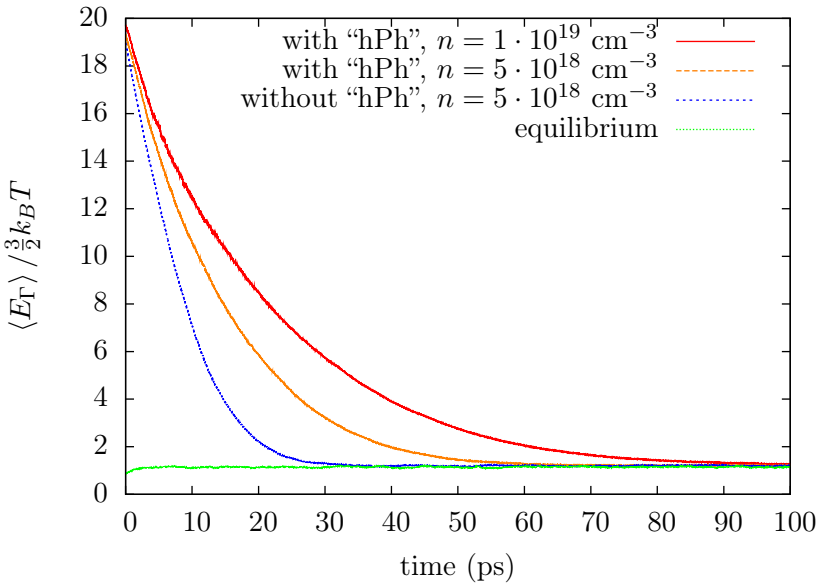
NTNU
Norwegian University of
Science and Technology

Master's thesis

Monte Carlo Simulation of Semiconductors – Program Structure and Physical Phenomena

Monte Carlo Simulering av Halvledere – Programstruktur og Fysiske Fenomen

Ole Christian Norum



June 15th 2009

Preface

This master thesis constitute the 10^{th} semester of the master degree study in technical physics at the Department of Physics at the Norwegian University of Science and Technology (NTNU) and meets the requirements for the subject TFY4900.

This work has been carried out in collaboration with the Norwegian Defence Research Establishment (FFI) under the supervision of Trond Brudevoll (FFI), Asta Katrine Storebø (FFI) and Jon Andreas Støvneng (NTNU).

A huge thanks go to Trond Brudevoll and Asta K. Storebø for their support and excellent guidance throughout the writing of this thesis. I would also like to Øyvind Olsen, my collaborator. And last, but not least, thanks also go to Jon Andreas Støvneng for his help and encouragement.

In order to obtain the latest available source code for the program, an e-mail can be sent to `ole.norum@gmail.com`.

Abstract

During a mere 18 months periode, a Monte Carlo program has been developed to include many features. Both a Poisson solver and hot phonon effects have been implemented. The present Poisson solver has been added only as a "proof of concept". The solver can easily be replaced by more sophisticated solvers or they may be added alongside. The results produced from this solver has proven that the framework is working properly and by implementing a more advanced solver, the user is allowed to simulate e.g. novel and complex transistor structures. An extensive study on the available litterature on the hot phonon effect was carried out and the emphasis of this thesis has been dedicated to the implementation of and the results from this effect. After strong optical stimuli of e.g. a Nd:YAG laser on CdHgTe , the results show a slow-down in the cooling of the carriers by up to 100 ps.

Sammendrag

I løpet av 18 måneder har et Monte Carlo-program med mange egenskaper blitt utviklet. Både en Poissonløser og «hot phonon»-effekter har blitt implementert. Den nåværende Poissonløseren er kun lagt til some et «proof of concept». Løseren kan med letthet byttes ut med mer sofistikerte løsere, eller de kan legges til ved siden av. Resultatene som den nåværende løseren gir viser at rammeverket for Poissonløserene fungerer ordentlig og ved å implementere mer avanserte løsere kan brukeren for eksempel simulere nye og komplekse transistorstrukterer. Et omfattende litteratursøk om «hot phonons» har blitt gjennomført og hovedvekten av denne rapporten omhandler implementasjonen av og resultatene fra denne effekten. Etter sterk optisk stimulans fra for eksempel en Nd:YAG-laser på CdHgTe , viser resultatene at kjølingen av ladningsbærerne kan ta opp til 100 ps lenger om man inkluderer «hot phonon»-effekten.

Contents

Chapter 1

Introduction

This master's thesis describes the development of a semi-classical Monte Carlo program for simulation of two very important semiconductors, namely CdHgTe and GaAs. Although CdHgTe and GaAs have the focus in this program, any semiconductor with a similar bandstructure can be simulated by editing the materialparameters and by further development, any solid state material. GaAs is used in devices such as microwave integrated circuits, solar cells, infrared light emitting diodes and laser diodes. CdHgTe on the other hand is suitable for thermal imaging, night vision, imaging through dense fog, and avalanche photodiodes. [1,2]

The present program has been created in only 1.5 years. Two summerjobs of 2 months each, two project works [3,4] of 4 months each with 50% workload, and finally two master theses [5] (including the present one) of 5 months. This sums up to 18 months. During a project assignment in the autumn of 2008, [3] and the current master's thesis, a lot of work has been invested in eradicating errors and flaws, making the program faster, and of course adding new features. This master's thesis will describe two new features added during this spring. In addition, it will also present the general program structure and operation. The program at hand is a multi purpose program containing a range of physical features.

The emphasis of this thesis has been on the simulation of hot phonon effects. [6–10] During the 1990's this was necessary to understand the slow cooling of the carrier plasma observed in pump-probe laser experiments. The effects of the hot phonons seriously impede the cooling of the hot photoexcited carriers. The simulations of the hot phonon effects require simultaneous evaluation of the electron and hole dynamics, and carrier-carrier interactions. Some of the results presented in this thesis will also be published at this years EDISON[1] conference. [11] On this years conference, a large part of the papers

[1] The 16th international conference on Electron Dynamics In Semiconductors, Opto-

are on hot carriers, underscoring the relevance of the presented program.

Although the program currently focuses on the treatment of bulk for the selected semiconductors, a framework for adding sophisticated Poisson solvers has been created. By adding 2D, or even 3D solvers, the program can very well simulate complex and novel devices. By surveying the abstracts submitted for this years EDISON conference the study of HEMTs is still very interesting. Due to constraints and the priority of the hot phonon effect, such a simulation has not yet been conducted.

There is a vast selection of programs available that includes a Poisson solver, but almost none includes hot phonon effects. Hence, the construction of a modern program is essential in understanding new physics in nanoscale devices. In order to add physical phenomena numerically, a very deep understanding of the physical process is required.

In Chapter 2 the program structure and operation will be explained and presented in detail. Further, the hot phonon effect will be explained in Chapter 3. In Chapter 4 the theory of the present Poisson solver is presented. The results produced from the program will be presented in Chapter 5 and finally, in Chapter 6, a discussion of the program and some concluding remarks will be made. In the Appendix A, the material parameters are presented along with the band structure of CdHgTe and GaAs. To the plots presented in this thesis, a reader has been developed. This can be found in Appendix B along with some plotting scripts written in MATLAB and gnuplot.

The title page plot is discussed in Section 5.2.

electronics and Nanostructures. Can be visited at `www.edison16.net`

Chapter 2

The Program

One of the main features is a Poisson solver, though crude, it is easily replaceable, hot phonon effects, the Pauli principle, a more realistic band structure for $Cd_xHg_{1-x}Te$, and several scattering rates, including carrier-carrier, carrier-plasmon and alloy scatterings.

In addition, the user interface has been hugely improved by implementing a progress bar. This shows the user the estimated time left, as well as how many percent of the execution that is complete. The estimated time left is rather unstable at the beginning but quickly stabilizes to give accurate estimates of the remaining time.

2.1 Program Structure

The program is written in Fortran 90/95. It operates in three different stages:

1. Initialization
2. Execution
3. Post-processing

It contains one main module and several subroutines. During stage 1 the `PROGRAM` module operates as a user I/O interface and allows for user input to be given. It then sends the information to the `MCkernel` subroutine, which initiates the execution process. This subroutine acts as the maestro of the program and executes every action in the right order. It loops through all the timesteps and then loops through the entire ensemble at each timestep, calling the flight, scatter and Poisson subroutines. When the last timestep has been executed the program enters the final stage, post-processing, where the program prints a lot of information to files. This structure is illustrated in the somewhat simplified flowchart in Figure 2.1.

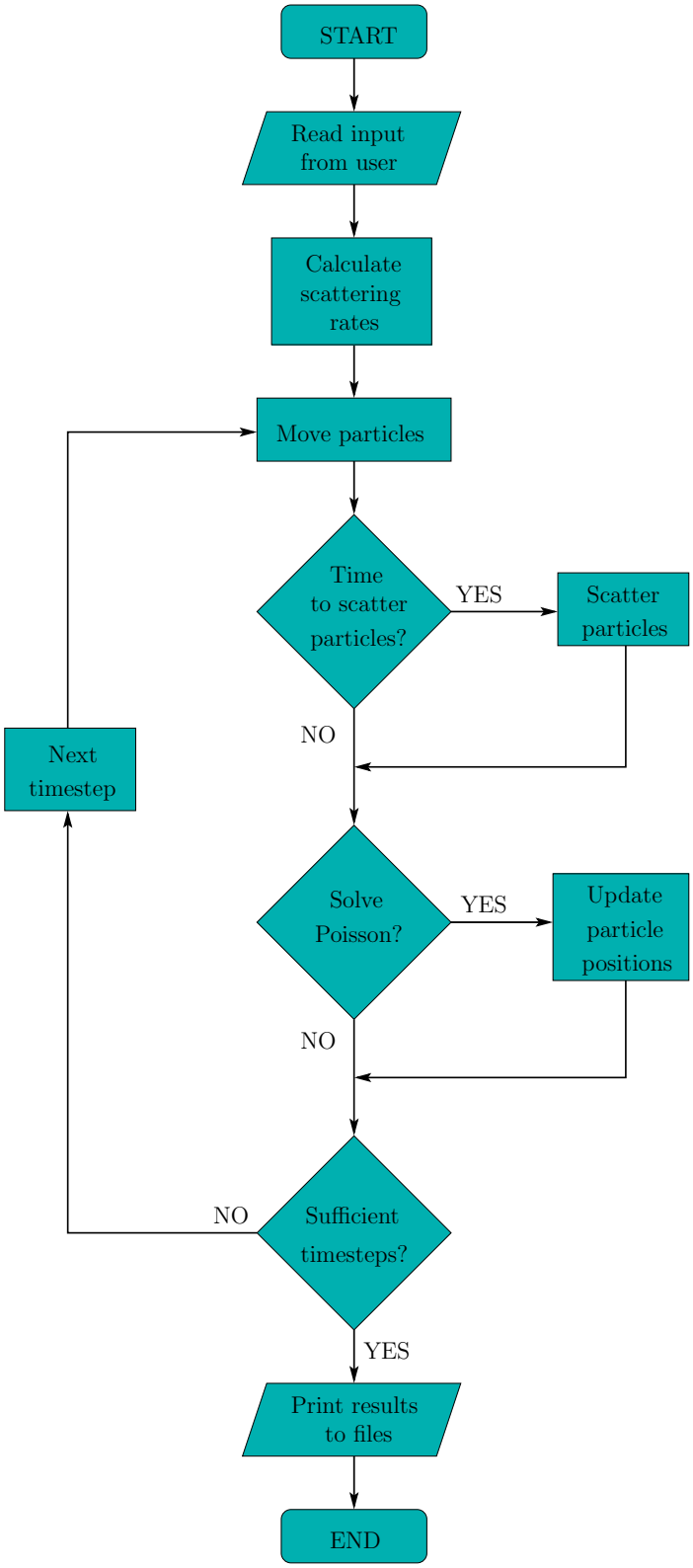


Figure 2.1: A flowchart displaying the most significant steps in the program.

2.1.1 Compilation and Parallelization

In order to compile the program it is recommended to use Intels Fortran compiler, ifort. This is an excellent compiler, and is free for use on Linux. The recommended options is:
`ifort MC2009.6.6.f90 -o mc -ipo -O3 -no-prec-div -static -xHost -parallel`

These options are selected to maximize speed throughout the entire program and optimizes the performance of the executable file. `-parallel` does indeed parallelize small parts of the code. The current Poisson solver is the largest piece of code that allows to be parallelized. If the program is to be run with a huge Poisson mesh with many carriers, it is recommended to use the option `-heap-arrays` to store the arrays on the heap in lieu of on the stack. It may result in some loss in efficiency, but is required in order to prevent the program from crashing due to the large amount of information that has to be stored.

It has been discovered that due to the random number generator[12,13] used in the program, the program will not parallelize. The author suggests three ways around this problem. One can could make the program run in parallel by hand, i.e. divide the large `DO`-loops into a suitable number of pieces and use a separate random number generator for each piece. This is not a very flexible solution and would require an immense amount of work. Another alternative may be to find a different random number generator that allows to be parallelized.[13] This is a viable solution, given that such a random number generator can be found. The last solution would be to implement a framework which tells the compiler how to parallize the code. Examples of such frameworks are OpenMP and MPI. OpenMP is integrated in Intels compiler (can be invoked with compiler option `-openmp`), and several others, and is the recommended framework if such a path is chosen.

To sum up the parallelization. The author would recommend finding a parallel random number generator, or if this cannot be achieved, implement OpenMP.

2.1.2 Executing the Program

The program runs interactively and requires input parameters from the screen. Every menu entry is quite self-explanatory. Two screenshots of the program shows how the A screenshot of the execution of the program is shown in Figures 2.2 and 2.3.

There are several features that may be turned on and off. The type of carriers may be freely set to electrons, holes, or both. The number of carriers does not have to be equal in the case of both electrons and holes. Next, carrier-carrier

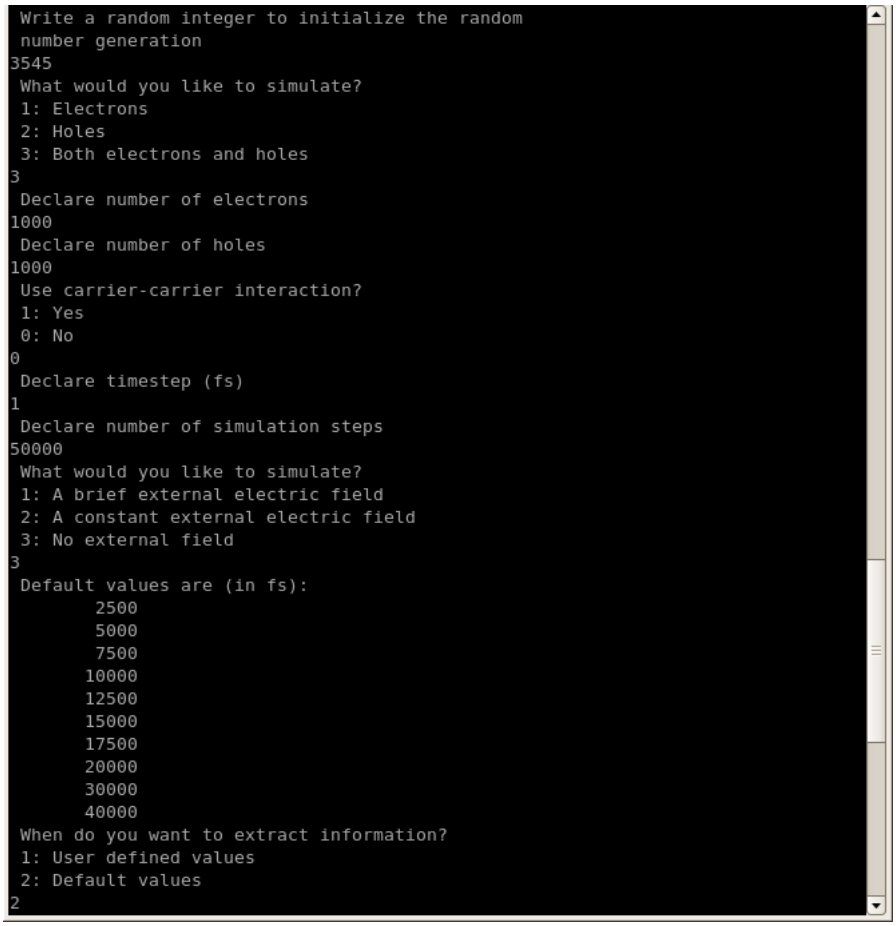


Figure 2.2: The upper part of the menu of the program.

interactions may be turned on and off. The details of this routine is described in Olsen's thesis[5]. Furthermore, the time passed at each simulation step is set. This should be somewhere in the interval 0.1–5 fs. The upper limit is set by the largest scattering rate. If the timestep is any lower than this it will inflict the physical results produced. If carrier-carrier scattering is selected the timestep should be no more than 0.1 fs, depending on the ensemble size.[5]

The next entry in the menu is whether or not a field should be applied. There are three options available. No external field, a constant external field for the entirety of the simulation, or a brief field which can be started and stopped at any point during the simulation. The last option allows study of the effect of an external field after the system has reached equilibrium, and also to study the relaxation effects after the field is turned off.

The program prints ensemble information to files at 12 points in time during

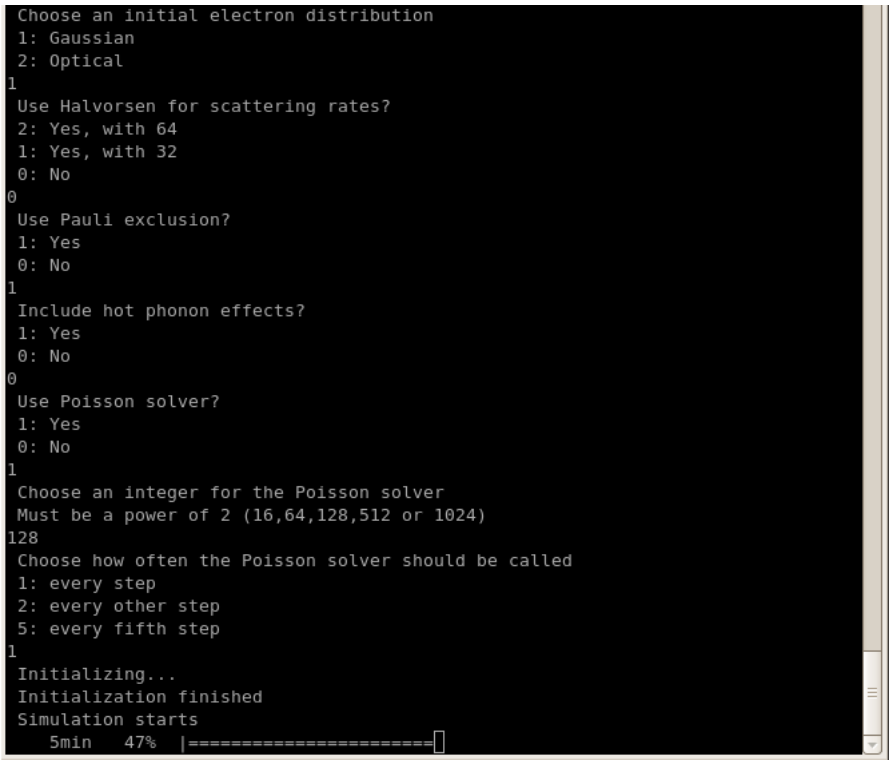


Figure 2.3: The lower part of the menu of the program.

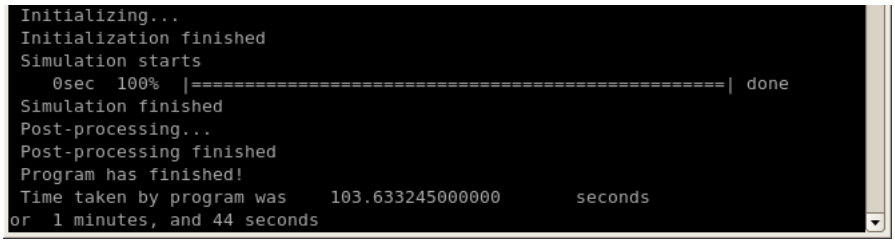


Figure 2.4: The message displayed when program has finished successfully.

the simulation. Once at the beginning and once at the end. During program execution, the program prints information at 10 user specified points in times. This allows closer study at critical points in the simulation, e.g. when an external field is turned on or off, or during the relaxation from an optical distribution. Examples of this is shown in Section 5.1.

The carriers may be distributed in two fashions, Gaussian and optically. When selecting Gaussian, every carrier is placed in the Γ-valley with a Gaussian distributed **k**-vector. If optical distribution is selected, the program ask for a spesific **k** to give to a user specified percentage of the ensemble. Ex-

ample of the optical distribution is also shown in Section 5.1.

The program has been made interoperable with a program developed by Halvorsen.[14] His program creates tabulated files containing empirical scattering rates. The tabulated files are created with either 32 or 64 meshpoints (16 and 32 counting from 0). The user may use any of these sets without recompiling the program, and the current routine, tabulating the scattering rates, scales according to the mesh used in Halvorsen. In the future, the program is planned to be interoperable with *ab initio* programs as well.

Now several of the key features are asked to be turned on or off. The first feature asked for is whether or not to include Pauli exclusion. This is simply an on/off-question. The next feature is whether or not to include hot phonon effects. If selected, the program will ask for a sampling time. The recommended value for this is 10 fs. The sampling time is discussed further in Chapter 3.

Finally the user may choose to use the Poisson solver. Even though the Poisson solver is parallel, it does provide a slow-down for large meshsizes ($\geq$ 512 meshpoints) and large ensemble sizes ($\geq$ 10000 carriers). For this reason, the Poisson solver may be run at every, every other, or every fifth simulation step to decrease time spent.

2.1.3 Post-Processing

After execution of the program has finished, detailed information about each carrier in the ensemble is printed to a file called `info.lst`. In this file the number of scatterings of each type, the position and momentum, and how long it has been in each valley is printed for each carrier in the ensemble. In addition, several other datafiles are created. These contain information in three different areas,

- General information about momentum, position, energy, and valley/band distributions.
- Information about carrier mesh distribution, potential and induced field.
- Information about hot phonon creation.

All but a few of the files are ready for plotting as is, but some require a program of its own to be refined and plotable. This program is displayed in its entirety in Appendix B.1 along with an explanation of its working. Small scripts written in MATLAB and gnuplot for plotting is available in Appendix B.2.

Chapter 3

Hot Phonons

The aggressive downscaling seen today, especially in the CPU industry, requires a whole new perspective. As device designs approach the lower tens of nanometes, heating becomes a huge problem.

Hot phonon effects are very important in terms of evaluating the cooling of the system. When stimulating a semiconducter material optically, a large density of hot photoexcited carriers will be produced. The hot carriers cool themselves by emitting phonons, see Figure 3.1. After a strong laser pulse, a lot of hot photoexcited carriers will start emitting hot phonons, thus altering the phonon occupation number, N_q. This has a large effect on the polar optical scattering rates, see Figure 3.2. As can be seen, the difference between the absorption and emission rates nearly vanish at large carrier temperatures. This impedes the cooling of the system.

3.1 The Hot Phonon Effect

The processes involved in the hot phonon effect is shown in Figures 3.3 and 3.4. In the first diagram, an electron loses some of its energy by emitting a phonon, while in the second diagram, an optical phonon decays into two acoustic phonons. Both of these processes can be reversed, i.e. absorption of a phonon, and the creation of an optical phonon, respectively.

In the program the difference between emitted and absorbed phonons is tracked and accounted for. When stimulated optically, a large number of electrons will be excited at a certain $\mathbf{k}$ corresponding to the wavelength of the laser. This can be set in the menu of the program. The hot electrons cool themselves by emitting phonons. These phonons will have a certain maximum $\mathbf{q}_0$. As can be seen from Figure 3.2 the emission rate is much larger than the absorption rate, and hence represents a cooling effect on the electrons as they deposit their energy to the lattice.

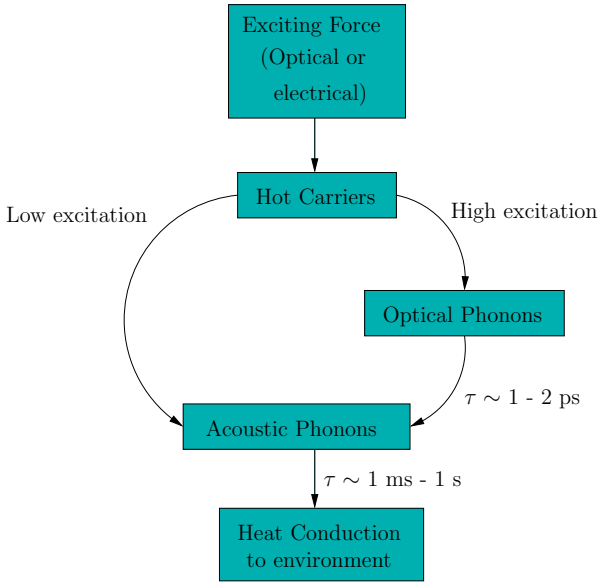


Figure 3.1: A small diagram of characteristic times and energies for cooling of CdHgTe .

But during laser excitation, phonons will accumulate and the carrier temperature will rise, affecting the scattering rates, see Figure 3.2. As can be seen, the difference between the scattering rates vanishes at sufficiently large carrier temperatures. When this happens, the cooling of the electrons is far, far smaller, since the electrons emit and equally often absorb phonons.

Hot optical phonons cool themselves by decaying into acoustic phonons, releasing their energy to the lattice, see Figure 3.1. This is how the system cool itself after optical stimuli.

The decay of optical phonons into acoustic phonons by anharmonic decay process can be described by[6]

$$\left(\frac{\partial N_q}{\partial t}\right)_{ph-ph} = -\frac{N_q - N_q(T_L)}{\tau_{ph}} \tag{3.1}$$

where τ_{ph} is the phonon lifetime, assumed equal for all q. N_q is the Bose-Einstein non-equilibrium phonon occupation number and $N_q(T_L)$ is the phonon occupation number at the lattice temperature T_L. N_q is given as

$$N_q = \frac{1}{\mathrm{e}^{\hbar\omega_0/k_B T} - 1} \tag{3.2}$$

The lifetime of the optical phonons can be calculated from *ab initio* electronic structure methods. According to Shah,[6,15] the liftime is of the order 1–2 ps. This is sufficiently long to assume semi-equlibrium and hence, we may allow

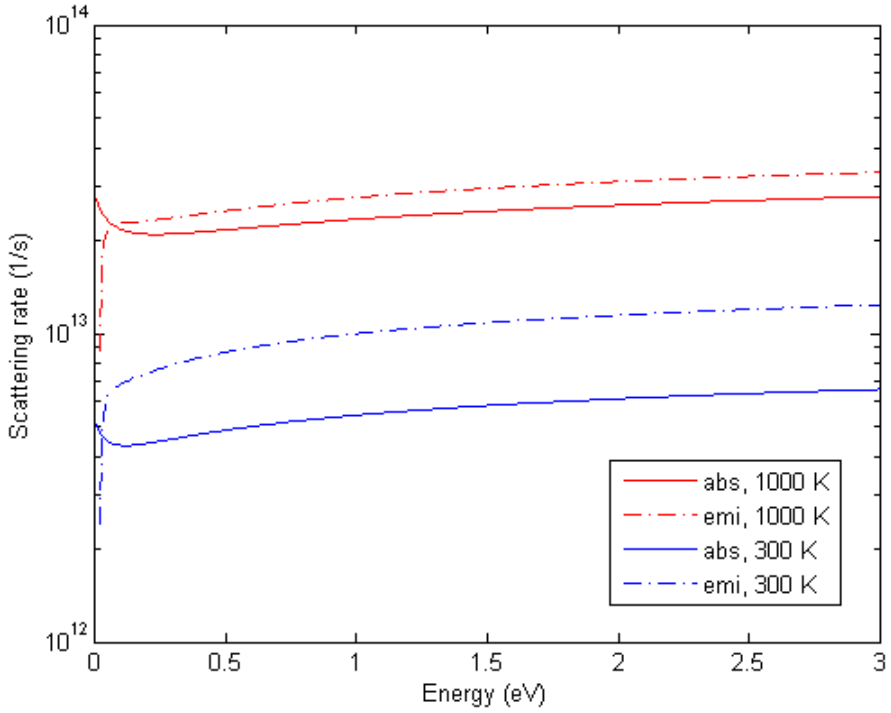


Figure 3.2: The polar optical rates for emission and absorption at two carrier temperatures, 300 K and 1000 K.

ourselves to set the right hand side of the above equation equal to zero. This is crude, but simplifies the calculations quite drastically. In addition, the phonons do not just decay and disappear from the system, they are also created through emission from hot electrons. Hence, we require a generating term as well. This gives us the following expression,

$$-\frac{N_q - N_q(T_L)}{\tau_{ph}} + \left(\frac{\#\text{net generated phonons}}{\text{unit volume and time}}\right) = 0 \tag{3.3}$$

The number of states in **q**-space is

$$\frac{V}{(2\pi)^3}\int_0^{q_0} \mathrm{d}^3 q = \frac{V}{(2\pi)^3}\frac{4\pi q_0^3}{3} \tag{3.4}$$

This introduces a V-term in our expression, which is not desired in expressions describing bulk materials. To get rid of the volume we can sacrifice the superparticle to real carrier ratio.[16] This ratio is given as

$$N_{ratio} = \frac{n_e}{n_{sup}} = \frac{N_e}{N_{sup}} = \frac{n_e V}{N_{sup}} \tag{3.5}$$

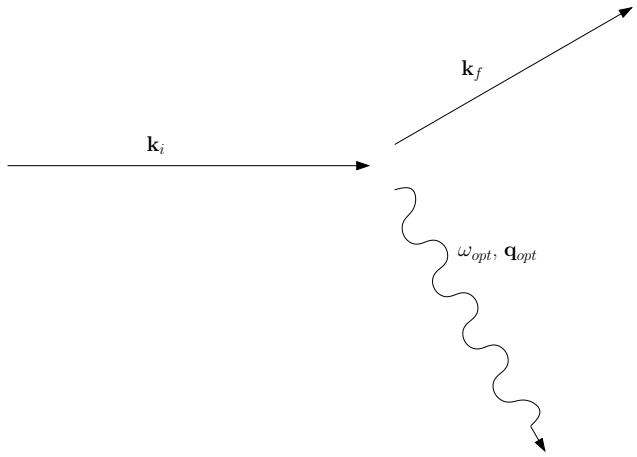


Figure 3.3: The emission of a phonon. This is the generating process in Eq. (3.1). The process may be reversed.

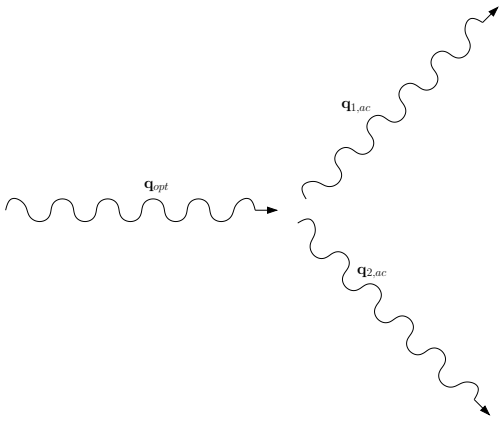


Figure 3.4: The decay of an optical phonon into two acoustic phonons. This is the decay term introduced in Eq. (3.4). The process may also be reversed.

with

$$n = \frac{N}{V} \tag{3.6}$$

where n is the particle density and N is the number of particles in the volume V.

Executing the sacrifice gives the final expression

$$N_q = \frac{\#\text{net generated superphonons } \frac{n_e V}{N_{sup}}}{\frac{V}{(2\pi)^3}\frac{4\pi q_0^3}{3}} \frac{\tau_{ph}}{t_{sample}} + N_q(T_L) \tag{3.7}$$

In the program an array called `phgentot` is reserved for counting the net

generated superphonons at each timestep. At each timestep another variable called `phgen` is calculated by summing up every net created superphonon during the last timesteps corresponding to a sampling time for the hot phonons. By introducing this and rewriting the expression we end up with

$$N_q = \texttt{phgen} \cdot \frac{n_e}{N_{sup}} \frac{6\pi^2}{q_0^3} \frac{\tau_{ph}}{t_{sample}} + N_q(T_L) \tag{3.8}$$

Hence, we end up with a numerical expression for the non-equilibrium phonon occupation number!

Chapter 4

The Poisson Equation

The most important field equation that needs to be solved in particle simulations is the Poisson equation, (4.1). The Poisson equation describes the variation of the potential ϕ due to the charge density ρ.

$$\nabla^2\phi = -\frac{\rho(\mathbf{r})}{\varepsilon} \tag{4.1}$$

where ε is the dielectric constant.

The program consists of a series of free flights that may end in a scattering event. Since the accelerated particles are charged, electrical fields will be induced, which will affect their behaviour during the free flight, especially in non-equilibrium conditions. This may result in effects like dynamic screening and may have a large effect on the results in several areas.

Because of the importance of the Poisson equation in particle simulations, there has been a great interest in solving it numerically. The drawback of solving the Poisson equation, especially in 3D, is that it easily becomes one of the most computer intensive parts of a simulation program. Over the years, several numerical schemes has been produced to solve this equation in a more economic way. In a previous report [3] several schemes for solving the Poisson equation was considered and reviewed.

In the next Section, the theory behind the present Poisson solver is presented. For more information on charge assignment schemes the reader is referred to the previous work of this author. [3]

4.1 Poisson solvers

There are several ways to attack this problem. Some popular choices include the Fourier Analysis/Cyclic Reduction algorithm (FACR) produced by Hockney in 1965 [12,17,18] and the Fast Multipole Method (FMM) [19–21].

However, in the current program, only a simple solver based on brute-force matrix multiplications has been implemented. The solver is to be considered more as a "proof of concept" rather than a fully fledged solver. All the framework has been designed to be very flexible with regards to the choice of Poisson solvers. In theory, any solver may be implemented in place of the existing one without any trouble. Some theory on the existing solver will be presented shortly.

The following notation will be used;

$$\partial_x^2 u(x,y) + \partial_y^2 u(x,y) = u_{xx} + u_{yy} = -f(x,y) \tag{4.2}$$

where the source term f, the charge distribution, is considered known, and $\partial_x \equiv \frac{\partial}{\partial x} u \equiv u_x$ is a convenient short hand notation. In the this treatment only two dimensions will be examined, although the methods easily extends to three dimensions. However, for the purpose of explaining the methods, concentrating on two dimensions is advantageous.

The Poisson problem falls into the category of *boundary value problems*, meaning that one wants the solution of the problem, $u(x,y)$, to satisfy the equation within the area of interest, (x,y), and also to satisfy the boundary conditions. Since all of the boundary conditions must be satisfied simultaneously, the solution of boundary value problems can be viewed as the solution of large linear sets of equations.

4.1.1 Discretization

To solve the Poisson equation numerically, it needs to be discretized. There are several ways to do this. In the program, finite difference has been chosen. In particular a five point finite difference scheme is selected.

Let us consider the Poisson problem in a rectangular domain, Ω, with lengths L_x and L_y, see Fig. 4.1.

The five points of interest is shown in Fig. 4.2. The first step is to rewrite Eq. (4.2) in difference form. This is done by Taylor expansions [22] as follows,

$$\begin{aligned} u(x+h,y) &= u(x,y) + h_x u_x(x,y) + \frac{h_x^2}{2} u_{xx}(x,y) + \frac{h_x^3}{6} u_{xxx}(x,y) + \dots \\ u(x-h,y) &= u(x,y) - h_x u_x(x,y) + \frac{h_x^2}{2} u_{xx}(x,y) - \frac{h_x^3}{6} u_{xxx}(x,y) + \dots \end{aligned} \tag{4.3}$$

where h_x is the mesh size in the x-direction. Now, subtract $u(x+h_x,y) - u(x-h_x,y)$, neglect higher order terms and solve for u_x. This leaves

$$u_x(x,y) \simeq \frac{1}{2h}(u(x+h_x,y) - u(x-h_x,y)) \tag{4.4}$$

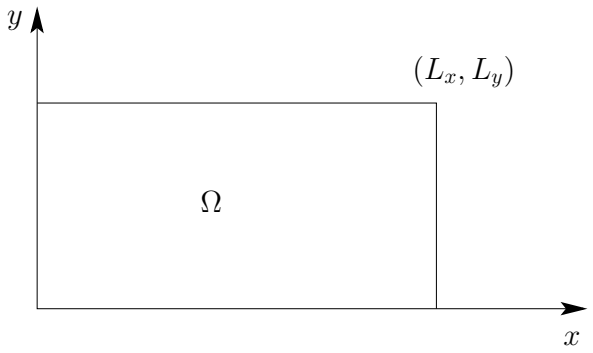


Figure 4.1: The domain of interest for the two-dimensional Poisson problem.

The same can be done for the y-direction giving

$$u_y(x,y) \simeq \frac{1}{2h}(u(x,y+h_y) - u(x,y-h_y)) \tag{4.5}$$

where h_y is the mesh size in the y-direction. Now, add the equations (4.3), neglect higher order terms and solve for u_{xx}. By the same procedure one finds for both u_{xx} and u_{yy}

$$\begin{aligned} u_{xx}(x,y) &\simeq \frac{1}{h_x^2}(u(x+h_x,y) - u(x-h_x,y)) \\ u_{yy}(x,y) &\simeq \frac{1}{h_y^2}(u(x,y+h_y) - u(x,y-h_y)) \end{aligned} \tag{4.6}$$

It can be shown that the mixed terms u_{xy} are not needed.[22]

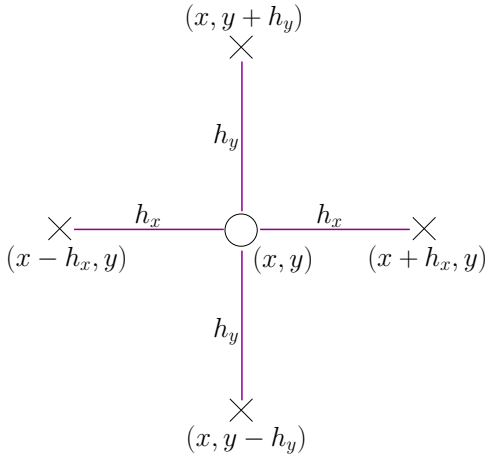


Figure 4.2: The five points of interest when differentiating the Poisson equation.

Now, substitute Eq. (4.6) back into the Poisson equation (4.2), this gives us the following difference equation

$$\begin{aligned} &\frac{2u(x,y) - u(x+h_x,y) - u(x-h_x,y)}{h_x^2} + \\ &\frac{2u(x,y) - u(x,y+h_y) - u(x,y-h_y)}{h_y^2} = f(x,y) \end{aligned} \tag{4.7}$$

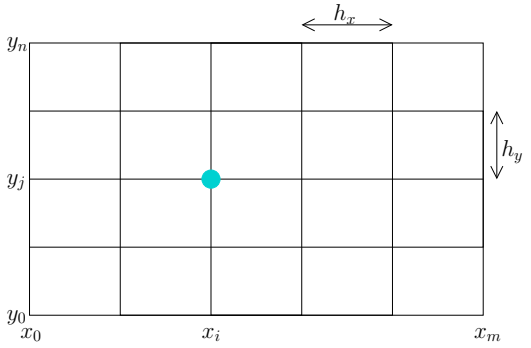


Figure 4.3: The finite difference grid.

The finite difference grid points (or nodes), as shown in Fig. 4.3, are given by

$$\begin{aligned} x_i &= i \cdot h_x \qquad i = 0, 1, \ldots, m \qquad h_x = \frac{L_x}{m} \\ y_j &= j \cdot h_y \qquad j = 0, 1, \ldots, n \qquad h_y = \frac{L_y}{n} \end{aligned} \tag{4.8}$$

Using the notation $u_{i,j} \simeq u(x_i, y_j) = u(ih_x, jh_y)$ and $f_{i,j} = f(x_i, y_j)$, see Figure 4.3. Also, assuming $h_x = h_y = h$ and $m = n$ renders Eq. (4.7) in a very neat form

$$\frac{2u_{i,j} - u_{i+1,j} - u_{i-1,j}}{h^2} + \frac{2u_{i,j} - u_{i,j+1} - u_{i,j-1}}{h^2} = f_{i,j} \qquad 1 \leq i, j \leq n-1 \tag{4.9}$$

We now have the discretized version of the Poisson equation.

4.1.2 Solving

The method in the program is a direct method based on diagonalization. The simplest is to explain the approach in context of the one dimensional problem.

$$u_{xx} = -f \tag{4.10}$$

Add the assumption of a uniform grid, i.e. $h_x = h_y = h$ and $m = n$, given by

$$x_i = x_0 + ih \qquad i = 1, \ldots, n-1 \tag{4.11}$$

By using the one dimensional form of Eq. (4.9) and the above assumptions, the corresponding system of equations may be written as

$$\begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & & 0 \\ 0 & -1 & 2 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & -1 \\ 0 & 0 & \cdots & -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{n-1} \end{pmatrix} = h^2 \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_{n-1} \end{pmatrix} \tag{4.12}$$

or equivalently

$$\mathbf{Tu} = h^2\mathbf{f} \tag{4.13}$$

Since $\mathbf{T}$ is symmetric positive definite, it can be diagonalized. This means finding the eigenvalues λ_j and the eigenvectors $\mathbf{q}_j$ of $\mathbf{T}$

$$\mathbf{Tq}_j = \lambda_j \mathbf{q}_j \qquad j = 1, \ldots, n-1 \tag{4.14}$$

where λ_j are positive eigenvalues ($\lambda_j > 0$) and the egenvectors are orthonormal ($\mathbf{q}_k^{\mathrm{T}}\mathbf{q}_j = \delta_{jk}$). Here, δ is the Kronecker delta.

By collecting all the eigenvectors $\mathbf{q}_j$ in an orthonormal matrix $\mathbf{Q}$

$$\mathbf{Q} = (\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{n-1}) \tag{4.15}$$

we can write

$$\mathbf{TQ} = \mathbf{Q}\Lambda \tag{4.16}$$

where

$$\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_{n-1}) = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_{n-1} \end{pmatrix} \tag{4.17}$$

Since the eigenvectors are orthonormal

$$\mathbf{Q}^{\mathrm{T}}\mathbf{Q} = \mathbf{I} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} \qquad \Rightarrow \mathbf{Q}^{\mathrm{T}} = \mathbf{Q}^{-1} \tag{4.18}$$

and hence

$$\mathbf{T} = \mathbf{Q}\Lambda\mathbf{Q}^{\mathrm{T}} \tag{4.19}$$

or equivalently

$$\mathbf{Q}^{\mathrm{T}}\mathbf{T}\mathbf{Q} = \Lambda \tag{4.20}$$

Now, in two dimensions $\mathbf{u}$ and $\mathbf{f}$ are not vectors any more, but matrices. Let

$$\mathbf{U} = \begin{pmatrix} u_{1,1} & \cdots & \cdots & u_{1,n-1} \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ u_{n-1,1} & \cdots & \cdots & u_{n-1,n-1} \end{pmatrix} \tag{4.21}$$

and

$$\mathbf{T} = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} \tag{4.22}$$

By multiplying $\mathbf{T}$ and $\mathbf{U}$ we get

$$\begin{array}{ll} (\mathbf{TU})_{i,j} = 2u_{i,j} - u_{i+1,j} & i = 1 \\ (\mathbf{TU})_{i,j} = 2u_{i,j} - u_{i+1,j} - u_{i-1,j} & i = 2, \ldots, n-2 \\ (\mathbf{TU})_{i,j} = 2u_{i,j} - u_{i-1,j} & i = n-1 \end{array} \tag{4.23}$$

Relating this to the discretized version of the Poisson equation, Eq. (4.9), we obtain

$$\frac{1}{h^2}(\mathbf{TU})_{i,j} \simeq -\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} \tag{4.24}$$

Equivalently for the derivative in the y-direction

$$\frac{1}{h^2}(\mathbf{UT})_{i,j} \simeq -\left(\frac{\partial^2 u}{\partial y^2}\right)_{i,j} \tag{4.25}$$

We can now rewrite Eq. (4.9)

$$\frac{1}{h^2}(\mathbf{TU} + \mathbf{UT})_{i,j} = f_{i,j} \qquad \text{for } 1 \leq i, j \leq n-1 \tag{4.26}$$

or

$$\mathbf{TU} + \mathbf{UT} = \mathbf{G} \tag{4.27}$$

where

$$\mathbf{G} = h^2 \begin{pmatrix} f_{1,1} & \cdots & \cdots & f_{1,n-1} \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ f_{n-1,1} & \cdots & \cdots & f_{n-1,n-1} \end{pmatrix} \tag{4.28}$$

By combining Eqs. (4.20) and (4.27) we get

$$\mathbf{Q}\Lambda\mathbf{Q}^{\mathrm{T}}\mathbf{U} + \mathbf{U}\mathbf{Q}\Lambda\mathbf{Q}^{\mathrm{T}} = \mathbf{G} \tag{4.29}$$

Multiplying Eq. (4.29) by $\mathbf{Q}$ from the right and $\mathbf{Q}^{\mathrm{T}}$ from the left and using $\mathbf{Q}^{\mathrm{T}}\mathbf{Q} = \mathbf{I}$, we get

$$\Lambda \underbrace{\mathbf{Q}^{\mathrm{T}}\mathbf{U}\mathbf{Q}}_{\equiv\widetilde{\mathbf{U}}} + \underbrace{\mathbf{Q}^{\mathrm{T}}\mathbf{U}\mathbf{Q}}_{\equiv\widetilde{\mathbf{U}}}\Lambda = \underbrace{\mathbf{Q}^{\mathrm{T}}\mathbf{G}\mathbf{Q}}_{\equiv\widetilde{\mathbf{G}}} \tag{4.30}$$

This implies that Eq. (4.9) may be solved in the following three steps.

Step 1: Compute the matrix-matrix products

$$\widetilde{\mathbf{G}} = \mathbf{Q}^{\mathrm{T}}\mathbf{G}\mathbf{Q} \tag{4.31}$$

Step 2: Solve the equation sets

$$\Lambda\widetilde{\mathbf{U}} + \widetilde{\mathbf{U}}\Lambda = \widetilde{\mathbf{G}} \tag{4.32}$$

This is done by finding $\widetilde{u}_{i,j}$

$$\begin{aligned} \lambda_i\widetilde{u}_{i,j} + \widetilde{u}_{i,j}\lambda_j &= \widetilde{g}_{i,j} \qquad 1 \leq i,j \leq n-1 \\ (\lambda_i + \lambda_j)\widetilde{u}_{i,j} &= \widetilde{g}_{i,j} \qquad 1 \leq i,j \leq n-1 \\ \widetilde{u}_{i,j} &= \frac{\widetilde{g}_{i,j}}{\lambda_i + \lambda_j} \qquad 1 \leq i,j \leq n-1 \end{aligned} \tag{4.33}$$

Step 3: Compute the matrix-matrix products

$$\mathbf{U} = \mathbf{Q}\widetilde{\mathbf{U}}\mathbf{Q}^{\mathrm{T}} \tag{4.34}$$

Hence, we are left with $\mathbf{U}$, which is the solution to the Poisson equation!

Chapter 5

Simulation Results

In this chapter, all the simulation results has been gathered. Both general simulation results displaying characteristics of the carrier distribution in the different valleys under influence of external forces, simulations showing the hot phonon effect, and the results from the Poisson solver at hand. They will be presented in the same order as the theory discusses them. Unless otherwise stated, all the simulations has been performed at 300 K and for $Cd_xHg_{1-x}Te$, $x = 0.275$.

The previous versions of the program used ∼30 hours to conclude a simulation of 20000 electrons and holes.[3] This is very slow compared to the present version of the program, which only uses ∼2 minutes using the same parameters and the same features as in that report. The reason for this reduction in time is mainly caused by the tabulation of the scattering rates. Also, time reduction was obtained by clean-up of the code. It should also be mentioned that when running the program with the same parameters including carrier-carrier interactions, the previous version required almost no extra computing time. The current carrier-carrier interactions is extremely slow.[5]

The simulations presented in the following sections have all been run on an Intel Core2 Quad CPU Q9400 machine running Red Hat Enterprice Linux 5. The processors are clocked at 2.66 GHz and the available memory is 4 GB RAM. The program has been compiled with Intels Fortran Compiler for Linux, version 11.0 with the following options

```
ifort MC2009.6.6.f90 -o mc -ipo -O3 -no-prec-div -static -xHost
-parallel
```

5.1 General Results

Since the program allows the user to pick 10 arbitrary points during the simulation to extract data, extensive study of different physical phenomena can be made, e.g. characteristic times for the relaxation of the **k**-distribution after optical stimuli, or the shift in the $\mathbf{k}_z$-distribution when applying an external field.

In the following, no discussion of such characteristic times will be made. The graphs are only presented to show the possibilites the program offers. In addition to the examples made above, the carrier distribution to the different valleys during an external field will also presented.

5.1.1 Optical distribution

The program is capable of simulating the relaxation of photoexcited carriers after optical stimuli. Before the execution of the program is started, a user specified portion of the carriers are given a user specified **k**. This is shown as the large initial peak in Figures 5.1 to 5.4. As can be seen in the figures, the peak quickly disperses and "wanders" and widens into a thermalized Maxwellian distribution. The lines corresponding to 16 and 20 ps indicate that the system has reached steady state.

Another effect that can be seen is the effect of the carrier-carrier interactions. It has a smoothing effect on the distributions. This is especially clear when looking at the distribution at 16 and 20 ps. The plots including the Pauli principle show that the distribution becomes somewhat lower and wider.

In all of the Figures 5.1, 5.2, 5.3 and 5.4, the plots has been cut at 750 superelectrons to better follow the movement of the distribution. The initial peak in all of the plots contains 1900 superelectrons. The wavelength of the laser that has been used to excite the initial carriers is $\lambda = 1064$ nm, corresponding to a Nd:YAG-laser.

In Figures 5.5 and 5.6 the corresponding $\mathbf{k}_z$ distribution and times from Figures 5.1 and 5.3 are shown. In Figures 5.7 and **??** the data for some points in time has been removed to better study the thermalization effects. What can be read from the figures is that after $\sim$2–4 ps a bellshape arises.

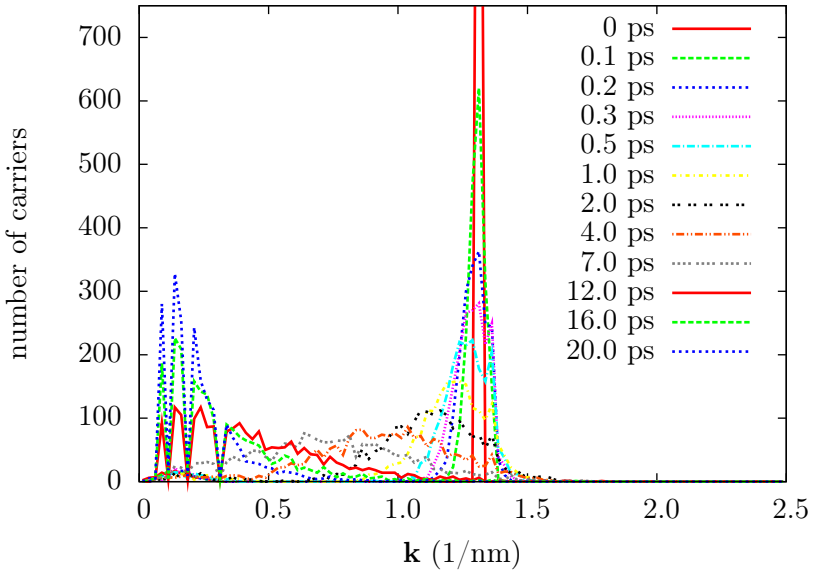


Figure 5.1: The **k**-distribution for $Cd_{0.275}Hg_{0.725}Te$ as time progresses. In this simulation there are 2000 superelectrons and $n = 10^{18}$ cm^{-3}.

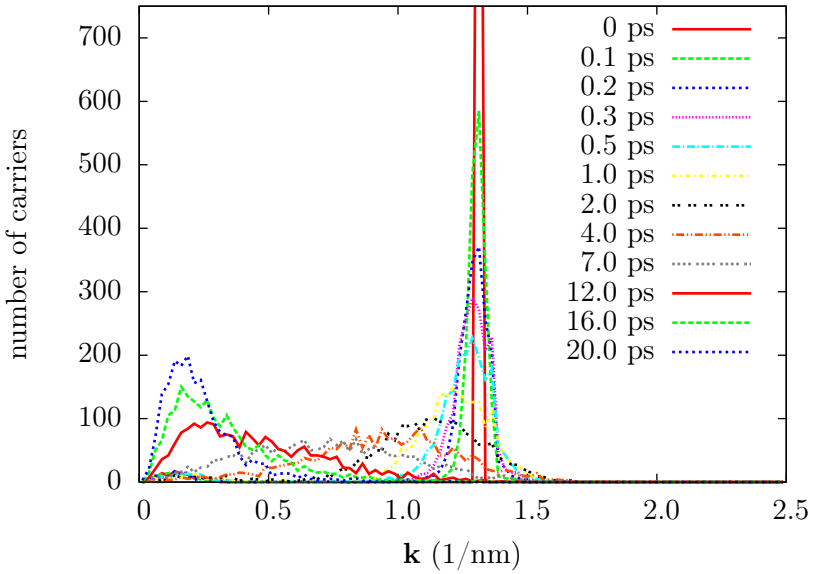


Figure 5.2: The **k**-distribution for $Cd_{0.275}Hg_{0.725}Te$ as time progresses including carrier-carrier interactions. In this simulation there are 2000 superelectrons and $n = 10^{18}$ cm^{-3}.

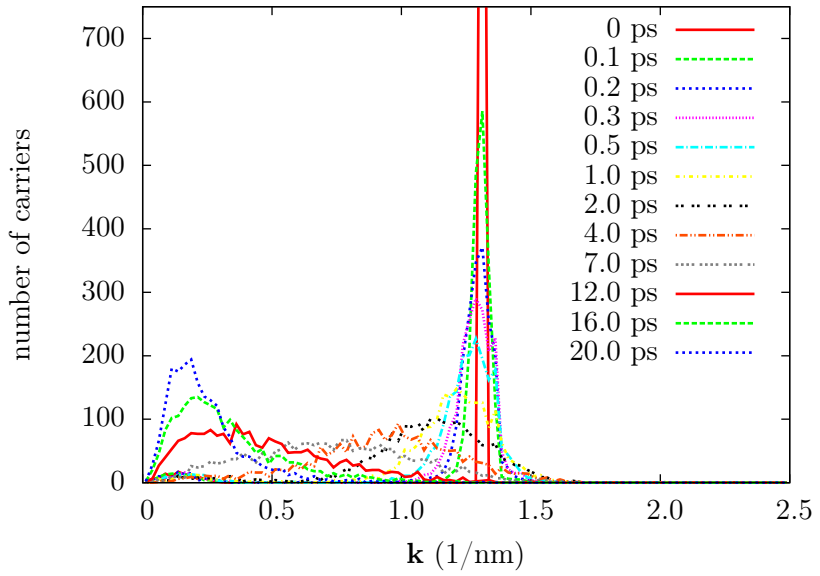


Figure 5.3: The **k**-distribution for $Cd_{0.275}Hg_{0.725}Te$ as time progresses including carrier-carrier interactions and the Pauli principle. In this simulation there are 2000 superelectrons and $n = 10^{18}$ cm^{-3}.

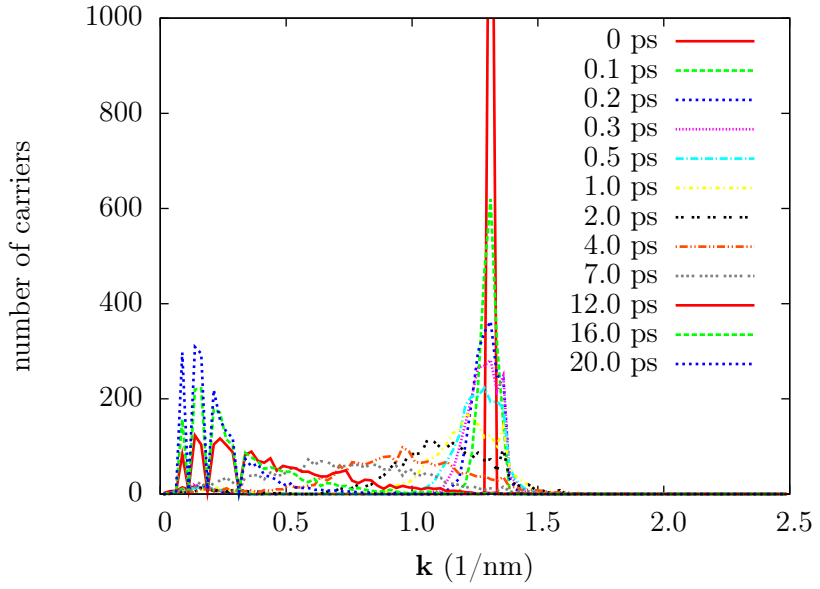


Figure 5.4: The **k**-distribution for $Cd_{0.275}Hg_{0.725}Te$ as time progresses with the Pauli principle. In this simulation there are 2000 superelectrons and $n = 10^{18}$ cm^{-3}.

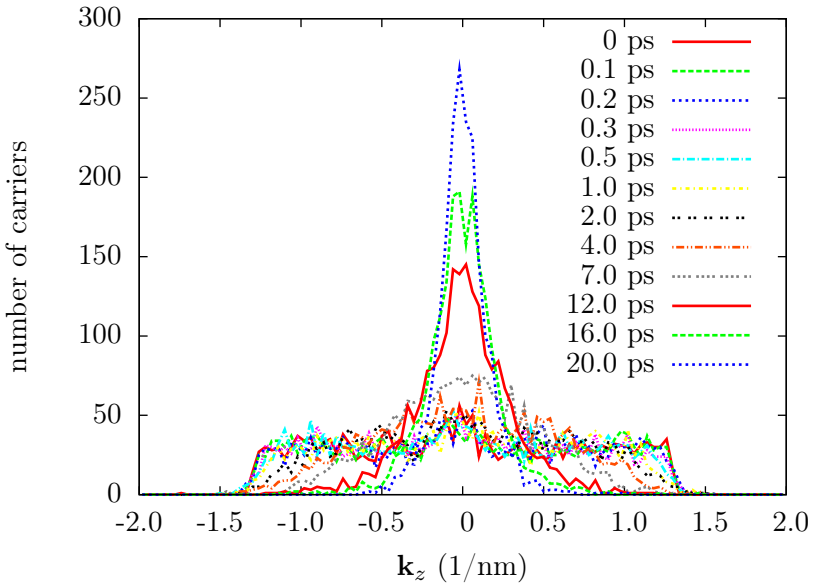


Figure 5.5: The $\mathbf{k}_z$-distribution for $Cd_{0.275}Hg_{0.725}Te$ as time progresses. In this simulation there are 2000 superelectrons and $n = 10^{18}$ cm^{-3}.

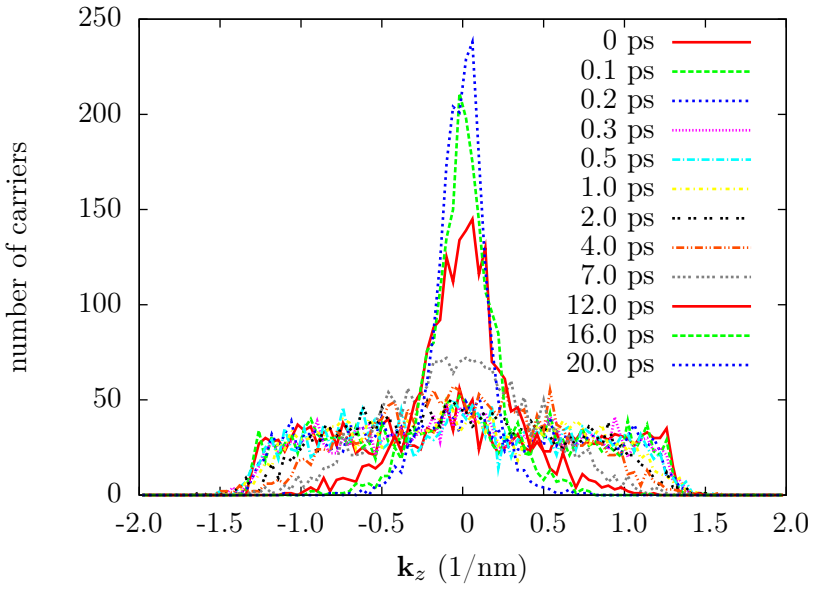


Figure 5.6: The $\mathbf{k}_z$-distribution for $Cd_{0.275}Hg_{0.725}Te$ as time progresses including carrier-carrier interactions and the Pauli principle. In this simulation there are 2000 superelectrons and $n = 10^{18}$ cm^{-3}.

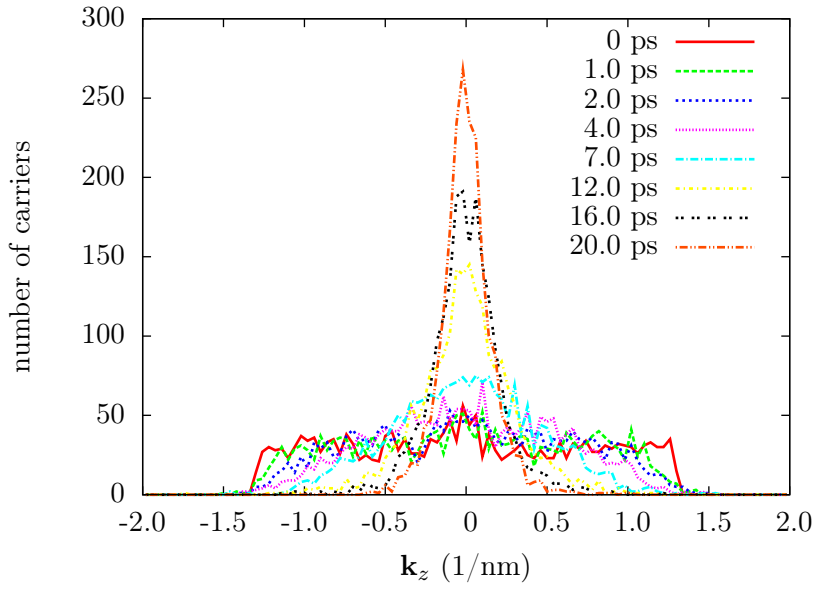


Figure 5.7: The $\mathbf{k}_z$-distribution for $Cd_{0.275}Hg_{0.725}Te$ as time progresses. In this simulation there is 2000 superelectrons and $n = 10^{18}$ cm^{-3}. In this plot, some of the time steps has been removed to clarify the effect.

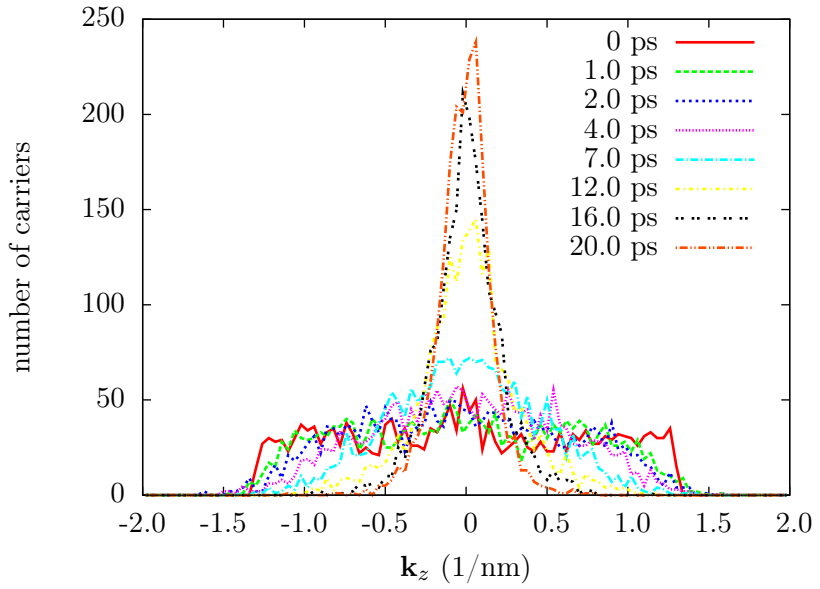


Figure 5.8: The $\mathbf{k}_z$-distribution for $Cd_{0.275}Hg_{0.725}Te$ as time progresses including carrier-carrier interactions and the Pauli principle. In this simulation there is 2000 superelectrons and $n = 10^{18}$ cm^{-3}. In this plot, some of the time steps has been removed to clarify the effect.

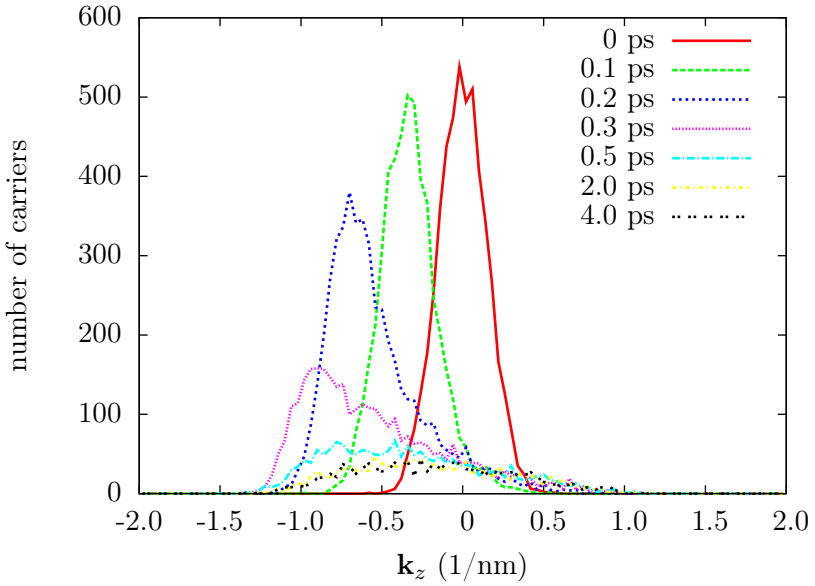


Figure 5.9: The $\mathbf{k}_z$-distribution for GaAs as time progresses. In this simulation there are 5000 superelectrons, $n_i = 1.84 \cdot 10^6$ cm^{-3}, and the external field is set to 25 kV/cm in the z direction.

5.1.2 Gaussian distribution with an applied field

In the program, the external field is applied along the z direction. In Figure 5.9 one can clearly see the shift in the $\mathbf{k}_z$-distribution. This happens very fast, but again, since the program allows the user to choose when to extract data, this can be observed and studied more closely. In these simulations, only the intrinsic carrier density has been used, i.e. no doping. Additionally, with such a strong field present, most of the carriers will leave the Γ valley. Looking at the data from the program an estimated 20% remain in the Γ at program termination.

In Figure 5.9, one can se the three initial peaks are quite large and shiftet. This indicates that the system requires some time to thermalize. From the plots, this time can be estimated to 0.4–0.5 ps. After this time, the distribution is warn and displaced. Studying Figure 5.10, it is clear that when the field strength increases, the shift in the $\mathbf{k}_z$ distribution also increases, as expected.

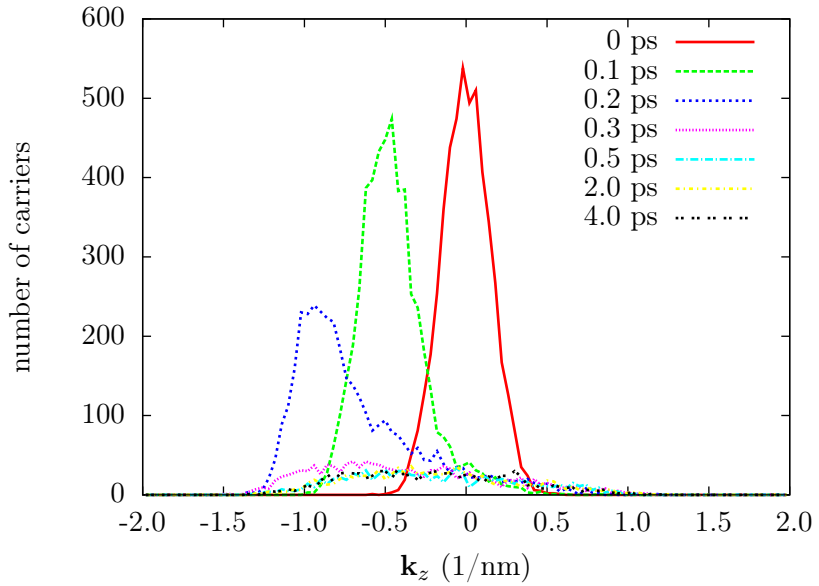


Figure 5.10: The $\mathbf{k}_z$-distribution for GaAs as time progresses. In this simulation there are 5000 superelectrons, $n_i = 1.84 \cdot 10^6$ cm^{-3}, and the external field is set to 35 kV/cm in the z direction.

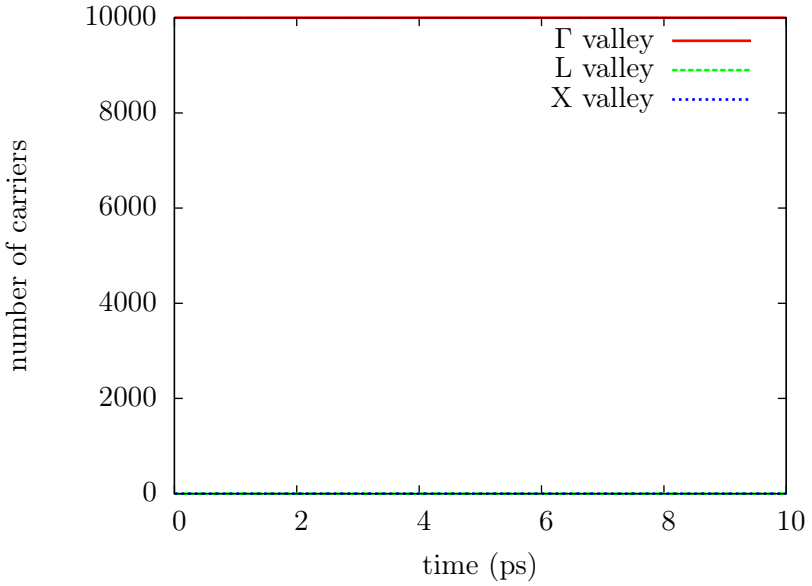


Figure 5.11: The electron distribution to the different valleys in GaAs with $n_i = 1.84 \cdot 10^6$ cm^{-3} without any external field applied.

5.1.3 Carrier distribution

In the following figures, the carrier distribution in the different valleys is shown as a function of time. In Figure 5.11, the simulations is run without any of the options turned on and only the intrinsic carrier density has been used, i.e. no doping. As can be seen, very litle happens here. Virtually every electron reside in the Γ valley and the diffusion to the other valleys are nearly non-present. Allthough the figure does not provide much information as is, a study of the datafiles confirms that there is some intervalley scattering occurring.

In Figure 5.12 an external field of strength 25 kV/cm is applied. The electrons now gain enough energy to "jump" to the higher lying valleys L and X. When the field is even stronger, the electron diffusion to the other valleys become more dominant, see Figure 5.13. The distribution displayed in Figures 5.12 and 5.13 is what should be expected. One may also notice that after an external field is applied, the system reaches steady state after about 2 ps, hence displaying plausible physical time constants.

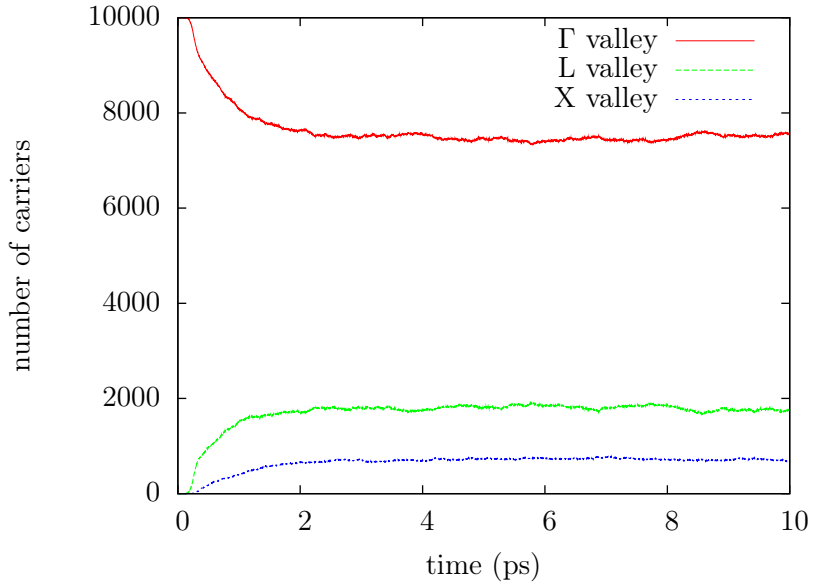


Figure 5.12: The electron distribution to the different valleys in GaAs with $n_i = 1.84 \cdot 10^6$ cm^{-3} during an external field of strength 25 kV/cm in the z direction.

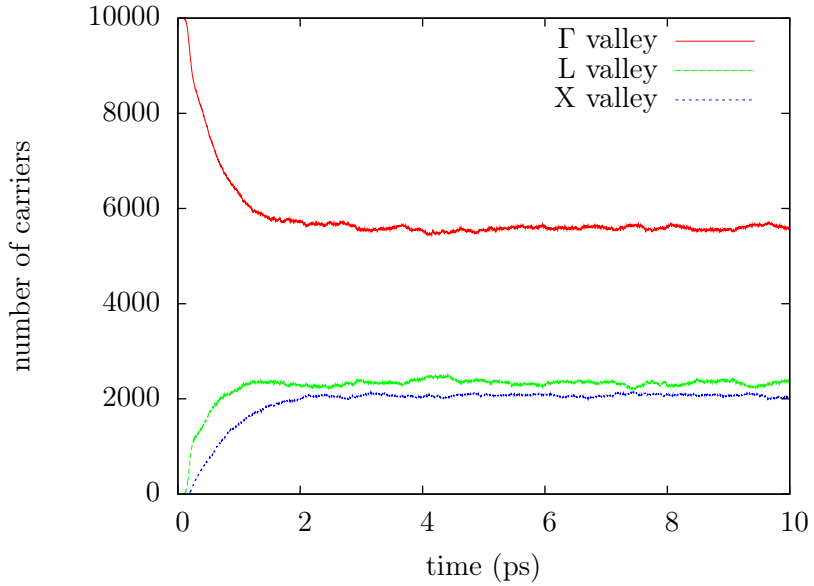


Figure 5.13: The electron distribution to the different valleys in GaAs with $n_i = 1.84 \cdot 10^6$ cm^{-3} during an external field of strength 35 kV/cm in the z direction.

5.2 Hot Phonon Results

In Figure 5.14, the hot phonon effect is displayed for two different elecron excitation levels, $n = 5{\cdot}10^{18}$ cm^{-3} and $n = 1{\cdot}10^{19}$ cm^{-3}, in $Cd_{0.275}Hg_{0.725}Te$. An equilibrium simulation has been added to the plot to show that the system cools to an appropriate level. The three optically stimulated simulations was stimulated with a laserbeam with $\lambda = 1064$ nm, corresponding to a Nd:YAG laser. It is assumed that 99% of the carriers becomes excited. This is due to the fact that the program handles integer input and that the program will crash if a 100% is selected.

As can be seen, there is a clear slowdown in the cooling of the system when hot phonon effects is taken into account. When the excited carrier concentration is $n = 5 \cdot 10^{18}$ cm^{-3}, the cooling does not suffer too bad. The production of hot phonons is not great enough to restrain the cooling of the system. When the excited carrier concentration is $n = 10^{19}$ cm^{-3}, the production of hot phonons is very high and the cooling of the system suffers very much. Without the hot phonon effect the system has reached equilibrium within ∼30 ps, while the low-excitation simulation requires approximately 25 ps longer to cool. The high-excitation simulation does not cool until 90–100 ps is reached which is a whopping ∼60 ps longer to cool down than without the hot phonon effect!

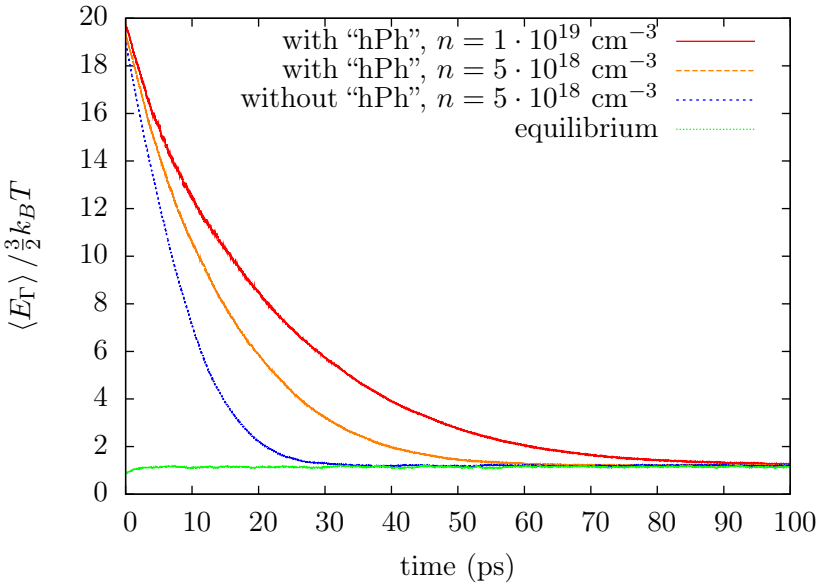


Figure 5.14: Cooling of $Cd_{0.275}Hg_{0.725}Te$ with and without hot phonon effects ("hPh"). The exciting laser has $\lambda = 1064$ nm and it is assumed that 95% of the carriers becomes excited. It is quite clear that with hot phonon effects turned on, the cooling of the system suffers.

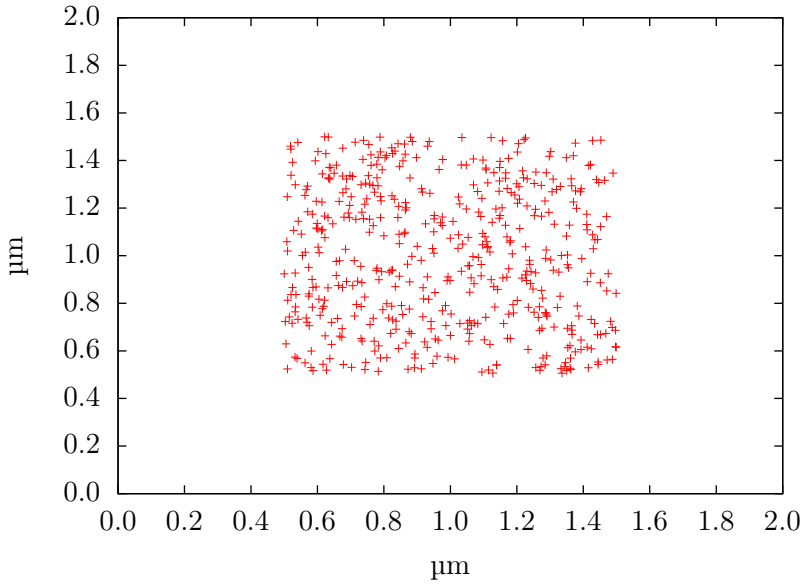


Figure 5.15: The diffusion of electrons without the Poisson solver. Initial spatial distribution.

5.3 Poisson Results

In the program, the carriers has been confined to a cube of 2 $\mu m \times 2\ \mu m \times 2\ \mu m$. The Poisson solver at hand is a 2D solver, hence the confinement in the third dimension is strictly not nescessary, but poses no problem either. The reason for the third "unnecessary" confinement is to be ready for a 3D solver. As mentioned previously, the framework has been built to be very flexible when it comes to swapping Poisson solvers.

Since the present Poisson solver only has been implemented as a "proof of concept", the presentation of data from it will be a bit scarce. Only the effect on the spatial carrier distribution is shown.

In the following, a series of plots show the spatial distribution of 500 superelectrons at four different times. The first four plots were simulated without the Poisson solver. The dispersion is purely diffusional in nature and the superelectrons do not fill the available space before 2.0 ps has passed. The last four plots show the spatial distribution when run with the Poisson solver. As can be seen from these plots, the superelectrons fill the space after 0.875 ps. This is roughly half the time of the run without the Poisson solver.

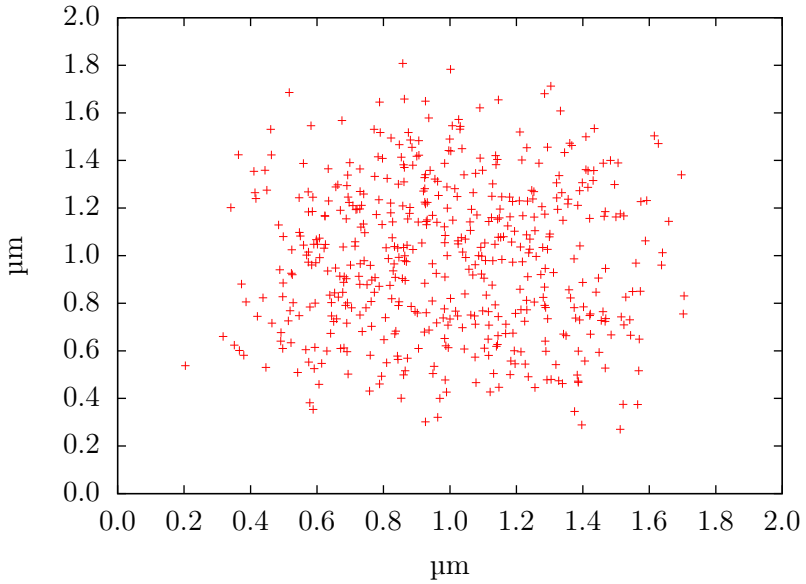


Figure 5.16: The diffusion of electrons without the Poisson solver. Spatial distribution after 0.50 ps.

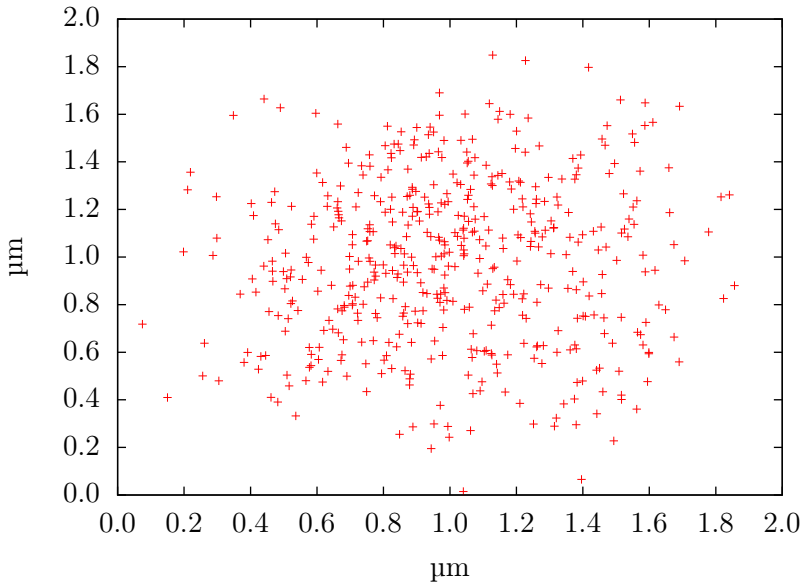


Figure 5.17: The diffusion of electrons without the Poisson solver. Spatial distribution after 0.875 ps.

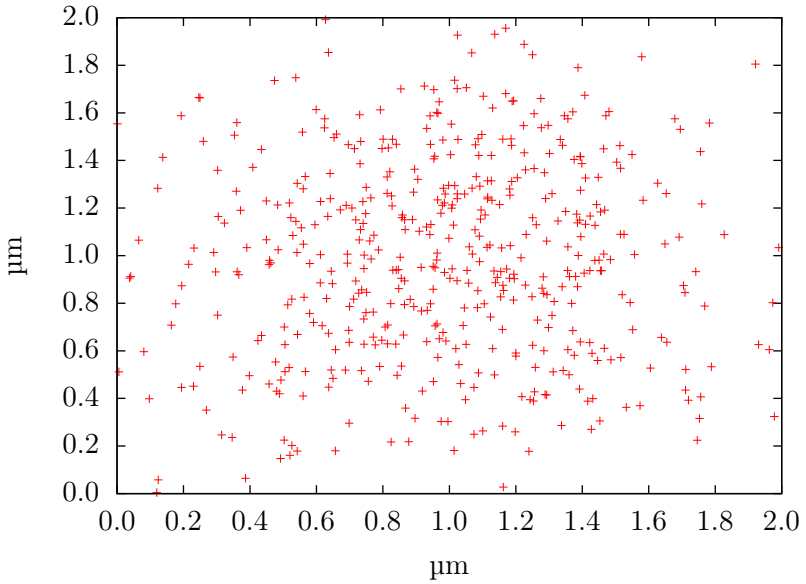


Figure 5.18: The diffusion of electrons without the Poisson solver. Spatial distribution after 2.00 ps.

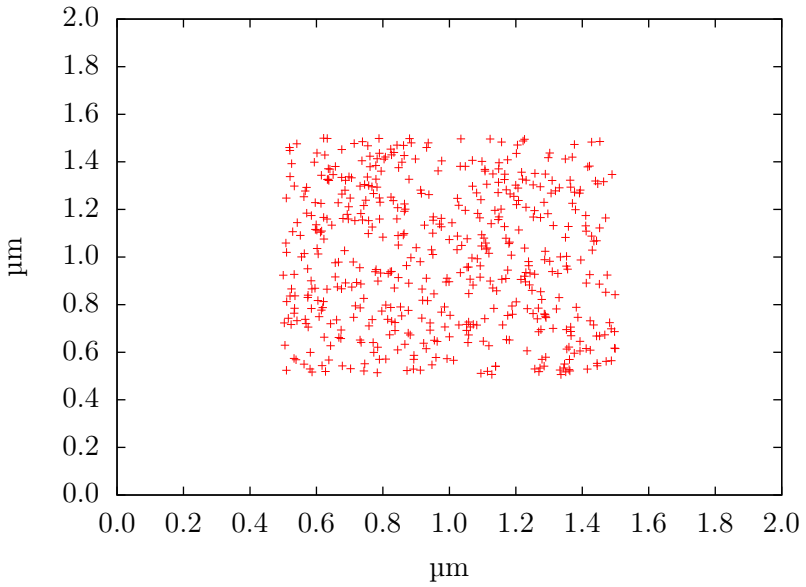


Figure 5.19: The diffusion of electrons with the Poisson solver on a mesh with 64 nodes. Initial spatial distributin

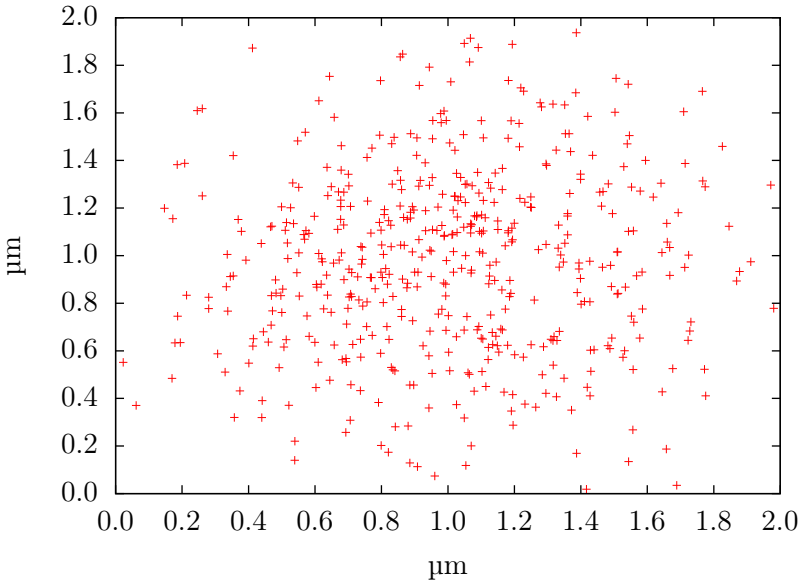


Figure 5.20: The diffusion of electrons with the Poisson solver on a mesh with 64 nodes. Spatial distribution after 0.50 ps.

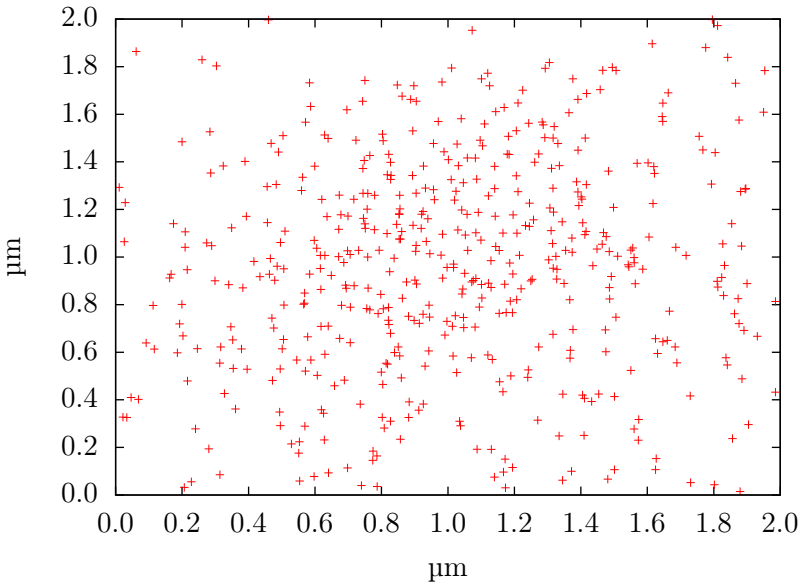


Figure 5.21: The diffusion of electrons with the Poisson solver on a mesh with 64 nodes. Spatial distribution after 0.875 ps.

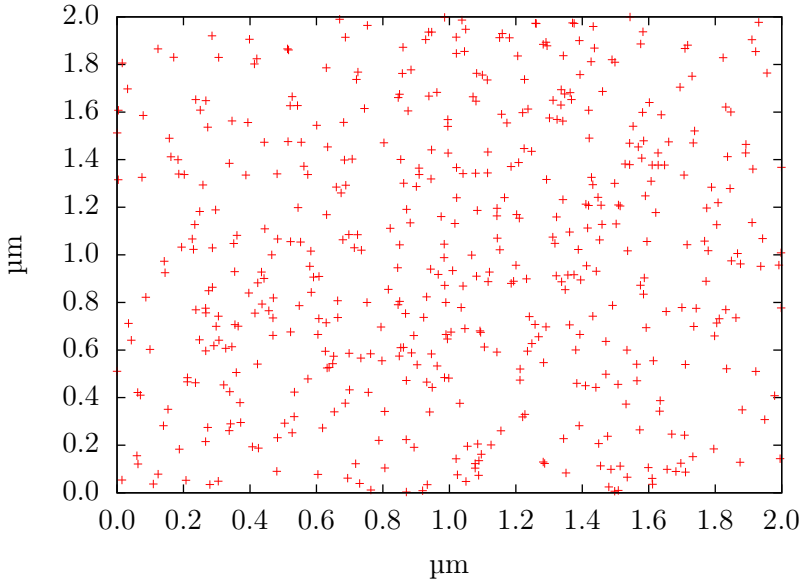


Figure 5.22: The diffusion of electrons with the Poisson solver on a mesh with 64 nodes. Spatial distribution after 2.0 ps

Chapter 6

Conclusion and Further Work

A program has been developed during two summer jobs, two project works, and two master works. During these 18 months a lot of improvements have been made and the program is now quite rich in features. The most important features are listed below.

- A "proof of concept" Poisson solver.
- The program is faster than ever, due to tabulated rates[5] and cleaning of the code.
- Hot phonon effects is implemented.
- Pauli principle is implemented.[23,24]
- More realistic bandstructures.[5]
- Tabulated scattering rates for significant speed-up.
- Several new scattering rates have been included, such as carrier-plasmon and alloy scatterings.
- A more thorough carrier-carrier scatterings has been implemented.[5]

The implemented scheme for the hot phonon effect show a clear slow-down of the cooling rate of a photoexcited system.

Solving the Poisson equation is very important in particle simulations, and a "proof of concept" solver has been implemented. The results show that the framework is working well and switching solver should be very easy.

Future plans for the program, with emphasis on the topics discussed in this thesis, can be summarized in list-form

- Make the program parallel and more CPU efficient.
- Create several new Poisson solvers for use in the program, both 2D and 3D, to enable simulation of novel semiconductor designs.
- Dynamic carrier ensembles to enable excitation and recombination effects.[25]
- Allow the lattice temperature to change when hot phonon effects has been chosen.
- A more CPU-efficient carrier-carrier routine should be made.[26]
- Implement the ability to select several semiconductor materials without recompiling.

If the reader is interested in the source code of our program, an e-mail can be sent to `ole.norum@gmail.com`

Appendix A

Materials

The program can simulate two different materials, $Cd_xHg_{1-x}Te$ and GaAs. The parameters of these are listed in section A.1, and the band structures are displayed in section A.2

A.1 Material Parameters

Technical difficulties has forced us to create two versions of the program, one for $Cd_xHg_{1-x}Te$ and another for GaAs. The parameters used in each of the versions are listed in table A.1 below. I will not include the natural constants, e.g. speed of light, fine structure constant, since these are widely available and should be equal everywhere. The natural constants used in the program has been taken from the Particle Data Group[27]

All parameters has been placed in `MODULE`-constructs, thus making them accessible from anywhere within the program. This creates a tidy structure in the code and updating constants is very easy.

A.2 Band structures

The two materials have quite different band structures. $Cd_xHg_{1-x}Te$ is a narrow gap semiconductor while GaAs have a much larger bandgap. The band structures are shown in Figures A.1 and A.2.

Table A.1: Material parameters for GaAs and $Cd_xHg_{1-x}Te$ (CMT) used in the program. All parameters are considered at $T = 300$ K.

Name	Constant	Value		Unit
		GaAs	CdHgTe	
High frequency dielectric constant$^{GaAs:[28] CMT:[29]}$	ε_∞	10.88	$15.2 - 15.6x + 8.2x^2$	
Low frequency dielectric constant$^{GaAs:[28] CMT:[29]}$	ε_s	12.85	$20.5 - 15.6x + 5.7x^2$	
Intrinsic electron density$^{GaAs:[1] CMT:[29]}$	n	$1.84 \cdot 10^6$	$10^{18.58-9.89*x}$	$\frac{1}{\mathrm{cm}^3}$
Mass density$^{GaAs:[28] CMT:[29]}$	ρ	5.317	$8.05 - 2.3x$	$\frac{\mathrm{g}}{\mathrm{cm}^3}$
Longitudinal sound velocity$^{GaAs:[30,31]}$	S	5240	4570	$\frac{m}{\mathrm{s}}$
Lattice constant$^{GaAs:[28] CMT:[29]}$	a	5.656	$6.477x + 6.490(1 - x)$	Å
Effective mass in Γ valley$^{GaAs:[1,32] CMT:[33]}$	m_Γ	0.067	$\frac{3\hbar^2 E_g}{(2 \cdot 8.28 \cdot 10^{-10})^2}$	$\frac{m^*}{m_0}$
Effective mass in L valley$^{GaAs:[30]}$	m_L	0.222	0.222^a	$\frac{m^*}{m_0}$
Effective mass in X valley$^{GaAs:[30]}$	m_X	0.580	0.580^a	$\frac{m^*}{m_0}$
Effective mass in HH band$^{GaAs:[32] CMT:[29]}$	m_HH	0.500	0.530	$\frac{m^*}{m_0}$
Effective mass in LH band$^{GaAs:[34] CMT:[33]}$	m_LH	0.074	$\frac{3\hbar^2 E_g}{(2 \cdot 8.28 \cdot 10^{-10})^2}$	$\frac{m^*}{m_0}$
Optical phonon frequency$^{GaAs:[28] CMT:[35]}$	ω_0	5.372	$\frac{2.92+2 \cdot 2.81}{3}$ [b]	$\frac{10^{13}}{s}$
Optical phonon lifetime$^{GaAs:[7]}$	τ_{ph}	7	1.5	ps
Nonparabolicty parameter in Γ valley$^{GaAs:[30,31] CMT:[16,33,36]}$	α_Γ	0.610	$(1 - \frac{m_\Gamma}{m_0})^2 / E_g$	$\frac{1}{\mathrm{eV}}$
Nonparabolicty parameter in L valley$^{GaAs:[30,31]}$	α_L	0.461	0.461^a	$\frac{1}{\mathrm{eV}}$
Nonparabolicty parameter in X valley$^{GaAs:[30,31]}$	α_X	0.204	0.204^a	$\frac{1}{\mathrm{eV}}$
Energy gap[c] $^{GaAs:[28] CMT:[37]}$	E_Γ	1.424	$E_{g0} + \frac{(6.3x_c - 3.25x - 5.92xx_c)10^{-4}T^2}{11x_c + 78.7x + T}$	eV
Valence band parameters$^{GaAs:[33,38]}$				
	A	7.98	7.98^a	
	B	5.16	5.16^a	
	C	6.56	6.56^a	

[a] Assumed equal to that of GaAs due to lack of parameters.

[b] The optical phonon frequency has been averaged over the two transverse and the longitudinal modes, since these are not distinguished in the program. Parameters taken at 80 K

[c] Where $E_{g0} = -0.303x_c + 1.606x - 0.132xx_c$ and $x_c = 1 - x$

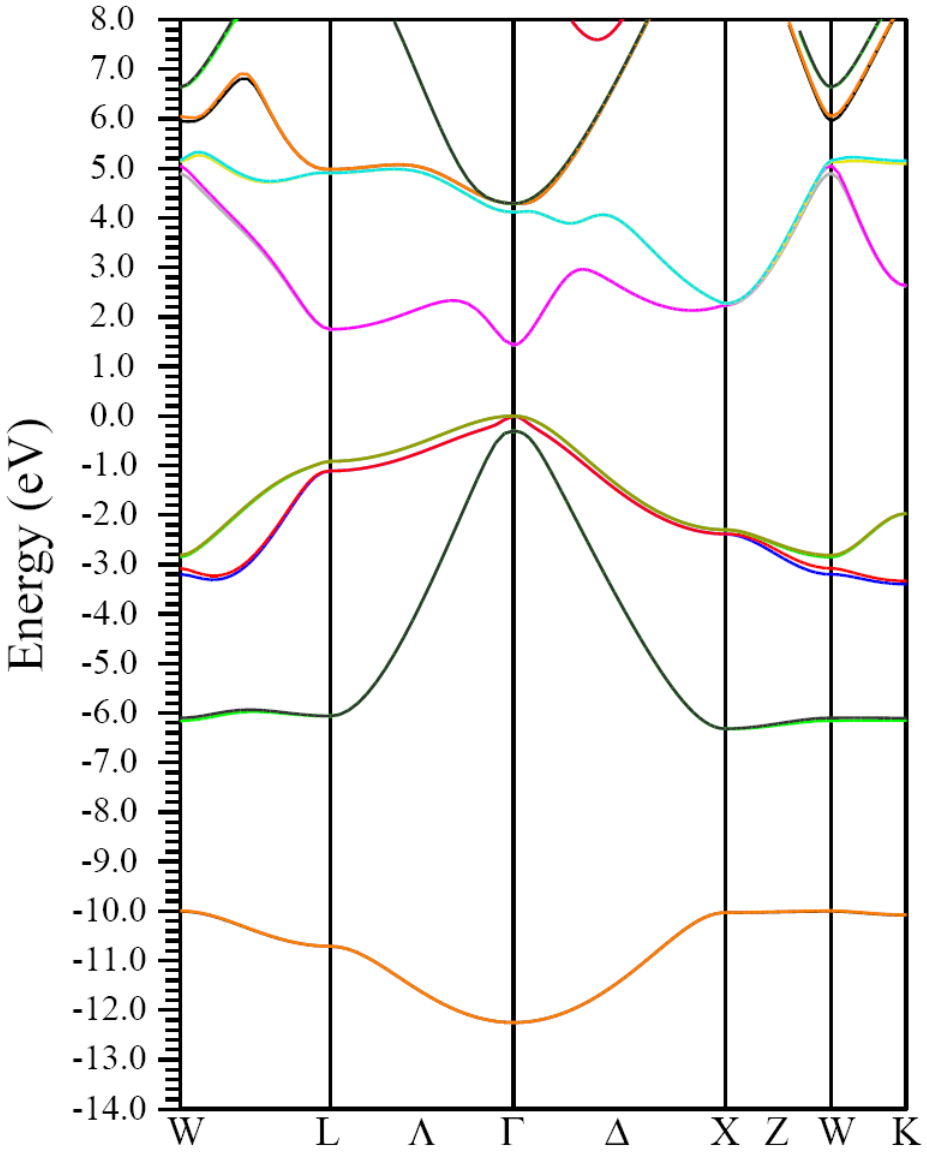


Figure A.1: The band structure of GaAs at T=300 K.

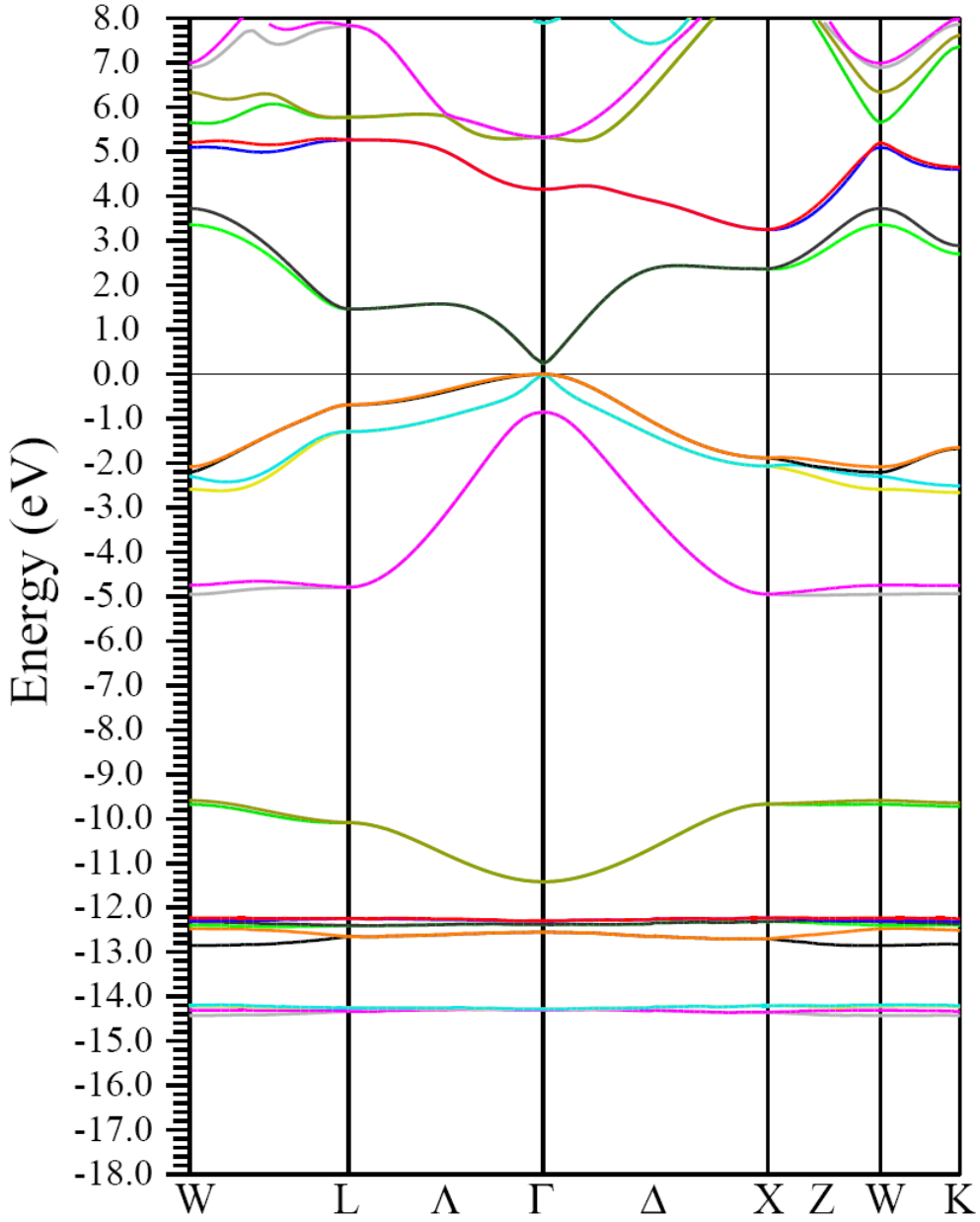


Figure A.2: The band structure of $Cd_{0.275}Hg_{0.725}Te$ at T=300 K.

Appendix B

Creating Results from the Program

In order to reproduce the results given throughout this thesis, the following program and scripts has been used. All plots has been made with Matlab or gnuplot.

B.1 Code to produce script input

The program presented here, "the reader", is developed to refine some of the output files produced from the main program. The reader is written in fortran 90/95 and requires no special compiler options. Any fortran 90/95 compiler will compile the presented code. The operation of the program is straight forward and well explained when executed.

The raw files containing information about the $\mathbf{k}$ and $\mathbf{k}_z$ distribution are made by looping through the ensemble and dumping the $\mathbf{k}$ and $\mathbf{k}_z$ for each valley into files. The reader takes these raw files as input and sorts the $\mathbf{k}$'s into a mesh producing files that are readable by MATLAB or gnuplot.

```
1   PROGRAM reader
2   IMPLICIT NONE
3   INTEGER,DIMENSION(12) :: n
4   INTEGER :: choice,selector,rerun
5   DOUBLE PRECISION :: kmin,kmax
6   CHARACTER(LEN=8),DIMENSION(10) :: filelist
7   CHARACTER(LEN=8) :: filename
8   n=0; choice=0; selector=0; rerun=0; kmin=0.; kmax=0;
9   filelist(1)='G.lst'
10  filelist(2)='L.lst'
11  filelist(3)='X.lst'
```

```
12      filelist(4)='HH.lst'
13      filelist(5)='LH.lst'
14      filelist(6)='Gz.lst'
15      filelist(7)='Lz.lst'
16      filelist(8)='Xz.lst'
17      filelist(9)='HHz.lst'
18      filelist(10)='LHz.lst'
19      WRITE(*,*)'Max k-vector'
20      READ(*,*)kmax
21      DO
22          WRITE(*,*)'What do you wish to examine?'
23          WRITE(*,*)'1: The entire picture'
24          WRITE(*,*)'2: The component parallel to the field'
25          READ(*,*)choice
26          IF(choice == 1) THEN
27              kmin=0
28          ELSE IF(choice == 2) THEN
29              kmin=-kmax
30          END IF
31          WRITE(*,*)'What valley/band would you like to examine?'
32          WRITE(*,*)'1: G valley'
33          WRITE(*,*)'2: L valley'
34          WRITE(*,*)'3: X valley'
35          WRITE(*,*)'4: HH band'
36          WRITE(*,*)'5: LH band'
37          READ(*,*)selector
38          IF(choice == 2) selector=selector+5
39          filename=filelist(selector)
40          WRITE(*,*)'In file ',filename,'you should find the number of'
41          IF(selector == 1 .OR. selector == 6) THEN
42              WRITE(*,*)'electrons in the G valley at 12 snapshots.'
43          ELSE IF(selector == 2 .OR. selector == 7) THEN
44              WRITE(*,*)'electrons in the L valley at 12 snapshots.'
45          ELSE IF(selector == 3 .OR. selector == 8) THEN
46              WRITE(*,*)'electrons in the X valley at 12 snapshots.'
47          ELSE IF(selector == 4 .OR. selector == 9) THEN
48              WRITE(*,*)'electrons in the HH band at 12 snapshots.'
49          ELSE IF(selector == 5 .OR. selector == 10) THEN
50              WRITE(*,*)'electrons in the LH band at 12 snapshots.'
51          END IF
52          WRITE(*,*)'Enter these below:'
53          WRITE(*,*)'Snapshot 1:'
54          READ(*,*)n(1)
55          WRITE(*,*)'Snaphsot 2:'
56          READ(*,*)n(2)
57          WRITE(*,*)'Snaphsot 3:'
58          READ(*,*)n(3)
59          WRITE(*,*)'Snaphsot 4:'
60          READ(*,*)n(4)
61          WRITE(*,*)'Snaphsot 5:'
62          READ(*,*)n(5)
63          WRITE(*,*)'Snaphsot 6:'
64          READ(*,*)n(6)
65          WRITE(*,*)'Snaphsot 7:'
66          READ(*,*)n(7)
67          WRITE(*,*)'Snaphsot 8:'
68          READ(*,*)n(8)
69          WRITE(*,*)'Snaphsot 9:'
70          READ(*,*)n(9)
71          WRITE(*,*)'Snaphsot 10:'
72          READ(*,*)n(10)
73          WRITE(*,*)'Snaphsot 11:'
```

```
74        READ(*,*)n(11)
75        WRITE(*,*)'Snaphsot 12:'
76        READ(*,*)n(12)
77  !     Here we call the printer which prints according to
          spesifications
78        CALL printer(n,kmax,kmin,filename,selector)
79        WRITE(*,*)'Do you wish to examine another valley/band?'
80        WRITE(*,*)'1: Yes'
81        WRITE(*,*)'0: No'
82        READ(*,*)rerun
83        IF(rerun == 0) EXIT
84     END DO
85     STOP
86     END PROGRAM reader
87     SUBROUTINE printer(n,kmax,kmin,filename,selector)
88     IMPLICIT NONE
89     INTEGER,INTENT(IN) :: selector
90     INTEGER,DIMENSION(12),INTENT(IN) :: n
91     DOUBLE PRECISION,INTENT(IN) :: kmax,kmin
92     CHARACTER(LEN=8),INTENT(IN) :: filename
93     INTEGER :: i,j,out,meshsize
94     DOUBLE PRECISION,DIMENSION(100,12) :: mesh
95     DOUBLE PRECISION :: k,step
96     CHARACTER(LEN=13),DIMENSION(10,12) :: meshname
97     out=0; meshsize=100; mesh=0.; k=0.;
98     step=(kmax-kmin)/meshsize
99     meshname(1,1)='gmesh1.lst'
100    meshname(1,2)='gmesh2.lst'
101    meshname(1,3)='gmesh3.lst'
102    meshname(1,4)='gmesh4.lst'
103    meshname(1,5)='gmesh5.lst'
104    meshname(1,6)='gmesh6.lst'
105    meshname(1,7)='gmesh7.lst'
106    meshname(1,8)='gmesh8.lst'
107    meshname(1,9)='gmesh9.lst'
108    meshname(1,10)='gmesh10.lst'
109    meshname(1,11)='gmesh11.lst'
110    meshname(1,12)='gmesh12.lst'
111    meshname(2,1)='lmesh1.lst'
112    meshname(2,2)='lmesh2.lst'
113    meshname(2,3)='lmesh3.lst'
114    meshname(2,4)='lmesh4.lst'
115    meshname(2,5)='lmesh5.lst'
116    meshname(2,6)='lmesh6.lst'
117    meshname(2,7)='lmesh7.lst'
118    meshname(2,8)='lmesh8.lst'
119    meshname(2,9)='lmesh9.lst'
120    meshname(2,10)='lmesh10.lst'
121    meshname(2,11)='lmesh11.lst'
122    meshname(2,12)='lmesh12.lst'
123    meshname(3,1)='xmesh1.lst'
124    meshname(3,2)='xmesh2.lst'
125    meshname(3,3)='xmesh3.lst'
126    meshname(3,4)='xmesh4.lst'
127    meshname(3,5)='xmesh5.lst'
128    meshname(3,6)='xmesh6.lst'
129    meshname(3,7)='xmesh7.lst'
130    meshname(3,8)='xmesh8.lst'
131    meshname(3,9)='xmesh9.lst'
132    meshname(3,10)='xmesh10.lst'
133    meshname(3,11)='xmesh11.lst'
134    meshname(3,12)='xmesh12.lst'
```

```
135     meshname(4,1)='hhmesh1.lst'
136     meshname(4,2)='hhmesh2.lst'
137     meshname(4,3)='hhmesh3.lst'
138     meshname(4,4)='hhmesh4.lst'
139     meshname(4,5)='hhmesh5.lst'
140     meshname(4,6)='hhmesh6.lst'
141     meshname(4,7)='hhmesh7.lst'
142     meshname(4,8)='hhmesh8.lst'
143     meshname(4,9)='hhmesh9.lst'
144     meshname(4,10)='hhmesh10.lst'
145     meshname(4,11)='hhmesh11.lst'
146     meshname(4,12)='hhmesh12.lst'
147     meshname(5,1)='lhmesh1.lst'
148     meshname(5,2)='lhmesh2.lst'
149     meshname(5,3)='lhmesh3.lst'
150     meshname(5,4)='lhmesh4.lst'
151     meshname(5,5)='lhmesh5.lst'
152     meshname(5,6)='lhmesh6.lst'
153     meshname(5,7)='lhmesh7.lst'
154     meshname(5,8)='lhmesh8.lst'
155     meshname(5,9)='lhmesh9.lst'
156     meshname(5,10)='lhmesh10.lst'
157     meshname(5,11)='lhmesh11.lst'
158     meshname(5,12)='lhmesh12.lst'
159     meshname(6,1)='gzmesh1.lst'
160     meshname(6,2)='gzmesh2.lst'
161     meshname(6,3)='gzmesh3.lst'
162     meshname(6,4)='gzmesh4.lst'
163     meshname(6,5)='gzmesh5.lst'
164     meshname(6,6)='gzmesh6.lst'
165     meshname(6,7)='gzmesh7.lst'
166     meshname(6,8)='gzmesh8.lst'
167     meshname(6,9)='gzmesh9.lst'
168     meshname(6,10)='gzmesh10.lst'
169     meshname(6,11)='gzmesh11.lst'
170     meshname(6,12)='gzmesh12.lst'
171     meshname(7,1)='lzmesh1.lst'
172     meshname(7,2)='lzmesh2.lst'
173     meshname(7,3)='lzmesh3.lst'
174     meshname(7,4)='lzmesh4.lst'
175     meshname(7,5)='lzmesh5.lst'
176     meshname(7,6)='lzmesh6.lst'
177     meshname(7,7)='lzmesh7.lst'
178     meshname(7,8)='lzmesh8.lst'
179     meshname(7,9)='lzmesh9.lst'
180     meshname(7,10)='lzmesh10.lst'
181     meshname(7,11)='lzmesh11.lst'
182     meshname(7,12)='lzmesh12.lst'
183     meshname(8,1)='xzmesh1.lst'
184     meshname(8,2)='xzmesh2.lst'
185     meshname(8,3)='xzmesh3.lst'
186     meshname(8,4)='xzmesh4.lst'
187     meshname(8,5)='xzmesh5.lst'
188     meshname(8,6)='xzmesh6.lst'
189     meshname(8,7)='xzmesh7.lst'
190     meshname(8,8)='xzmesh8.lst'
191     meshname(8,9)='xzmesh9.lst'
192     meshname(8,10)='xzmesh10.lst'
193     meshname(8,11)='xzmesh11.lst'
194     meshname(8,12)='xzmesh12.lst'
195     meshname(9,1)='hhzmesh1.lst'
196     meshname(9,2)='hhzmesh2.lst'
```

```
197     meshname(9,3)='hhzmesh3.lst'
198     meshname(9,4)='hhzmesh4.lst'
199     meshname(9,5)='hhzmesh5.lst'
200     meshname(9,6)='hhzmesh6.lst'
201     meshname(9,7)='hhzmesh7.lst'
202     meshname(9,8)='hhzmesh8.lst'
203     meshname(9,9)='hhzmesh9.lst'
204     meshname(9,10)='hhzmesh10.lst'
205     meshname(9,11)='hhzmesh11.lst'
206     meshname(9,12)='hhzmesh12.lst'
207     meshname(10,1)='lhzmesh1.lst'
208     meshname(10,2)='lhzmesh2.lst'
209     meshname(10,3)='lhzmesh3.lst'
210     meshname(10,4)='lhzmesh4.lst'
211     meshname(10,5)='lhzmesh5.lst'
212     meshname(10,6)='lhzmesh6.lst'
213     meshname(10,7)='lhzmesh7.lst'
214     meshname(10,8)='lhzmesh8.lst'
215     meshname(10,9)='lhzmesh9.lst'
216     meshname(10,10)='lhzmesh10.lst'
217     meshname(10,11)='lhzmesh11.lst'
218     meshname(10,12)='lhzmesh12.lst'
219     OPEN(11,FILE=filename,ACCESS='SEQUENTIAL',STATUS='OLD')
220     DO i=1,n(1)
221        READ(11,*)k
222        IF(ABS(k) > kmax) out=out+1
223        DO j=1,meshsize
224           IF((k >= (kmin+step*(j-1)) .AND. (k < (kmin+step*j)))) THEN
225              mesh(j,1)=mesh(j,1)+1
226              EXIT
227           END IF
228        END DO
229     END DO
230     DO i=1,n(2)
231        READ(11,*)k
232        IF(ABS(k) > kmax) out=out+1
233        DO j=1,meshsize
234           IF((k >= (kmin+step*(j-1)) .AND. (k < (kmin+step*j)))) THEN
235              mesh(j,2)=mesh(j,2)+1
236              EXIT
237           END IF
238        END DO
239     END DO
240     DO i=1,n(3)
241        READ(11,*)k
242        IF(ABS(k) > kmax) out=out+1
243        DO j=1,meshsize
244           IF((k >= (kmin+step*(j-1)) .AND. (k < (kmin+step*j)))) THEN
245              mesh(j,3)=mesh(j,3)+1
246              EXIT
247           END IF
248        END DO
249     END DO
250     DO i=1,n(4)
251        READ(11,*)k
252        IF(ABS(k) > kmax) out=out+1
253        DO j=1,meshsize
254           IF((k >= (kmin+step*(j-1)) .AND. (k < (kmin+step*j)))) THEN
255              mesh(j,4)=mesh(j,4)+1
256              EXIT
257           END IF
258        END DO
```

```
259     END DO
260     DO i=1,n(5)
261         READ(11,*)k
262         IF(ABS(k) > kmax) out=out+1
263         DO j=1,meshsize
264             IF((k >= (kmin+step*(j-1)) .AND. (k < (kmin+step*j)))) THEN
265                 mesh(j,5)=mesh(j,5)+1
266                 EXIT
267             END IF
268         END DO
269     END DO
270     DO i=1,n(6)
271         READ(11,*)k
272         IF(ABS(k) > kmax) out=out+1
273         DO j=1,meshsize
274             IF((k >= (kmin+step*(j-1)) .AND. (k < (kmin+step*j)))) THEN
275                 mesh(j,6)=mesh(j,6)+1
276                 EXIT
277             END IF
278         END DO
279     END DO
280     DO i=1,n(7)
281         READ(11,*)k
282         IF(ABS(k) > kmax) out=out+1
283         DO j=1,meshsize
284             IF((k >= (kmin+step*(j-1)) .AND. (k < (kmin+step*j)))) THEN
285                 mesh(j,7)=mesh(j,7)+1
286                 EXIT
287             END IF
288         END DO
289     END DO
290     DO i=1,n(8)
291         READ(11,*)k
292         IF(ABS(k) > kmax) out=out+1
293         DO j=1,meshsize
294             IF((k >= (kmin+step*(j-1)) .AND. (k < (kmin+step*j)))) THEN
295                 mesh(j,8)=mesh(j,8)+1
296                 EXIT
297             END IF
298         END DO
299     END DO
300     DO i=1,n(9)
301         READ(11,*)k
302         IF(ABS(k) > kmax) out=out+1
303         DO j=1,meshsize
304             IF((k >= (kmin+step*(j-1)) .AND. (k < (kmin+step*j)))) THEN
305                 mesh(j,9)=mesh(j,9)+1
306                 EXIT
307             END IF
308         END DO
309     END DO
310     DO i=1,n(10)
311         READ(11,*)k
312         IF(ABS(k) > kmax) out=out+1
313         DO j=1,meshsize
314             IF((k >= (kmin+step*(j-1)) .AND. (k < (kmin+step*j)))) THEN
315                 mesh(j,10)=mesh(j,10)+1
316                 EXIT
317             END IF
318         END DO
319     END DO
320     DO i=1,n(11)
```

```
321         READ(11,*)k
322         IF(ABS(k) > kmax) out=out+1
323         DO j=1,meshsize
324             IF((k >= (kmin+step*(j-1)) .AND. (k < (kmin+step*j)))) THEN
325                 mesh(j,11)=mesh(j,11)+1
326                 EXIT
327             END IF
328         END DO
329     END DO
330     DO i=1,n(12)
331         READ(11,*)k
332         IF(ABS(k) > kmax) out=out+1
333         DO j=1,meshsize
334             IF((k >= (kmin+step*(j-1)) .AND. (k < (kmin+step*j)))) THEN
335                 mesh(j,12)=mesh(j,12)+1
336                 EXIT
337             END IF
338         END DO
339     END DO
340     CLOSE(11)
341     OPEN(21,FILE=meshname(selector,1),ACCESS='SEQUENTIAL',STATUS='
            REPLACE')
342     DO i=1,meshsize
343         WRITE(21,*)(kmin+step*(i-0.5)),mesh(i,1)
344     END DO
345     CLOSE(21)
346     OPEN(22,FILE=meshname(selector,2),ACCESS='SEQUENTIAL',STATUS='
            REPLACE')
347     DO i=1,meshsize
348         WRITE(22,*)(kmin+step*(i-0.5)),mesh(i,2)
349     END DO
350     CLOSE(22)
351     OPEN(23,FILE=meshname(selector,3),ACCESS='SEQUENTIAL',STATUS='
            REPLACE')
352     DO i=1,meshsize
353         WRITE(23,*)(kmin+step*(i-0.5)),mesh(i,3)
354     END DO
355     CLOSE(23)
356     OPEN(24,FILE=meshname(selector,4),ACCESS='SEQUENTIAL',STATUS='
            REPLACE')
357     DO i=1,meshsize
358         WRITE(24,*)(kmin+step*(i-0.5)),mesh(i,4)
359     END DO
360     CLOSE(24)
361     OPEN(25,FILE=meshname(selector,5),ACCESS='SEQUENTIAL',STATUS='
            REPLACE')
362     DO i=1,meshsize
363         WRITE(25,*)(kmin+step*(i-0.5)),mesh(i,5)
364     END DO
365     CLOSE(25)
366     OPEN(26,FILE=meshname(selector,6),ACCESS='SEQUENTIAL',STATUS='
            REPLACE')
367     DO i=1,meshsize
368         WRITE(26,*)(kmin+step*(i-0.5)),mesh(i,6)
369     END DO
370     CLOSE(26)
371     OPEN(27,FILE=meshname(selector,7),ACCESS='SEQUENTIAL',STATUS='
            REPLACE')
372     DO i=1,meshsize
373         WRITE(27,*)(kmin+step*(i-0.5)),mesh(i,7)
374     END DO
375     CLOSE(27)
```

```
376     OPEN(28,FILE=meshname(selector,8),ACCESS='SEQUENTIAL',STATUS='
            REPLACE')
377     DO i=1,meshsize
378         WRITE(28,*)(kmin+step*(i-0.5)),mesh(i,8)
379     END DO
380     CLOSE(28)
381     OPEN(29,FILE=meshname(selector,9),ACCESS='SEQUENTIAL',STATUS='
            REPLACE')
382     DO i=1,meshsize
383         WRITE(29,*)(kmin+step*(i-0.5)),mesh(i,9)
384     END DO
385     CLOSE(29)
386     OPEN(30,FILE=meshname(selector,10),ACCESS='SEQUENTIAL',STATUS='
            REPLACE')
387     DO i=1,meshsize
388         WRITE(30,*)(kmin+step*(i-0.5)),mesh(i,10)
389     END DO
390     CLOSE(30)
391     OPEN(31,FILE=meshname(selector,11),ACCESS='SEQUENTIAL',STATUS='
            REPLACE')
392     DO i=1,meshsize
393         WRITE(31,*)(kmin+step*(i-0.5)),mesh(i,11)
394     END DO
395     CLOSE(31)
396     OPEN(32,FILE=meshname(selector,12),ACCESS='SEQUENTIAL',STATUS='
            REPLACE')
397     DO i=1,meshsize
398         WRITE(32,*)(kmin+step*(i-0.5)),mesh(i,12)
399     END DO
400     CLOSE(32)
401     WRITE(*,*)'Carriers outside k-mesh: ',out
402     WRITE(*,*)'Data written to files: ',meshname(selector,:)
403     RETURN
404     END SUBROUTINE printer
```

B.2 Scripts to produce figures

Some of the MATLAB scripts to reproduce the graphs presented in the report are given here. Additionally, gnuplot files are also presented for those preferring free software. The following are just some example files as you will need to change the input files according to what the user is interested in. The first two will reproduce the Figures 5.1 to 5.3. To reproduce the Figures 5.9 and 5.10, just change the input files from `gmeshx.lst` to `gzmeshx.lst`, where x is a number from 1 to 12. The next to scripts will reproduce the Figures 5.11 to 5.13.

```
clear all;
close all;
clc;

load gmesh1.lst;
load gmesh2.lst;
load gmesh3.lst;
load gmesh4.lst;
load gmesh5.lst;
load gmesh6.lst;
load gmesh7.lst;
load gmesh8.lst;
load gmesh9.lst;
load gmesh10.lst;
load gmesh11.lst;
load gmesh12.lst;
k=gmesh1(:,1);
n1=gmesh1(:,2);
n2=gmesh2(:,2);
n3=gmesh3(:,2);
n4=gmesh4(:,2);
n5=gmesh5(:,2);
n6=gmesh6(:,2);
n7=gmesh7(:,2);
n8=gmesh8(:,2);
n9=gmesh9(:,2);
n10=gmesh10(:,2);
n11=gmesh11(:,2);
n12=gmesh12(:,2);

% o- give circles, or- gice red circles, s- give squares
% r=red, g=green, b=blue, c=cyan, m=mangenta, y=yellow, k=black, w=white
plot(k,n1,'k-',k,n2,'b-',k,n3,'g-',k,n4,'y',k,n5,'c-',k,n6,'m-',k,n7,'
    b-.',k,n8,'g-.',k,n9,'y-.',k,n10,'r-.',k,n11,'k-.',k,n12,'b-');
legend('0 ps','1.5 ps','3.0 ps','4.5 ps','6.0 ps','7.5 ps','9.0 ps','
    10.5 ps','12.0 ps','18.0 ps','24.0 ps','30.0 ps');

title('Carrier distribution in GaAs \Gamma-valley, 500 electrons in
    ensemble');
xlabel('Wavevector |k| (1/m)');
ylabel('Number of carriers');
```

```
set terminal epslatex color
# set autoscale
set size 0.9,0.9
set output 'k.eps'
set ylabel "number of carriers"
set xlabel "k (1/nm)"
# set title 'Carrier distribution in GaAs $\Gamma$ valley after
    optical stimuli, 2000 electrons in ensemble'
plot "gmesh1.lst" title '0 ps' with lines lw 4,\
"gmesh2.lst" title '0.1 ps' with lines lw 4,\
"gmesh3.lst" title '0.2 ps' with lines lw 4,\
"gmesh4.lst" title '0.3 ps' with lines lw 4,\
"gmesh5.lst" title '0.5 ps' with lines lw 4,\
"gmesh6.lst" title '1.0 ps' with lines lw 4,\
"gmesh7.lst" title '2.0 ps' with lines lw 4,\
"gmesh8.lst" title '4.0 ps' with lines lw 4,\
"gmesh9.lst" title '7.0 ps' with lines lw 4,\
"gmesh10.lst" title '12.0 ps' with lines lw 4,\
"gmesh11.lst" title '16.0 ps' with lines lw 4,\
"gmesh12.lst" title '20.0 ps' with lines lw 4
```

```
clear all;
close all;
clc;

load gdist.lst;
load ldist.lst;
load xdist.lst;
n1=gdist(:,1);
n2=ldist(:,1);
n3=xdist(:,1);

% o- give circles, or- gice red circles, s- give squares
% r=red, g=green, b=blue, c=cyan, m=mangenta, y=yellow, k=black, w=white
plot(n1,'k-');
hold all;
plot(n2,'b-');
plot(n3,'g-');
legend('G','L','X');

title('Number of electrons in valleys');
xlabel('Time (10 fs)');
ylabel('Number of carriers');
```

```
set terminal epslatex color
# set autoscale
set size 0.9,0.9
set output 'dist.eps'
set ylabel "number of carriers"
set xlabel "time (ps)"
# set title 'Electron distribution in GaAs, 5000 electrons in ensemble
    , with 25kV/cm external field'
plot "Gdist.lst" title '$\Gamma$ valley' with lines lw 4,\
"Ldist.lst" title 'L valley' with lines lw 4,\
"Xdist.lst" title 'X valley' with lines lw 4
```

In order to reproduce the Poisson plots presented in Section 5.3, the following scripts may be used. If the user chooses the MATLAB script, the user must manually remove the bottom line, containing text, from the output files `ex.lst` and `ez.lst`. The last script will produce a movie in MATLAB, the author has not made an equivalent for gnuplot in this particular case.

```
clear all;
close all;
clc;

xt=load('pois64/ex.txt');
zt=load('pois64/ez.txt');
x=xt(1:1000);
z=zt(1:1000);
scatter(x,z,'.');
xlim([0 0.000002]);
ylim([0 0.000002]);
```

```
set terminal epslatex color
# set autoscale
set size 0.5,0.5
set xrange [0:2e-6]
set yrange [0:2e-6]
set ylabel "mu m"
set xlabel "mu m"
# set title 'xz'
set output 'xz1.eps'
plot "< paste ex.lst ez.lst" every ::0::999 with points
set output 'xz2.eps'
plot "< paste ex.lst ez.lst" every ::1000::1999 with points
set output 'xz3.eps'
plot "< paste ex.lst ez.lst" every ::2000::2999 with points
set output 'xz4.eps'
plot "< paste ex.lst ez.lst" every ::3000::3999 with points
```

```
set output 'xz5.eps'
plot "< paste ex.lst ez.lst" every ::4000::4999 with points
set output 'xz6.eps'
plot "< paste ex.lst ez.lst" every ::5000::5999 with points
set output 'xz7.eps'
plot "< paste ex.lst ez.lst" every ::6000::6999 with points
set output 'xz8.eps'
plot "< paste ex.lst ez.lst" every ::7000::7999 with points
set output 'xz9.eps'
plot "< paste ex.lst ez.lst" every ::8000::8999 with points
set output 'xz10.eps'
plot "< paste ex.lst ez.lst" every ::9000::9999 with points
set output 'xz11.eps'
plot "< paste ex.lst ez.lst" every ::10000::10999 with points
set output 'xz12.eps'
plot "< paste ex.lst ez.lst" every ::11000::11999 with points
```

```
clear all;
close all;
clc;
%number of gridpoints-1
n=127;
%number of timesteps
t=12; %12 for f.lst, 11 for u.lst and p.lst
A=load('f.lst'); % change to proper file, f.lst, u.lst or p.lst
for j=1:t-1
    Az=A(n*n*j+1 :n*n*(j+1),1);
    Az=reshape(Az,n,n);
    surf(1:n, 1:n, Az); %mesh(),surf()
    %axis([0 128 0 128 -1 1]); %-4.e19 4.e19]);
    view(0,90); %view(70,70); view(0,90);
    F(j*3-2)=getframe;
    F(j*3-1)=getframe;
    F(j*3)=getframe;
end
movie2avi(F,'poisson.avi')
```

The script for producing the figure in Section 5.2 is only available for gnuplot.

```
set terminal epslatex color
# set autoscale
set size 0.9,0.9
set output 'hot.eps'
set ylabel "E/3kBT/2"
set xlabel "time (ps)"
# set title 'Electron distribution in GaAs, 5000 electrons in ensemble
    , with 25kV/cm external field'
plot "cmt25/energydistG.lst" title 'with, n=19' with lines lw 2,\
"cmt24/energydistG.lst" title 'with, n=18' with lines lw 2,\
"cmtno24/energydistG.lst" title 'without, n=18' with lines lw 2,\
"cmtmax/energydistG.lst" title 'equilibrium' with lines lw 2
```

References

[1] Jasprit Singh. *Electronic and Optoelectronic Properties of Semiconductor Structures*. Cambridge University Press, first paperback edition, 2007.

[2] M.A. Kinch, J.D. Beck, C.-F. Wan, and J. Campbell. HgCdTe Electron Avalanche Photodiodes. *Journal of Electronic Materials*, 33(6):630–639, 2004.

[3] Ole Christian Norum. *A Review of Poisson Solving Techniques for Monte Carlo Simulation of Carrier Dynamics in Semiconductors*. Norwegian University of Science and Technology, December 2008.

[4] Øyvind Olsen. *Modelling of Hole Scattering in a Monte Carlo Transport Kernel for Semiconductors*. Norwegian University of Science and Technology, December 2008.

[5] Øyvind Olsen. Construction of a Transport Kernel for an Ensemble Monte Carlo Simulator. Master's thesis, June 2009.

[6] Jagdeep Shah. *Ultrafast Spectroscopy of Semiconductors and Semiconductor Devices*. Springer, second enlarged edition, 1998.

[7] P. Lugli, C. Jacoboni, and L. Reggiani. Monte Carlo algorithm for hot phonons in polar semiconductors. *Applied Physics Letters*, 50(18): 1251–1253, 1987.

[8] Eric Pop, Sanjiv Sinha, and Kenneth E. Goodson. Heat Generation and Transport in Nanometer-Scale Transistors. *Proceedings of the IEEE*, 94 (8):1587–1601, 2006.

[9] Sandip Mazumder and Arunava Majumdar. Monte Carlo Study of Phonon Transport in Solid Thin Films Including Dispersion and Polarization. *Journal of Heat Transfer*, 123:749–759, 2001.

[10] M. Rieger, P. Kocevar, P. Lugli, P. Bordone, L. Reggiani, and S.M. Goodnick. Monte Carlo studies of nonequilibrium phonon effects in

polar semiconductors and quantum wells. II. Non-Ohmic transport in n-type gallium arsenide. *Physical review B*, 39(11):7866–7875, 1988.

[11] A.K. Storebø, T. Brudevoll, O. Olsen, O.C. Norum, and M. Breivik. *Energy relaxation in IR laser excited $Hg_{1-x}Cd_xTe$.* Accepted for presentation in EDISON 16, 2009.

[12] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in Fortran 77: The Art of Scientific Computing*, volume 1 of *Fortran Numerical Recipes*. Cambridge University Press, second edition, 2006.

[13] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in Fortran 90: The Art of Parallel Scientific Computing*, volume 2 of *Fortran Numerical Recipes*. Cambridge University Press, second edition, 1999.

[14] Einar Halvorsen. Numerical calculation of valence band structure and hole scattering rates in GaAs, February 1991.

[15] S. Krishnamurthy, M.A. Berding, Z.G. Yu, C.H. Swartz, T.H. Myers, D.D. Edwall, and R. DeWames. Model for Minority Carrier Lifetimes in Doped HgCdTe. *Journal of Electronic Materials*, 34(6):873–879, 2005.

[16] Geir Uri Jensen. *Monte Carlo simulation of III-V semiconductor devices.* PhD thesis, The Norwegian Institute of Technology, June 1989.

[17] R.W. Hockney. A fast direct solution of Poisson's equation using Fourier analysis. *Journal of the Association of Computing Machinery*, 12:95–113, 1965.

[18] R.W. Hockney and J.W. Eastwood. *Computer Simulation Using Particles.* Adam Hilger, special student edition, 1988.

[19] L. Greengard and V. Rokhlin. A Fast Algorithm for Particle Simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.

[20] Rick Beatson and Leslie Greengard. A short course on fast multipole methods. URL `http://www.math.nyu.edu/faculty/greengar/shortcourse_fmm.pdf`.

[21] L. Greengard and V. Rokhlin. A new version of the Fast Multipole Method for the Laplace equation in three dimensions. *Acta Numerica*, 6:229–269, 1997.

[22] Erwin Kreyszig. *Advanced Engineering Mathematics.* Wiley, 8th edition, 1999.

[23] Takayuki Fukui, Tadayoshi Uechi, and Nobuyuki Sano. Three-dimensional Monte Carlo Simulation of Electron Transport in Si Including Full Coulomb Interaction. *Applied Physics Express*, 1, 2008.

[24] Mona Zebarjadi, Ceyhun Bulutay, Keivan Esfarjani, and Ali Shakouri. Monte Carlo Simulation of Electron Transport in Degenerate Semiconductors. *Applied Physics Letters*, 90, 2006.

[25] C.H. Grein, M.E. Flatté, and Yong Chang. Modeling of Recombination in HgCdTe. *Journal of Electronic Materials*, 2008.

[26] Oliver Bonno and Jean-Luc Thobel. Monte Carlo modeling of carrier-carrier scattering in semiconductors with nonparabolic bands. *Journal of Applied Physics*, 104, 2008.

[27] C. Amsler *et. al.* (Particle Data Group). The Review of Particle Physics. *Physics Letters*, B667(1), 2008.

[28] Geir U. Jensen, Bjørnar Lund, Tor A. Fjeldly, and Michael Shur. Monte Carlo simulation of semiconductor devices. *Computer Physics Communications*, 67:1–61, 1991.

[29] Peter Capper. *Narrow Gap Cadmium-based Compounds*. Electronic Materials Information Service Darareviews Series. INSPEC, 1994.

[30] Karl Hess. *Advanced theory of semiconductor devices*. Prentice-Hall International, 1988.

[31] D.S. Kim. *Monte Carlo Modeling of Carrier Dynamics in Photoconductive Terahertz Sources*. PhD thesis, Georgia Institute of Technology, August 2006.

[32] Charles M. Wolfe, Nick Holonyak, and Gregory E. Stillman. *Physical properties of semiconductors*. Prentic-Hall International, 1989.

[33] B. Lund. *Monte Carlo Simulation of Charge Transport in Semiconductors and Semiconductor Devices*. PhD thesis, The Norwegian Institute of Technology, March 1992.

[34] Ben G. Streetman and Sanjay Kumar Banerjee. *Solid state electronic devices*. Pearson Prentice Hall, 6th edition, 2006.

[35] Boris Gelmont, Bjørnar Lund, Ki-Sang Kim, Geir U. Jensen, Michael Shur, and Tor A. Fjeldly. Monte Carlo simulation of electron transport in mercury cadmium telluride. *Journal of Applied Physics*, 71(10):4977–4982, 1992.

[36] T. Brudevoll. *Monte Carlo Algorithms for Simulation of Hole Transport in Homogeneous Semiconductors.* PhD thesis, The Norwegian Institute of Technology, May 1991.

[37] J.P. Laurenti, J. Camassel, A. Bouhemadou, B. Toulouse, R. Legros, and A. Lusson. Temperature dependence of the fundamental absorption edge of mercury cadmium telluride. *Journal of Applied Physics*, 67(10): 6454–6460, 1990.

[38] Peter Lawætz. *The Influence of Holes on the Phonon Spectrum of Semiconductors.* PhD thesis, The Technical University of Denmark, January 1978.