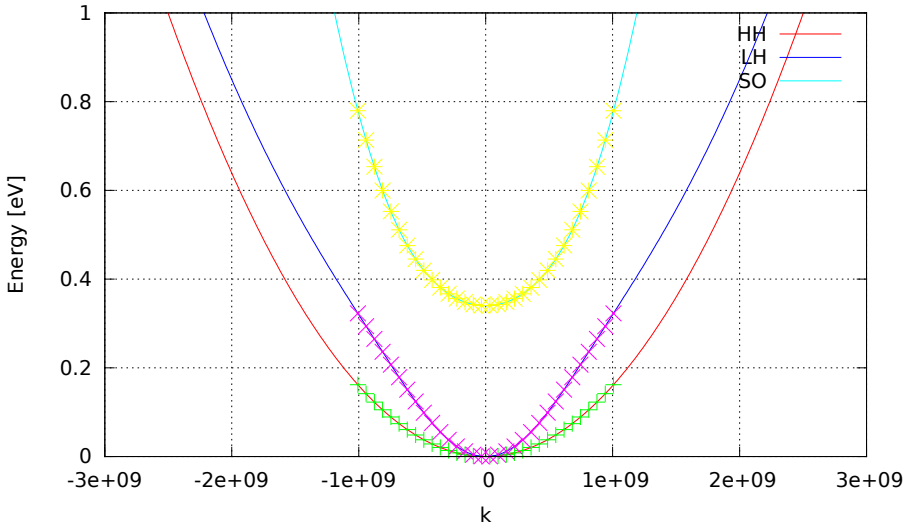
$$E_\nu(\mathbf{k}) = \sum_{\lambda=1}^{8} W_\lambda E_{\nu,\lambda}(\mathbf{k}), \quad (3.2)$$

with the weights given by

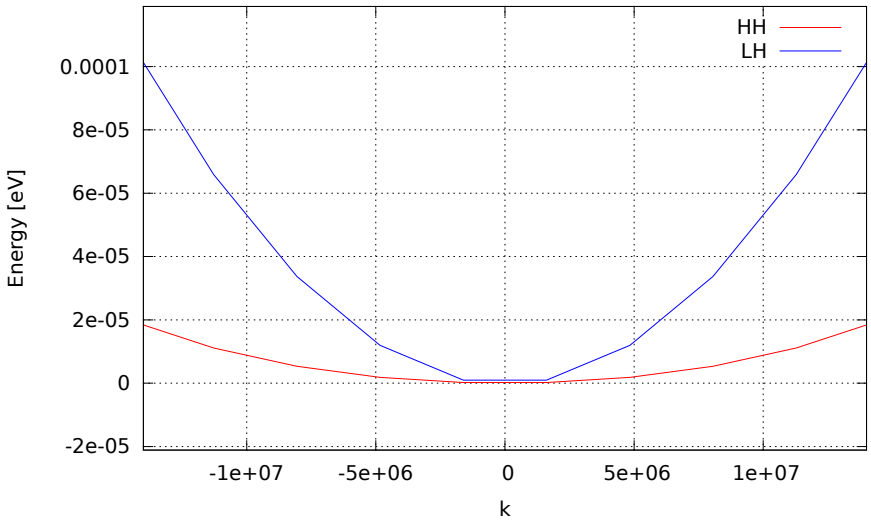
$$W_\lambda = \left(1 - \frac{|k_x - k_{x,\lambda}|}{L}\right)\left(1 - \frac{|k_y - k_{y,\lambda}|}{L}\right)\left(1 - \frac{|k_z - k_{z,\lambda}|}{L}\right), \quad (3.3)$$

where L is the side length of the cube, i.e. the distance between mesh points. This method is exact for parabolic bands, and works well for our purposes.

Fig. 3.2a shows the result of the interpolation scheme applied to the valence bands near the Brillouin zone center, using 500 interpolated points. It may appear that the bands have small ripples, but this is only an artifact of the displayed resolution. The original vector graphics figure has no such ripples, the bands are smooth. The markings shown in the figure denote the energies in the fine mesh points, with the bands calculated using Halvorsen's 6×6 $\mathbf{k} \cdot \mathbf{p}$ method. A zoomed in version plotted using 2500 points, Fig. 3.2b, shows that good accuracy is achieved also near $\mathbf{k} = 0$. It is important to note that when running the full MC program, the interpolation routine is called every time $E(\mathbf{k})$ is needed. That means that in practice we get even better accuracy than can be seen here without plotting a ridiculous number of points. A final figure showing the bandstructure in 3D is Fig. 3.3. It shows a cross section of the band structure as a function of k_x and k_y, while k_z is an arbitrary given value. The area shown here is near the zone center. This plot is not made using interpolated data, but rather only the data in the mesh points.



(a) 500 points.



(b) 2500 points.

Figure 3.2: Valence bands drawn in the [111] direction using interpolation. k is in units of 1/m.

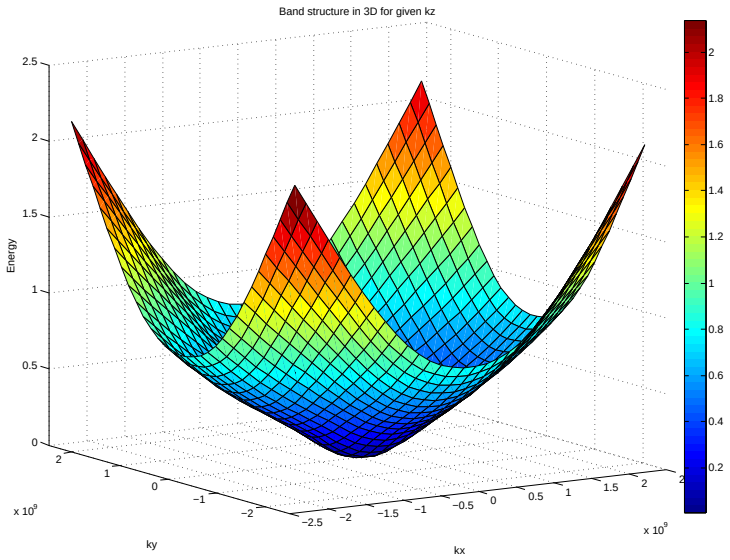
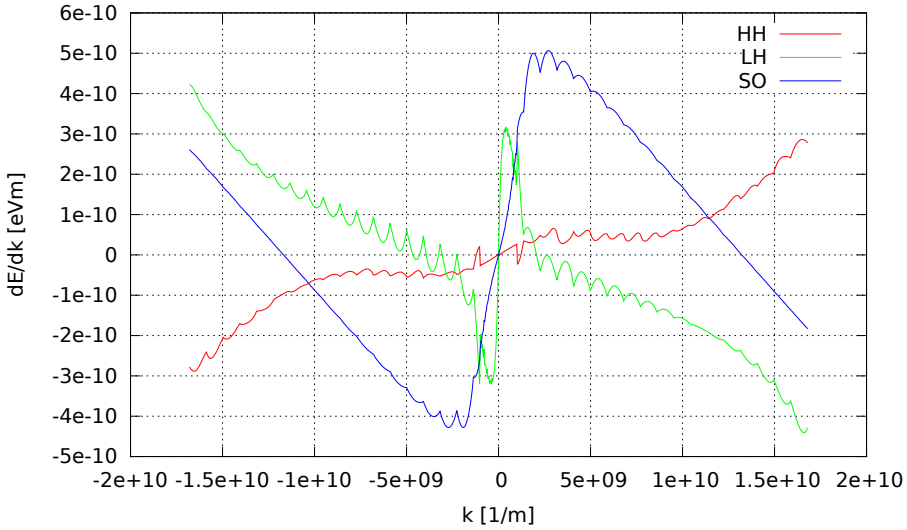


Figure 3.3: Light hole band in 3D for a given k_z. k is in 1/m, energy in eV.

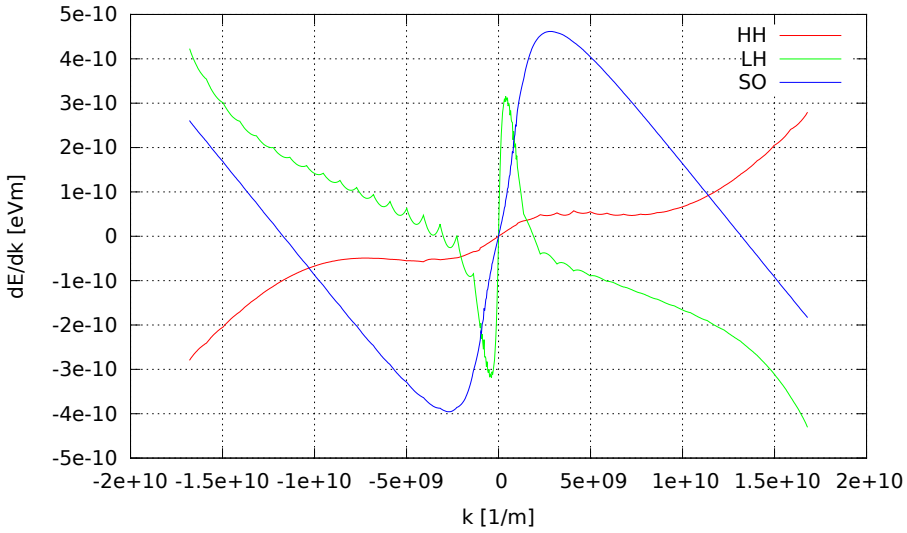
A very similar function had to be written for interpolating the derivatives of the band structure. Some modifications had to be made to the routine, seeing as the third derivatives of the band structure are not given. This modification was simply to neglect the second derivative term in equation (3.1) and insert the second derivatives in place of the first derivatives, while the first derivatives take the place of the energy.

The derivatives are used in the functions calculating particle velocity, making it necessary to know derivatives at arbitrary points in the mesh, not just at the mesh points. In terms of number of code lines, this is a very small, but important, part of the program. Particle velocities calculated using the derivatives of the analytical band structure while the rest of the program operates with the full-band structure would be wrong. For the 6×6 method the derivatives look smooth and nice. To illustrate a problem with the 14×14 method, figures showing how the interpolation of derivatives works are plotted with data from the 14×14 method. Figures 3.4a and 3.4b illustrate this. The first one is plotted using the method described in the previous paragraph. Immediately one can see problems with these derivatives. There is a rippling effect happening with the interpolation in between the mesh points. Fig. 3.4b is plotted without using the second derivatives, and so the rippling is much less severe, but it is still quite prominent.

Reasons for this are not entirely clear, but Karlsen and Brudevoll discussed the



(a) Second derivatives.



(b) No second derivatives.

Figure 3.4: X-derivatives of hole bands in the [111] direction using 1000 interpolated points.

issue and speculated that it is caused by the poor precision of the eigenvalue solver currently used in the band calculating routine `KPBAND`. This solver is the LAPACK routine `zheev` [20]. Karlsen also said that the solver was not performing optimally when calculating the band structure. This will then have an effect on the calculation of the first and second derivatives, which depend on values returned from this solver. This leads to the second derivatives being wrong, causing a severe increase in this ripple effect. Suspecting that the ripples are caused by problems outside of the scope of this thesis, the current implementation is left in. Disabling the second derivatives is a temporary band-aid on the problem until a solution can be found. The fact that the Fischetti and Laux interpolation method works so well on the 6×6 model also hints that the problem lies elsewhere.

3.4 Rate interpolation

Like the band structure data, the rates are only given for a discrete number of mesh points. Thus we once again need an efficient and accurate interpolation scheme. However, unlike for the band structure, we do not have the analytically calculated first and second derivatives of the rates. This means that we cannot directly apply the same interpolation scheme to this data. A few other simple interpolation schemes were tried on the data, namely simple linear interpolation and some sort of cubic spline interpolation. Some of this work was done in Fortran with self-made methods, while others were simply tested using built-in MATLAB routines. The results were quickly found to be unsatisfactory, and these methods were discarded. Therefore, in order to avoid introducing an entirely different interpolation scheme, we are left with two options: either we can use the same scheme without taking into account the first and second derivatives, or we can try to estimate the derivatives numerically from the given data.

A big concern with the numerical differentiation was whether or not the results would be accurate enough to be usable. The decision was made to try it out, beginning with one dimension before expanding into the real three-dimensional world. Routines for numerically differentiating the scattering rates were then implemented. This was done using various simple finite difference approximations, the details of which can be found in any mathematical textbook on numerical methods. Rates are in reality dependent on k_x, k_y and k_z, but since differentiation is a linear operation (and for simplicity) we denote the rates simply as $f(x)$ in the following equations. A simple five-point stencil was used for the central points of the mesh:

$$f_c'(x) \approx \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h}, \tag{3.4}$$

$$f_c''(x) \approx \frac{-f(x+2h) + 16f(x+h) - 30f(x) + 16f(x-h) - f(x-2h)}{12h^2}, \tag{3.5}$$

where h is the distance between mesh points. For the mesh points nearest to the edge points, we resort to a central two-point formula:

$$f'_{\mathrm{ne}}(x) \approx \frac{f(x+h) - f(x-h)}{2h}. \tag{3.6}$$

The second derivatives in these points were calculated in two different ways. For the xy-, yz- and xz-derivatives the same method was simply applied a second time, with $f_{\mathrm{ne}}(x) \to f'_{\mathrm{ne}}(x)$. The xx-, yy- and zz-derivatives were found by a three-point stencil:

$$f''_{\mathrm{ne}}(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{3.7}$$

Finally, for the edge points, the simplest two-point estimation of the derivative was used:

$$f'_{\mathrm{e}}(x) \approx \frac{f(x+h) - f(x)}{h}. \tag{3.8}$$

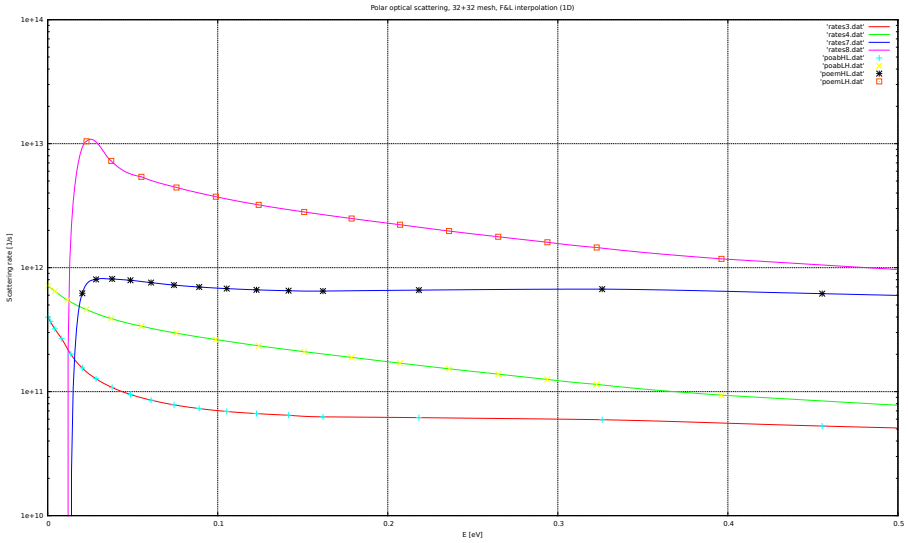
As with the near-edge points, the method is applied once more in order to find the second derivatives. The edge points will typically be at the edge of the first Brillouin zone, an area where high accuracy is not as critical as it is closer to the center.

Figures 3.5, 3.6 and 3.7 compare the Fischetti and Laux method (including the derivatives) and simple linear interpolation for some of the scattering mechanisms in one dimension. These figures are plotted with energy along the x-axis, as that made it easier to compare with the analytical rates already in use by the program. It is clear that in these cases the F&L method gives quite good results; certainly better than the linear interpolation. However, that was not always the case when moving into three dimensions, as the inaccuracies of the numerical derivatives caused more prominent rippling.

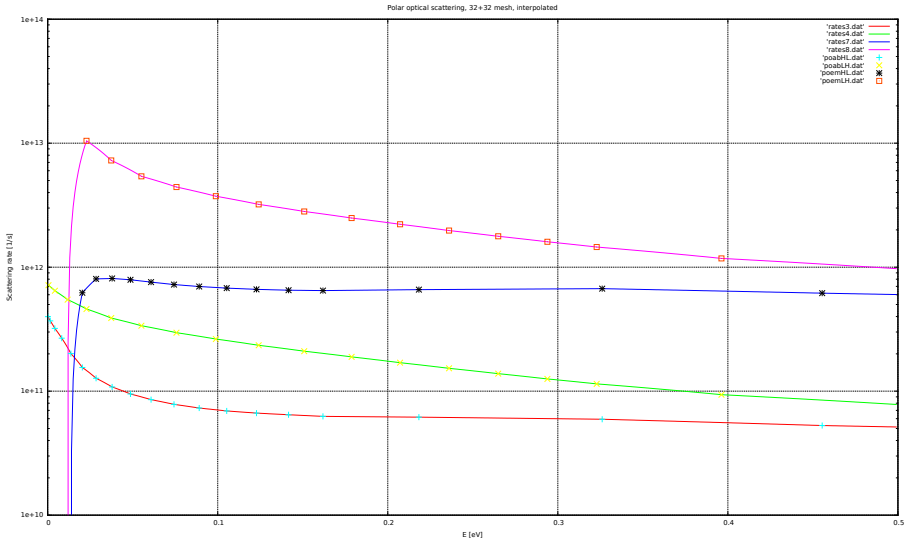
A third alternative for interpolating was then implemented: a smoothing function which averages the rate over a cube. The interpolation routine (without taking the derivatives into account) was then averaged over the given **k**-vector and eight points chosen to be corners of a surrounding cube. An adequate size for this cube was found through trial and error. All routines are programmed to automatically determine whether to use the coarse or the fine mesh when choosing mesh points used in the interpolation.

Comparisons of rates calculated by the various interpolation routines for the three-dimensional case can be seen in Fig. 3.8. These figures only show the rates for **k**-values close to the center, where the derivatives are relatively well behaved. Nonetheless, it can be seen in some of the figures that the rates interpolated with non-zero derivatives begin to exhibit a rippling pattern, in that the rates oscillate away from the real values in between mesh points. These ripples only get worse for larger **k**-values.

An important thing to explain about these figures is the ugly spike present on the negative side in all figures for the rates calculated while taking the derivatives

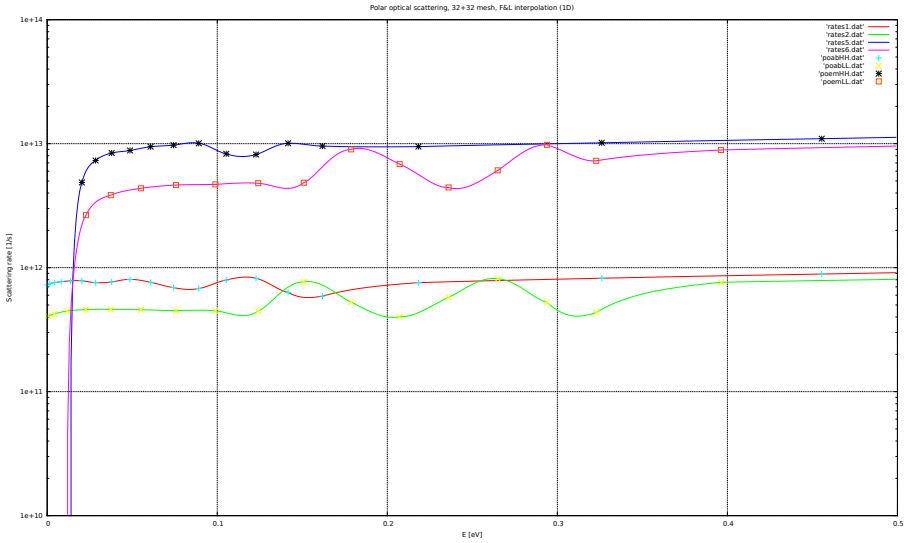


(a) F&L

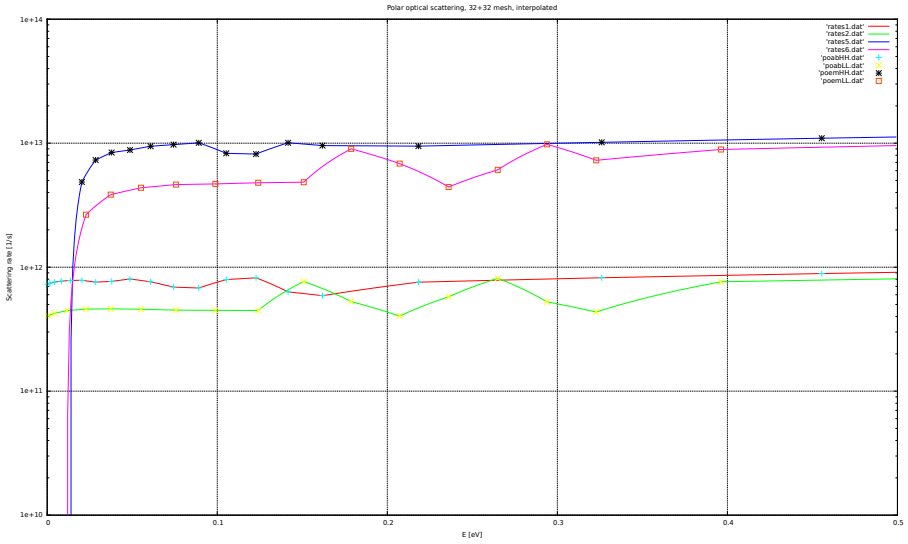


(b) Linear

Figure 3.5: Polar optical interband phonon scattering rates in one dimension.

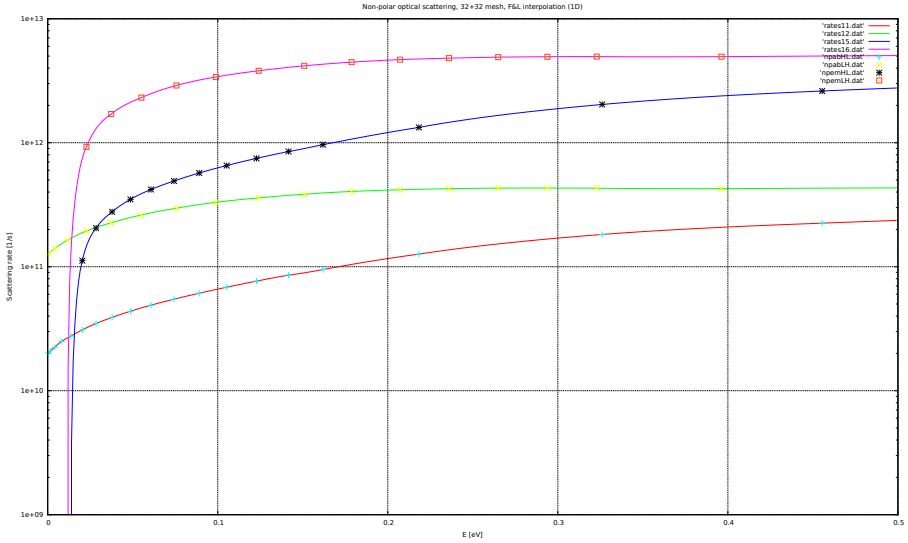


(a) F&L

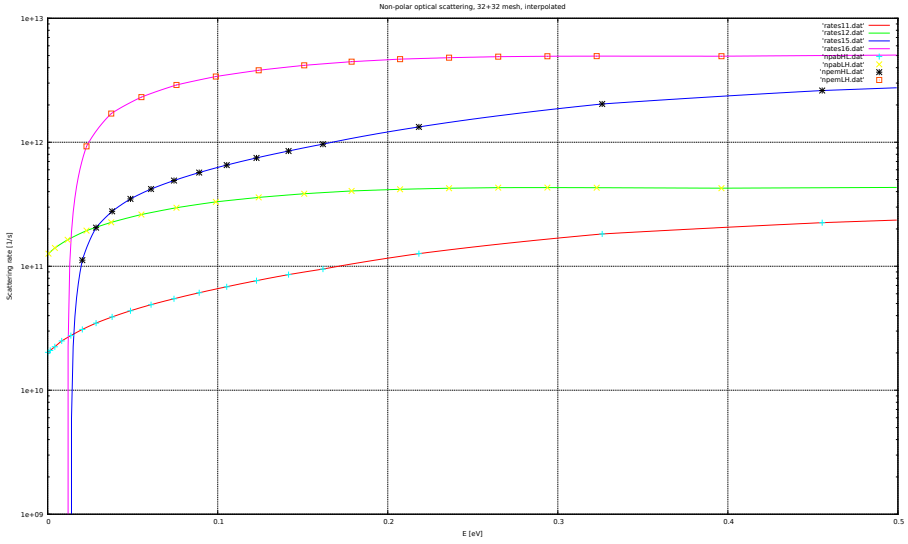


(b) Linear

Figure 3.6: Polar optical intraband phonon scattering rates in one dimension.



(a) F&L



(b) Linear

Figure 3.7: Non-polar optical interband phonon scattering rates in one dimension.

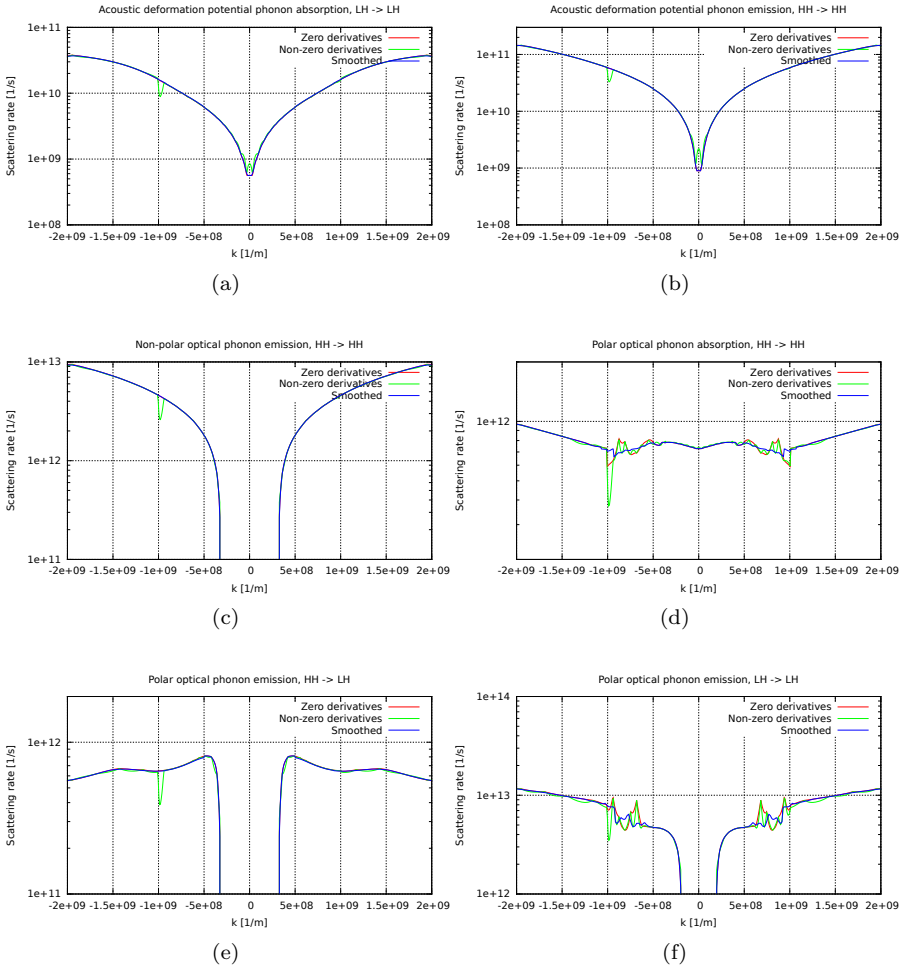


Figure 3.8: Scattering rates in three dimensions along the [111]-direction.

into account. This is most likely a bug with the way the differentiation routines handle the transition from the coarse to the fine mesh from the negative side. Since the derivatives introduced unacceptable errors to the rates, and as such will not be used in the program, the bug was deemed minor at this time.

In most figures the rate plotted using zero derivatives can hardly be seen, as it lies right on top of the smoothed rates. There is, however, a noticeable difference in some of the less well-behaved rates. Examples of this are the polar optical rates in figures 3.8d and 3.8f. The smoothing is by no means perfect, but it is clearly better than the alternatives.

Another minor detail that needs to be taken into account is the low energy cutoff

for certain emission rates. At energies below this cutoff these scattering rates are zero. This applies to polar optical and non-polar optical emission rates. The cutoff is equal to the optical phonon energy, $\hbar\omega_0$. This was handled by a straight forward energy interpolation to see if the energy at the given $\mathbf{k}$ was below this value and setting the rate to zero if that was found to be the case. This is not the optimal way of handling the problem, as it causes a discontinuity in the rates where it suddenly drops to zero. The best way of handling it would be to find exactly where in $\mathbf{k}$-space the cutoff is and use this point as the edge of the cube. Then the rate would smoothly decrease until zero exactly at the cutoff. However, this will not make much of a difference, so it was handled in the easier way.

3.5 Free flight

The Yorston method of self-scattering is great for large time steps. If the time steps are large, there is a possibility that the scattering rate will differ significantly between the start and the end of the time step. If that is the case, scattering rates can be inaccurate at the scattering time. This problem is what the Yorston method is meant to counter. In a proper implementation of this method the scattering check can occur at any time in the interval, and the scattering rate at that time is taken into account when determining if a scattering actually occurs. However, that is tricky to implement in our program without a rather substantial restructuring of the relevant routines. Some work has been done on this, but only a pseudo-Yorston method has been implemented for now. What this means is discussed below.

Each time step is currently handled by calling two separate routines, called `flight` and `scatter`, as explained in section 3.1. The program was originally designed for use with short time steps, so there is no mechanism implemented for checking if scattering occurs during the time step. Before this work, the `flight` routine would update the $\mathbf{k}$-vector according to ordinary laws of motion for a particle in an electric field,

$$\mathbf{k_f} = \mathbf{k_i} + \frac{e\mathbf{E}}{\hbar}\Delta t, \tag{3.9}$$

and the `scatter` routine would then evaluate the relevant parameters at the end of the time step and determine whether or not a scattering event occurs. This did not take directly into account the possiblity of varying scattering rates over the interval. The problem of changing rates was instead handled by a very generous estimate of the maximum value the sum of all scattering rates could take. This led to a huge number of self-scatterings, and was not a very efficient way to handle the problem.

The pseudo-Yorston method that has been implemented to replace this provides a more accurate estimate of the maximum total scattering rate encountered over

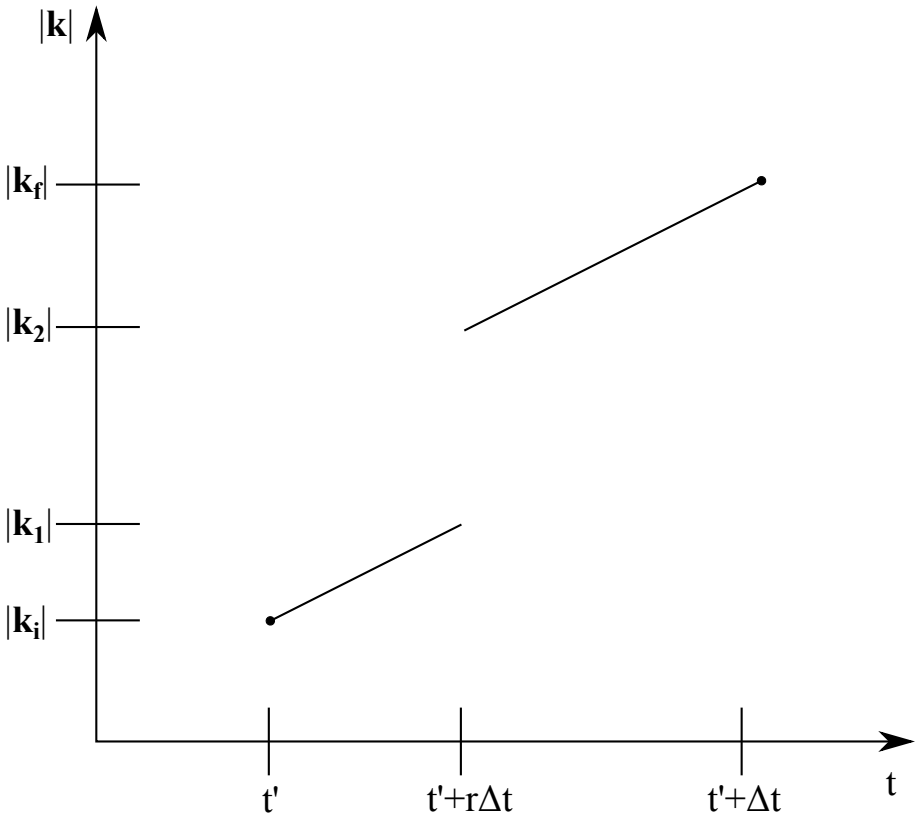


Figure 3.9: Effects of Yorston method on $|\mathbf{k}|$ illustrated.

the interval, and it works as follows: First the rate interpolation is used to find an accurate total rate at the beginning of the interval and this is compared with the rate at the end of the interval. If the total rate has gone down, the rate sum at the start of the interval is used as the maximum value. If it has gone up, the final sum is used instead. Since scatterings only occur at the end of an interval, this means that in practice self-scatterings can only happen if the rate sum has gone down over the course of a time step. It should be restated here that when we say that the rate varies over a time step we are not talking directly about time-dependent rates, but rather $\mathbf{k}$-dependent rates with a time-dependent $\mathbf{k}$, $S(\mathbf{k}(t))$.

This implementation retains the problem of not allowing scatterings to happen at any other time than the end of a time step. For the small time steps most commonly used, this is not a big problem. However, it is something that should definitely be looked at for future work. One way of handling it could be to simply use a random number between zero and one to determine how far into the time step the scattering check is performed, perform the check at that

time, execute eventual scatterings and then propagate the particle for the rest of the time step. Figure 3.9 illustrates this process. For simplicity only the absolute value of the **k**-vector is shown. A particle starts the time step at time t' with momentum $\mathbf{k_i}$. A random number, r, between 0 and 1 is then drawn and the particle is propagated until $t = t' + r\Delta t$, where the particle now has momentum $\mathbf{k_1}$. In the case shown in the figure, a scattering then occurs. This scattering brings the momentum up to $\mathbf{k_2}$, and the particle is then propagated until $t = t' + r\Delta t + (1 - r)\Delta t = t' + \Delta t$, which is the end of the time step. The particle now has momentum $\mathbf{k_f}$, and the time step is over. With the current method the jump happens at the end of the interval every time, leaving only a discrete number of possible scattering times. The method detailed here would make it possible for scattering to occur at an arbitrary time within each interval, meaning a more realistic simulation and allowing for longer time steps.

3.6 Finding final states after scattering

This has been discussed quite thoroughly in sections 2.1 and 2.3, so we will not repeat the details here. Instead, we will look at specifics regarding the implementation and how well the algorithm works. Finding the final states is handled in a routine called `Etok`. This routine is given an energy, band after scattering, scattering mechanism and a particle number, and returns the new **k**-vector. The particle number is the position in the arrays containing position in **k**-space of the particle being scattered.

A search is then performed throughout the entire mesh, both coarse and fine, to find cubes which contain the given energy between the minimum and maximum energies found earlier. The maximum and minimum energies for each cube can be found in several ways, but at the moment a brute force technique is in use. After reading in the band structure data from files at the start of the program, the interpolation routine is applied to N^3 points in the mesh, where N is chosen as a compromise between speed and accuracy. For each iteration the program checks which cube the given **k**-vector is in and whether the returned value is higher or lower than the previous values for maximum and minimum energies, then updates accordingly. In the rate calculations, Halvorsen uses a simple extrapolation from the gradient in the center of the cube for this. This method was tested, but found to provide too low accuracy at times, especially for low energies near the zone center. At the moment $N = 300$, which provides good enough accuracy and completes the process in less than 20 seconds. For serious simulations one might wish to increase this number, but with a runtime scaling of N^3 the runtime can quickly become very long.

The next step is to estimate the maximum possible weight for any cube; the number C talked about in section 2.3.2. Weights are given by equation 2.28, so we need the maximum values for factors in the numerator of that equation for each of the two meshes, as well as the minimum gradient length in the mesh.

Coarse cubes have a much larger volume, therefore they contain a lot more possible final states, which subsequently leads to them having a larger weight. Using the same maximum values for both fine and coarse meshes would lead to a selection bias towards coarse mesh cubes in the rejection technique, which would give less accurate results.

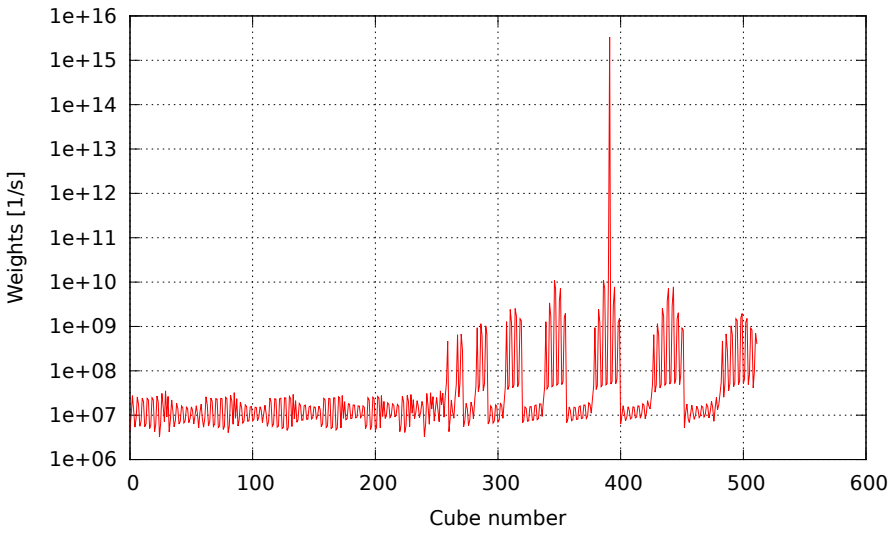
Maximum value of the area of the cross section can be found by a simple geometrical analysis. The largest possible cross section between a cube and a plane is when the plane is parallell to one of the cube's diagonals and also passes through four corners of the cube. If we still use L to denote the side length of a cube, we get a maximum area of $\sqrt{2}L^2$.

Gradient minimum can also be found through several methods. The first method is the simplest one, which only reads in the mesh point values from files and chooses the lowest absolute value. Since we here only need the minimum gradient value for the entire mesh, as opposed to each cube, this method is currently in use. As mentioned in a previous section, an interpolation routine has also been implemented for the derivatives of the band structure. If the current method is found to be inadequate, a similar brute force technique as used in finding the energy maximum and minimum can easily be implemented.

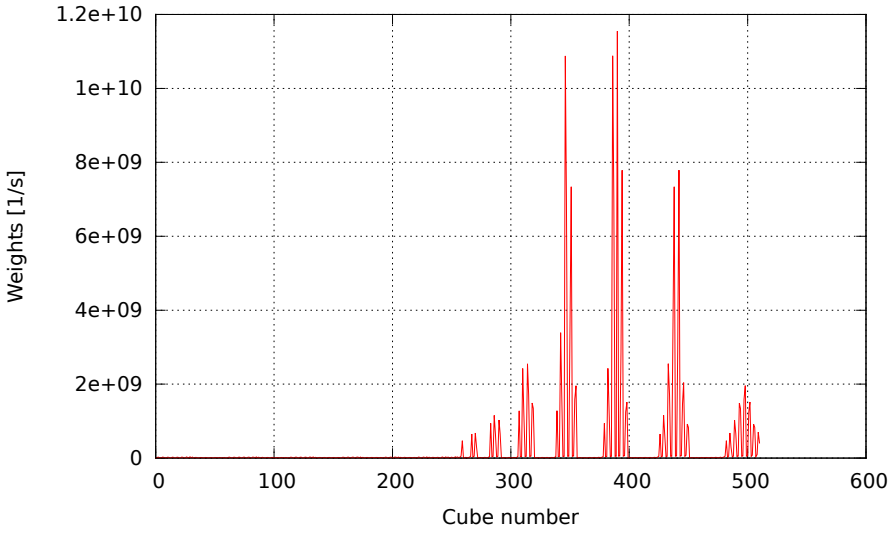
Overlap integrals have a value between zero and one. Hence the maximum value is one, and is then obviously the chosen value.

Estimating the maximum value of the matrix element turned out to be the hardest part. Some matrix elements depend on q, meaning a maximum q had to be found as well. This also appears to be basically a geometry problem. The biggest possible scattering in a geometrical sense is from, e.g., the corner located in the all positive octant to the corner in the all negative octant. This would then give $q_{\max} = 2\sqrt{3}k_{\max}$. But then we run into physical problems, such as the fact that through a shift by a reciprocal lattice vector, these two points are actually really close to one another (but not equivalent, as the square mesh is not a direct representation of the 1BZ). With the square mesh currently in use, we end up covering more than the 1BZ. A proper model of the 1BZ should be implemented in the future. In the current version the mesh itself is treated as the 1BZ. This means that particles which are scattered (or, more likely, translated by an electric field) outside the mesh are simply transformed back into the mesh via a simple translation by $2k_{\max}$. This obviously assumes that particles are never scattered more than twice the maximum wave vector outside of the mesh, which is a reasonable assumption. There is also the fact that different types of scattering mechanisms depend on q in different ways. When one mechanism is proportional with q^{-2} and another one is proportional with just q, the same maximum estimate for q can obviously not be used in calculating the maximum matrix element for both mechanisms.

The issue of differing q-dependences has led to the need for some ad-hoc adjustments of the maximum weight estimates. With the maximum weight too low, nearly all suggestions are immediately accepted and the rejection technique does



(a)



(b)

Figure 3.10: Polar optical absorption, HH → HH. Initial state at 0.12 eV on the [111] axis. Cubes in the fine mesh.

not work as planned. If the estimates are too high, the number of rejections will be huge, and the run time will suffer. For some scattering mechanisms, e.g. non-polar phonon emission, as seen earlier in Fig. 2.8, this is not much of a problem. Here the difference between the lowest and highest weight is about a factor of ten, but most of them are not that far apart. This will lead to the rejection technique quickly accepting a cube. But some scattering mechanisms have weights that do not behave this nicely. An example of this is the polar optical absorption, shown in Fig. 3.10. The two subfigures show the same process, but Fig. 3.10a illustrates the problem with matrix elements inversely proportional to q. Here we can see that at one point a scattering occured with a very small q, leading to a large matrix element and therefore a large weight. It is completely unreasonable to set the maximum weight at six orders of magnitude larger than the value of nearly all weights, as that would cause a tremendous amount of rejections. Fig. 3.10b is the same data, but now with the extreme case removed and no longer with a logarithmic scale. There is still a large variation between weights, so the rejection technique will on average take longer to find an accepted cube than it does for non-polar phonon emission where the weights are closer in magnitude, even when discounting the most extreme cases.

With the maximum weight estimate in place, the next step of the algorithm is to select a candidate cube. This is a completely random choice, independent of mesh type, by the following formula:

$$\text{cube} = \texttt{FLOOR}(r(N_{\text{coarse}} + N_{\text{fine}}) + 1. \tag{3.10}$$

`FLOOR` is a Fortran function for rounding down to the nearest integer, r is a random number between zero and one, and N_{coarse} and N_{fine} are number of cubes found in the coarse and fine mesh. After the selection of a cube, the routine basically follows the procedure as explained in section 2.3.2 up until equation 2.28.

With the weight calculated, the rejection technique is applied. Another random number between zero and one is drawn, and if the product of this random number with the estimated maximum weight is less than the weight of the cube being tested, the cube is accepted. Otherwise, a new cube is picked and the procedure repeats itself until an acceptable **k**-vector is found.

Initially there were problems with finding an acceptable cube. One of the issues was the aforementioned accuracy problems with finding the minimum and maximum energies of a cube. Inaccurate estimates from the gradient method led to the selection of cubes which did not really contain the desired energy. This caused errors which propagated through the gradient length and into the `CORREC` routine, where the energy conserving plane of the cube was actually found outside the cube. This does not make sense, so naturally `CORREC` produced an error.

To test how well this procedure works a test was run for 2000 energy values ranging from 0 to 2 eV, comparing the input energy to the energy at the returned

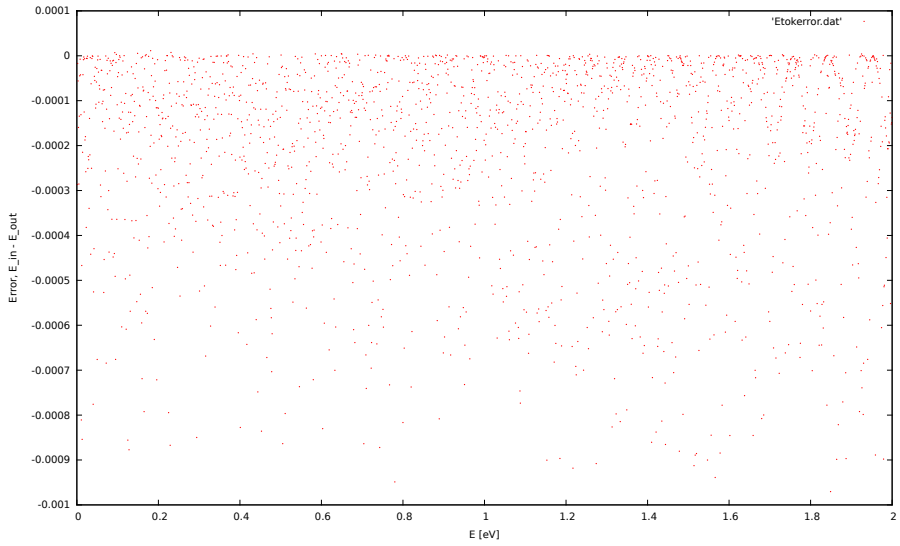


Figure 3.11: Energy difference between input and output energies from Etok. Near k = 0.

k-vector. This was accomplished by using the interpolation routine to find the energy at the **k**-vector returned by `Etok`. Results from an early test can be seen in Fig. 3.11. Here the 6×6 method is used on a mesh close to the zone center. Therefore, the results are very good, with all of the errors below 1 meV and most of them below even 0.1 meV. However, there is an issue in the fact that the errors are always negative. That is, the energy at the **k**-vector returned by `Etok` is always higher than the input energy, which could lead to some artificial heating (energy increase) of the ensemble. This issue is discussed in the next chapter.

However, this method does not work as well when the mesh points are further apart. Fig. 3.12 shows the results of the same procedure, but now with the mesh calculated over the entire 1BZ. A maximum energy of 2 eV is still used, which is far away from the edge of the 1BZ, but a larger mesh naturally causes a larger spacing of the mesh points. Most of the errors are still below Fischetti and Laux' value of 4 meV, but a significant number is also well above. Average energy difference between input and output energy for this figure is -2.9 meV. Due to the randomness in the selection of a final cube, this figure will differ somewhat from run to run. A number of tests have been performed using 10000 points over 2 eV, resulting in average errors in the range of -2.3 to -5.5 meV. The average error is always negative. Noticeable ripples and patterns, like for the bands and rates, can be seen in this data. It is believed, once again, that the problems responsible for the ripples in the band structure also cause this effect.

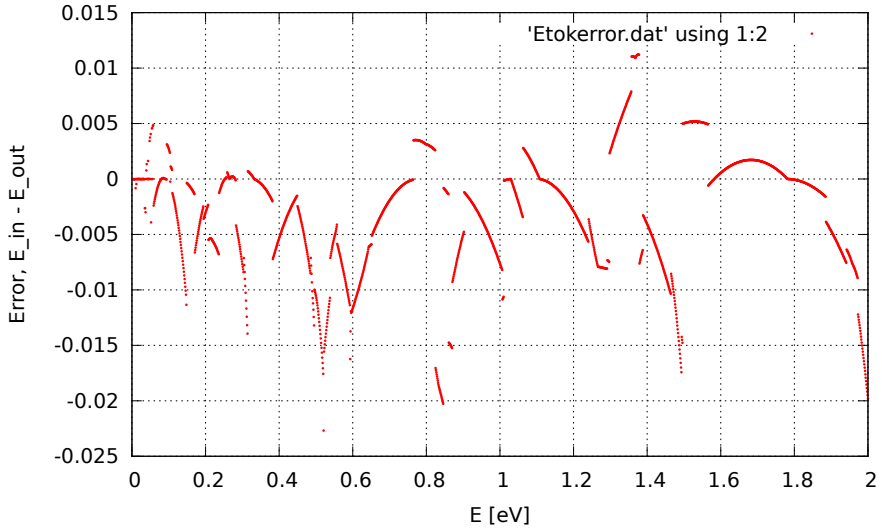


Figure 3.12: Energy difference between input and output energies from Etok. Full zone.

CHAPTER 4

RESULTS AND DISCUSSION

A vast majority of the work in this thesis has been implementing the algorithms detailed in the previous chapter and fixing bugs so everything works properly. Results produced by these routines have already been showcased. Therefore, this chapter will focus on showing results from a full simulation, demonstrating that initial goals have been achieved. Results are plotted from simulations using the 6×6 method and Halvorsen's parameters for GaAs, with a time step length of 1 fs.

It is important that the algorithms for finding the final states are accurate enough to prevent artificial heating or cooling of the particle ensemble. Checking for this is a simple matter of running the simulation with zero external electric field over many time steps and recording the average energy of the particles. In the ideal case with parabolic bands, this energy should converge to $\frac{3}{2}k_BT$. However, since the bands are not ideally parabolic, our results will differ somewhat from this. Olsen found the ensemble hole energy to differ from the thermal energy by about 15% in his thesis [2].

Fig. 4.1 shows the results from a 10 ps long simulation. Particles are initialized with a Gaussian k-distribution. In this figure the ensemble seems to cool down from its initial state rather quickly, getting pretty close to the thermal energy after 5–6 ps. A slight energy increase can be seen over the next 4 ps, with minor oscillations in the energy.

Following the 10 ps long simulation, a second simulation was run for 25 ps. Results can be seen in Fig. 4.2, and the two figures appear qualitatively similar. Curious results appear in Fig. 4.3, when a 40 ps simulation was run after some minor changes to the program, which should be insignificant, from the two other simulations. Here the energy immediately drops, as it does previously, but then

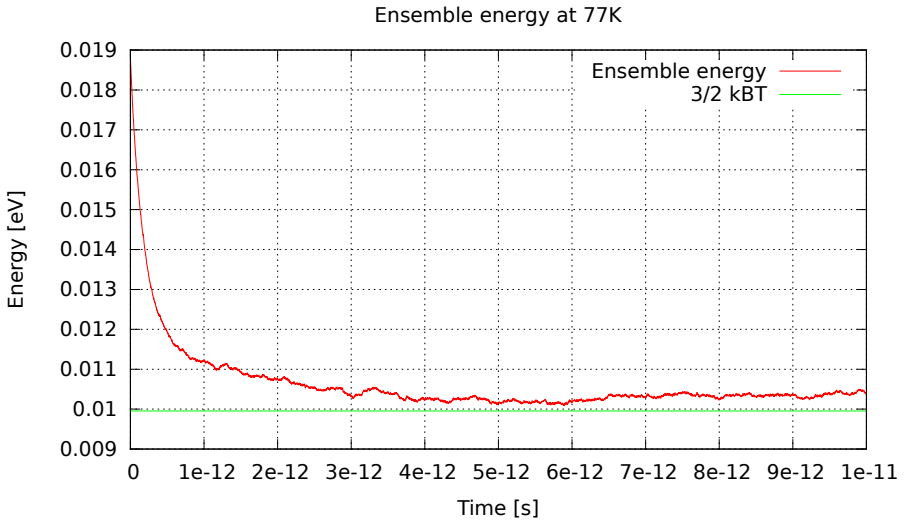


Figure 4.1: Ensemble energy at 77K, 10 ps.

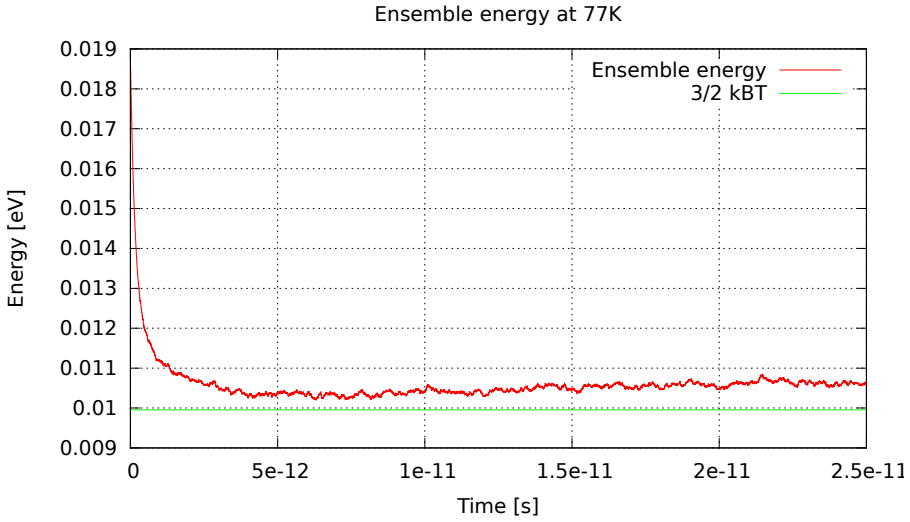


Figure 4.2: Ensemble energy at 77K, 25 ps.

it increases quite a bit and seems to stabilize after about 20 ps. What causes this change in behavior is unclear. One thing that is clear, is that while the current implementation works, it is not yet completely bug free and ready for serious application.

Fig. 4.4 has another display of curious behavior. This simulation run was only

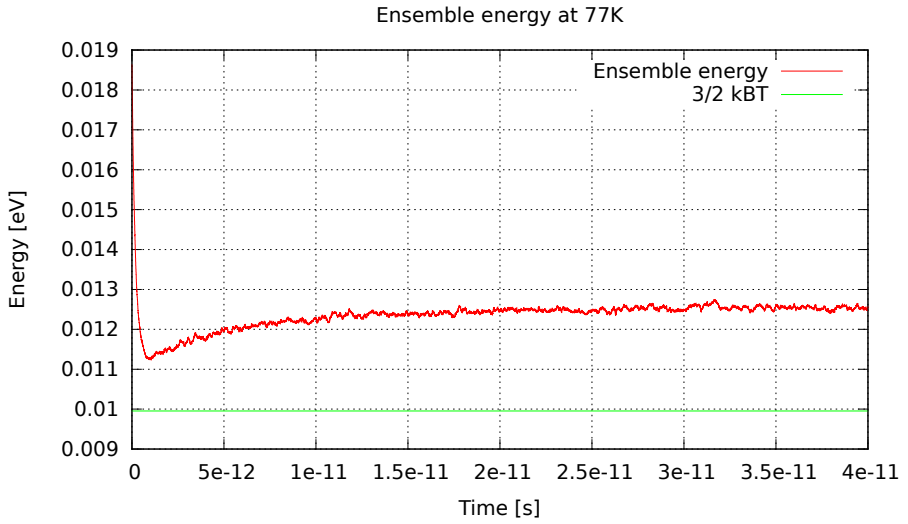


Figure 4.3: Ensemble energy at 77K, 40 ps.

2 ps, yet the energy has nearly completely settled down after as little as 0.7 ps. The corresponding momentum distribution is shown in Fig. 4.5. Momentum distribution is written to file at several times during the simulation. This figure displays the momentum distribution of the holes initially and after 2 ps. After the energy had settled, the momentum distribution did the same. Since there is zero variation in the energy after this point, the results seem almost too good to be true. Other figures display minor oscillations all the time, as one would expect from a numerical method depending on random numbers, while this run converges completely.

While the results look excellent, they can not be trusted to be accurate, due to the drastically different way the energy evolves compared to the earlier runs. It was verified by looking at the numbers themselves that the ensemble energy remains exactly the same from just before 1.2 ps until the simulation was over. With no external electric field to influence the particles, the only way this would happen is if the particles stopped scattering. This particular run had a much higher than usual number of self-scatterings, indicating that something may have happened to the maximum scattering rate estimates. If the estimate was suddenly way higher than it is supposed to be, the probability that self-scattering is the chosen scattering mechanism also increases by a lot. This would explain some of the results seen here.

Scattering mechanisms also appear to be working correctly, with polar optical emission dominating every simulation (except for the last one where self-scattering took over).

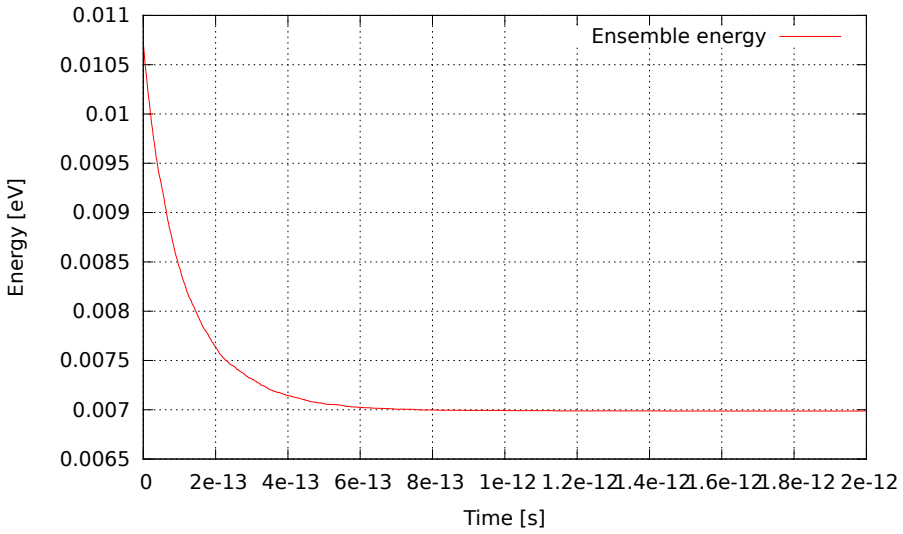


Figure 4.4: Ensemble energy at 77K, 2ps.

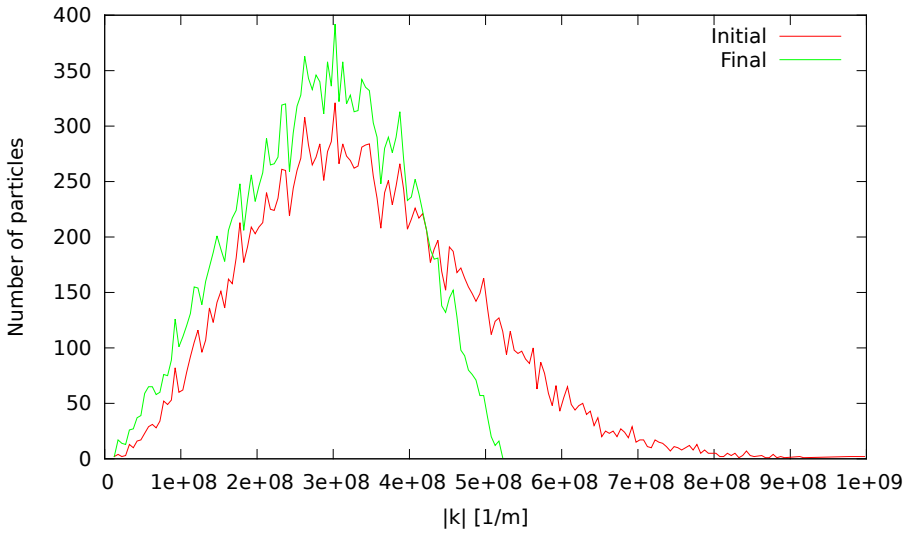


Figure 4.5: Momentum distribution.

While some serious and some less serious bugs still remain, it is clear from these and earlier results that the initial goal of implementing the full-band model into the MC program has been achieved.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

During this work a number of important achievements have been made:

- Halvorsen's old programs have been updated and made more modular, providing Karlsen with a starting point for his thesis.
- The entirety of Halvorsen's programs have been restructured from standalone programs into subroutines which can now be run from the main menu of the MC program.
- Interpolation routines have been implemented for band structure and scattering rates, beginning with one dimension and subsequently expanded to a full three dimensional version.
- Differentiation routines have been implemented for the scattering rates. Though the derivatives are not currently in use by the interpolation routine, they may be needed in the future.
- A smoothing algorithm using a moving average has been implemented as a way of counteracting the rippling in the scattering rates.
- A subroutine for determining final **k**-vector after scattering has been implemented.
- The MC program has been restructured in order to run with the full-band model provided by the Halvorsen-Karlsen programs.
- An extensive amount of minor bug fixes.

Making the program capable of using numerically calculated band structure and scattering rates has been a huge milestone for the development process.

While the program in its current state is capable of running bulk simulations and producing plausible and acceptable results, there is stil much work to do. Listing everything that remains to do on the program as a whole would make for a very long list. What follows is therefore a list of suggestions for future work related to the work done in this thesis.

- Continue improving the quality and readability of the code.
- Implement the Yorston method properly by allowing scattering to take place at arbitrary times within each time step.
- Construct a routine that models the first Brillouin zone properly.
- Design a better algorithm for estimating maximum weights in `Etok`.
- Try to improve the `CORREC` routine used in finding the correction to the final **k**-vector. Corrections should not necessarily be along the energy gradient.
- Determine for sure whether or not the rippled band structure leading to problems all throughout the program is caused by the inaccurate eigenvalue solver. If that is the case, find a better eigenvalue solver. If not, find the cause and fix it.

BIBLIOGRAPHY

[1] Ole Christian Norum. Monte Carlo simulation of semiconductors - program structure and physical phenomena. Master's thesis, Norwegian University of Science and Technology, 2009.

[2] Øyvind Olsen. Construction of a transport kernel for an ensemble Monte Carlo simulator. Master's thesis, Norwegian University of Science and Technology, 2009.

[3] Øyvind Skåring. Ultrashort relaxation dynamics in laser excited semiconductors. Master's thesis, Norwegian University of Science and Technology, Jun 2010.

[4] Camilla N. Kirkemo. Monte Carlo simulation of pn-junctions. Master's thesis, University of Oslo, 2011.

[5] Aksel J. Verne Vestby. Calculation of terminal currents in single photon excited avalanche photodiodes. Master's thesis, Norwegian University of Science and Technology, 2012.

[6] Bjørnar Karlsen. Carrier scattering rates in zincblende structure semiconductors derived from 14×14 $\vec{k} \cdot \vec{p}$ and ab initio pseudopotential methods. Master's thesis, Norwegian University of Science and Technology, 2013.

[7] Ken Vidar Falch. Ensemble averaged and single particle auger lifetimes in zincblende structure semiconductors. Master's thesis, Norwegian University of Science and Technology, 2013.

[8] Trond Brudevoll. *Monte Carlo algorithms for simulation of hole transport in homogeneous semiconductors.* PhD thesis, Norges Tekniske Høgskole, May 1991.

[9] Einar Halvorsen. Numerical calculation of valence band structure and hole scattering rates in GaAs. Master's thesis, Norges Tekniske Høgskole, Feb 1991.

[10] Ronald M Yorston. Free-flight time generation in the monte carlo simulation of carrier transport in semiconductors. *Journal of Computational Physics*, 64(1):177 – 194, 1986.

[11] Carlo Jacoboni and Lino Reggiani. The monte carlo method for the solution of charge transport in semiconductors with applications to covalent materials. *Rev. Mod. Phys.*, 55:645–705, Jul 1983.

[12] Bjørnar Karlsen. Calculation carrier-phonon and carrier-impurity scattering rates for zincblende structure semiconductors with the 8x8 and 14x14 $\mathbf{k} \cdot \mathbf{p}$-methods. *Project thesis, Norwegian University of Science and Technology*, Dec 2012.

[13] G. Gilat and L. J. Raubenheimer. Accurate numerical method for calculating frequency-distribution functions in solids. *Phys. Rev.*, 144:390–395, Apr 1966.

[14] J. D. Wiley. Polar mobility of holes in iii-v compounds. *Phys. Rev. B*, 4:2485–2493, Oct 1971.

[15] Massimo V. Fischetti and Steven E. Laux. Monte carlo analysis of electron transport in small semiconductor devices including band-structure and space-charge effects. *Phys. Rev. B*, 38:9721–9745, Nov 1988.

[16] H. Shichijo and K. Hess. Band-structure-dependent transport and impact ionization in GaAs. *Phys. Rev. B*, 23:4197–4207, Apr 1981.

[17] J. Y. Tang and Karl Hess. Impact ionization of electrons in silicon (steady state). *Journal of Applied Physics*, 54(9):5139–5144, 1983.

[18] Kevin Brennan and Karl Hess. Theory of high-field transport of holes in GaAs and InP. *Phys. Rev. B*, 29:5581–5590, May 1984.

[19] Charles Kittel. *Introduction to Solid State Physics*. John Wiley and Sons, Inc, 8th edition, 2005.

[20] LAPACK zheev.f file reference. `http://www.netlib.org/lapack/explore-html/d6/dee/zheev_8f.html`.