UPPSALA
UNIVERSITET

UPTEC F 22064
Examensarbete 30 hp
November 2022

Quantum Simulation of Quantum Effects in Sub-10-nm Transistor Technologies

Anders Winka

Civilingenjörsprogrammet i teknisk fysik

UPPSALA UNIVERSITET

Quantum Simulation of Quantum Effects in Sub-10-nm Transistor Technologies

Anders Winka

Abstract

In this master thesis, a 2D device simulator run on a hybrid classical-quantum computer was developed. The simulator was developed to treat statistical quantum effects such as quantum tunneling and quantum confinement in nanoscale transistors. The simulation scheme is based on a self-consistent solution of the coupled non-linear 2D SchrödingerPoisson equations. The Open Boundary Condition (OBC) of the Schrödinger equation, which allows for electrons to pass through the device between the leads (source and drain), are modeled with the Quantum Transmitting Boundary Method (QTBM) [1]. The differential equations are discretized with the finite-element method, using rectangular mesh elements. The self-consistent loop is a very time-consuming process, mainly due to the solution of the discretized OBC Schrödinger equation. To accelerate the solution time of the Schrödinger equation, a quantum assisted domain decomposition method is implemented. The domain decomposition method of choice is the Block Cyclic Reduction (BCR) method [2]. The BCR method is at least 15 times faster (CPU time) than solving the whole linear system of equations with the Python solver numpy.linalg.solve, based on the LAPACK routine _gesv [3]. In the project, we also propose an alternative approach of the BCR method called the "extra layer BCR" that shows an improved accuracy for certain types of solutions. In a quantum assisted version, the matrix inverses solved as a step in the BCR method, were computed on the D-Wave quantum annealer chip ADVANTAGE_SYSTEM4.1 [4]. Two alternative methods to solve the matrix inverses on a quantum annealer were compared. One is called the "unit vector" approach, based on work by Rogers and Singleton [5], the other called the "whole matrix" approach which was developed in the thesis. Because of the limited amount of qubits available on the quantum annealer, the "unit vector" approach was more suitable for adaption in the BCR method. Comparing the quantum annealer to the Python inverse solver numpy.linalg.inv, also based on LAPACK [3], it was found that an accurate solution can be achieved, but the simulation time (CPU time) is at best 500 times slower than numpy.linalg.inv.

Teknisk-naturvetenskapliga fakulteten

Uppsala universitet, Utgivningsort Uppsala/Visby

Handledare: Ahsan J. Awan Ämnesgranskare: Erik Sjöqvist

Examinator: Tomas Nyberg

Popular scientific summary

Transistors are the "brain cells" of computer chips. More transistors on a chip lead to improved computing power, enabling immense improvements in the capabilities of electronic devices. To increase the number of transistors on chips, the device sizes have to shrink. Transistor sizes have now entered into the sub-10 nanometer regime. When the device size is comparable to the electron wave function, quantum effects become present. Posing new challenges when manufacturing, but also modeling, new devices.

In this project, a device simulator is developed to capture static quantum effects (tunneling and confinement) when modeling transistors. To do this, we will solve a two-dimensional Schrödinger equation with open boundary conditions. The open boundary conditions account for the transistor's interaction with the two connected leads (the drain and the source). These simulations are very time-consuming. It includes solving a very large linear system of equations several times. To assist the simulation, a quantum annealer (a form of a quantum computer) is incorporated. However, the limitations of present-day quantum annealers necessitate us to reduce the size of the linear system of equations. Using a domain decomposition method called Block Cyclic Reduction (BCR), we can achieve this. When using the BCR method the linear system of equations is reduced by dividing it into several subproblems, where matrix inverses are computed. Theses matrix inverses can in fact be solved on the quantum annealer.

In the quantum assisted BCR method, the matrix inverses were computed on an actual quantum annealer. We used the D-Wave quantum annealer chip ADVANTAGE_SYSTEM4.1. Comparing the quantum assisted BCR method to the Python inverse solver *numpy.linalg.inv*, it was found that an accurate solution can be achieved, but the simulation time (CPU time) is at best 500 times slower than *numpy.linalg.inv*. Compared to solving the large linear system of equations with the Python solver *numpy.linalg.solver*, the quantum assisted BCR method is at best 25 times slower. The results suggests that the quantum computers have a long way to go before they efficiently can be incorporated into transistor modeling at the nanoscale.

Acknowledgements

Special thanks to my supervisor Ahsan J. Awan for help, support and guidance during the project. Also, thanks to Rana Pratap and Rui Hoi for fruitful discussions at our Monday meetings. Thanks also to David Bern for helping me and giving me a quick start with the quantum annealing formalism. Finally, lovely thanks to my partner for great support.

Table of Contents

List of Acronyms

BCR	Block Cyclic Reduction
BQM	Binary Quadratic Model
BTE	Boltzmann Transport Equation
DD	Drift-Diffusion
DDM	Domain Decomposition Method
DG	Density Gradient
DOS	Density Of States
LLS	Least Linear Squares
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
NEGF	Non-Equilibrium Green's Function
OBC	Open Boundary Condition
PME	Pauli Master Equation
QC	Quantum Computer
QTBM	Quantum Transmitting Boundary Method
QUBO	Quadratic Unconstrained Binary Optimization
RTD	Resonant Tunneling Device

1 Introduction

The evolution of 5G towards limitless connectivity, intelligent network platforms, and edge computing requires significant computational power in the telecommunication infrastructure. The performance demands of new apps may be fulfilled with new sub-10-nm transistor technologies, where quantum effects dominate the electrical, thermal, and switching characteristics of transistors. Modeling these effects accurately is important to ensure the integrity of circuit design in this domain.

Moore's law, first predicted by Gordon Moore in 1965 [6], states that the number of transistors that can be placed on an integrated circuit doubles every two years. So far, this law has been an observed fact with Samsung beginning their production of gate-all-around (GAA) 3-nm transistors this year (2022) [7]. The increase in transistor density, together with enhanced performances of single transistors, has lead to tremendous developments in the capabilities of electronic devices. For future developments to continue, the transistor technologies have to continuously improve. But as device sizes shrink, quantum mechanical effects become present in their operations.

When developing new transistor technologies, the use of computer aided design (CAD) tools has always been of great importance to predict design characteristics before fabrication. For classical simulations of Metal Oxide Semiconductor Field Effect Transistor (MOSFET) with gate lengths longer than 100nm, the Drift-Diffusion (DD) simulation [8] has been very popular and accurate for performance predictions. The DD model is however a classical model and cannot capture quantum effects such as energy quantization, quantum tunneling, the wave nature of electrons and holes, and the atomistic granularity, which are all important at the nanometer scale and physical effects which cannot be neglected. The need for simulation models which can incorporate these quantum effects is a necessity for the discovery and development of modern nanoelectronic devices.

There exist several such models, called quantum models (none are perfect), such as the self-consistent solution of the Schrödinger equation with open boundary condition used in this project. These models can more accurately simulate nanoscale devices compared to the DD simulation, but with the cost of increased computational burden. The computational complexity of the quantum models reduces their usability. Simulation domains are often limited to two dimensions when in fact the transistors are three dimensional devices, the simulation times can be very long and solutions hard to achieve. However,

with improving supercomputers (thanks to better transistors) and more important for this project; with the dawn of quantum computers, new opportunities for the quantum models can be facilitated.

To fully exploit the parallel capabilities of supercomputers (and quantum computers) a Domain Decomposition Method (DDM) can be employed which will severely improve simulation time. DDM's also has an important task when implementing solvers on the present existing quantum computers. Current quantum computers are limited by their amount of coherent logical qubits, and therefor they cannot solve large problems. The DDM reduces the large problem to several smaller sub-problems, which can be solved on the quantum computer with current limitations.

If Moore's law is to persist in the future, transistors have to shrink to sizes that are impossible to manufacture because of physical limitations. In one way, marking the lower limit is the single-atom transistors invented and first demonstrated by Prof. Thomas Schimmel, et al. in 2004 [9]. One way to move beyond this limitation is with quantum computers. Described as physical systems that with the laws of quantum mechanics performs arithmetic and logical operations much faster than a classical computer, or as cited by Harrow, Hassidim and Lloyd (HHL) [10], "quantum computers are devices that harness quantum mechanics to perform computations in ways that classical computers cannot." There currently exist two complementary specimens of quantum computers, circuit-based systems and quantum annealers. The circuit-based systems uses deeper quantum mechanical phenomenons such as coherence, entanglement, and non-locality, while the quantum annealers mainly exploit tunneling between metastable states and the ground state [11]. Presently, the quantum annealers have a bigger capacity, with D-Wave Quantum Annealers having 5000+ qubits [4], compared to the 127 [12] qubits which can be entangled into a fully coherent state in a circuit based quantum computer. The greater capacity makes the quantum annealer more suitable for linear algebra, which we will take advantage of in this project.

In this project a 2D device simulator run on a hybrid classical-quantum computer is presented. The simulator is developed to treat statistical quantum effects such as quantum tunneling and quantum confinement. This is done through a self-consistent iterating scheme of the solution from the coupled non-linear 2D Schrödinger-Poisson equation. This self consistent loop is very time consuming and to accelerate the simulation time, a quantum assisted domain decomposition method is implemented for the Schrödinger equation. The focus in the project was to accelerate the solution of the OBC Schrödinger equation. In the end no self-consistent solution of the Schrödinger and Poisson equation was achieved and is left as a future work. The open-boundary conditions of the Schrödinger equation is modelled with the QTBM [1]. The domain decomposition method of choice is the BCR method [2], which in the end reduces the size of the linear

system of equations that we end up with after discretization of the Schrödinger equation. In the BCR method several matrix inversions are computed, which will be solved on a quantum annealer.

1.1 Aim

The aim of this project is to simulate quantum circuits for quantum effects in sub-10 nm transistor technologies with a quantum algorithm. To do this a self-consistent model of the non-linear coupled Schrödinger-Poisson equation is employed, mainly based on the work of Laux et al. in [13] and Lent and Kirkner in [1]. The model is implemented in Python and discretized using the finite-element method with rectangular mesh elements, which transforms the physical equations to problems of linear algebra.

Focus in this project is on the solution of the discretized 2D effective mass Schrödinger equation, the most time-consuming part of the simulation scheme. We wish to solve this part on a quantum computer. Because of today's limitations of quantum computers (few qubits and poor connectivity), a DDM called the BCR method [2] is used to make the Schrödinger equation tractable to solve on the quantum computer. The quantum computer used in this project is the D-Wave ADVANTAGE_SYSTEM4.1 chip [4], a quantum annealer with 5000+ physical qubits. To map our problem to the quantum annealer, a Quadratic Unconstrained Binary Optimization (QUBO) problem is constructed for finding matrix inverses.

Results are presented for the performance of the BCR method, as adopted to our problem. A suggested improvement to the BCR method, called the "extra layer" BCR is also presented. The "extra layer" BCR can improve the accuracy of the solutions for certain injection energies over the regular BCR method. We also compare two different constructions of QUBO problems for the matrix inverse. The QUBO problem which uses the least amount of qubits, called the "unit vector" approach is implemented as part of the BCR method. The quantum assisted BCR method is compared to the Leap hybrid solver [14], a hybrid quantum-classical solver distributed by D-Wave.

1.2 Scope of project

Significant approximations are adopted in the physical model of this project. We only considers two-dimensions, even though transistors are three-dimensional devices. The empirical effective mass approximation is employed and a simple, six-valley, parabolic band structure is used for silicon. Also, we consider ballistic transport, so scattering is neglected. Theses approximations are made because of the computational complexity of first-principles models. Additional simulating schemes are outlined and discussed in

the Literature Review 4 and Discussion 6.

In this project, only the solution of the Schrödinger equation for a resonant tunneling device is considered in the results section. For a fully working device simulator, where current-voltage curves can be obtained, a self-consistent simulation scheme must be considered. In this project the simulation scheme is outlined, but because of the limited time we were not able to implement a functioning version of a self-consistent scheme. A fully working self-consistent scheme is left as a future work.

The thesis begins with a Background section, starting with a outline of the working principles of transistors and the quantum effects in nanoscale transistors. After that, the physical model employed in this project is presented. Here the equations, and their motivation, implemented in our simulation scheme are presented. In the Background the quantum annealer and the QUBO formalism for a least linear square problem are also outlined. Finally, we present the BCR method.

In the Methodology chapter the simulation scheme is introduced. Here it is described how the background is implemented in the Python program. This includes the construction of the linear system of equation corresponding to the discretized OBC Schrödinger equation, the adaption of the BCR method and the QUBO formalisms for the matrix inverse. As mentioned before, the whole self-consistent scheme is presented, however results are only presented for the solution of the OBC Schrödinger equation.

The report also consists of a comprehensive Literature Review. Since no previous work was available at Ericsson where a quantum algorithm is used to simulate transistors, a thorough literature study was conducted. In this section various nanoscale models are presented, from macroscopic models to models based on first-principles. We also outline other quantum algorithms for linear algebra problems and DDM considered in the literature in the making off this thesis.

In the results we will present the solutions of the discretized OBC Schrödinger equation for a resonant tunneling device. A suggested improvement to the BCR method, called the "extra layer" BCR is also presented. The "extra layer" BCR can improve the accuracy of the solutions for certain injection energies over the regular BCR method. We also compare two different constructions of QUBO problems for the matrix inverse. The QUBO problem which uses the least amount of qubits, called the "unit vector" approach is implemented as part of the BCR method. The quantum assisted BCR method is compared to the Leap hybrid solver [14], a hybrid quantum-classical solver distributed by D-Wave.

The results are followed by a Discussion, where the results will be discussed and an outlook to future projects in the field is presented. Lastly, the conclusions of the project are drawn.

2 Background

2.1 Transistors

To give a good background to the project, it seems fitting to start of with a walk through of the technical device that we will simulate, the transistor. First invented in 1947-48 by physicists Bardeen, Brattain and Shockley at the Bell Labs, who later won the Nobel price in Physics for their invention. The transistor is a semiconductor device used for switching and amplifying electrical signals. Being the active component of integrated circuits, it deeply embeds modern electronics. It usually comprise of three electrical leads, the source, the drain and the gate. In figure 2.1 a MOSFET (the most common transistor type) is shown where the three leads are marked out.

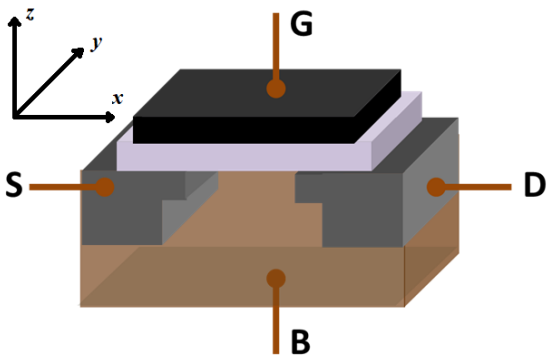


Figure 2.1: Structure of the MOSFET device. G is the gate, B the body and S and D are the source and drain. The gate is separated from the rest of the device with an insulated layer (the white block in the figure), which is usually made of an oxide [1]

The source and drain is connected with a channel of a semiconductor and the gate is isolated with an oxide (the white block in figure 2.1). An applied signal to the gate adjusts the electron bands in the semiconductor which influences its ability to conduct electrical current. A current then flows from the source to drain if a V_{SD} bias is applied, but no current is transmitted to the gate thanks to the oxide. The working principles of a MOSFET can be seen in figure 2.2. This means that the gate node can determine if a current can flow or not, by applying a small voltage of typically a few volts. In this manner the transistor acts as a switch, either the current flows and the circuit is active, or it does not and the circuit is off. The two distinct states corresponds to the binary 1's

[1] Brews ohare, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons. URL: https://commons.wikimedia.org/wiki/File:MOSFET_Structure.png

and 0's in digital computers. The transistor can also be used as an amplifier because the output power can be higher than the input power. [15]

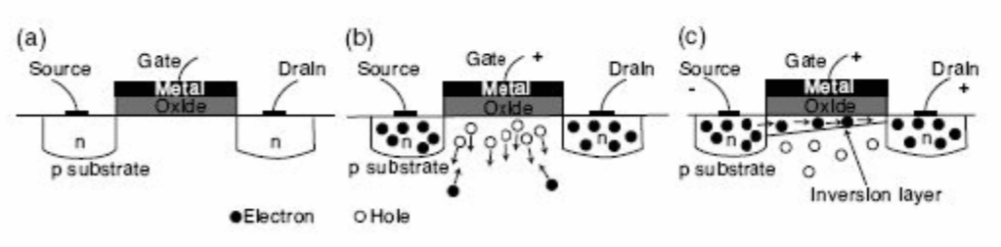


Figure 2.2: The working principles of a MOSFET device. In stage (a) no voltage is applied; in (b) a small positive gate voltage is applied; in (c) a gate voltage large enough to create an inversion layer is applied, aswell as a voltage between source and drain [2]

Since the discovery, the size of the transistors has steadily shrunk over the decades thanks to improved production techniques, explosively increasing the possible circuit complexity. The transistor sizes has now reached the nanometer regime, introducing with it new challenges.

The quantum effects are an actuality and cannot be avoided, but they might have work arounds. As discussed before in section 2.1, transistors are used as switches, with an on state and an off state, in circuits. This demands certain properties of these transistors, such as a rapid on-off transition. Also important is a low voltage in the on state and a very low off current, the current in the off state. A leakage current in the off state of a device in a circuit can cause serious problems with heat dissipation. In planar FET this is a problem, leading to a transition to FinFET devices where the off current is controlled by pinching it with two opposing voltages from either side of the fin. A similar transistion to gate-all-around quantum wire FETs is likely to happen. [17]

2.1.1 Challenges in sub-10nm

When the device size shrinks, static and dynamical quantum effects associated with the wave properties of electrons become present and crucial for the device's performance. Static quantum effects include tunneling, confinement and interference, and dynamical quantum effects include scattering. Static effects are physically well defined and can be implemented in conventional simulation schemes. One example is the Density Gradient (DG) method which can be incorporated in the DD simulation to account for tunneling and confinement [18]. Dynamical quantum effects associated with energy dissipating scattering in electron transport are physically much more involved and difficult to implement. These dynamical effects have never been critical in MOSFETs up until now, and basic device properties have been explained within the framework

[2] **Figure 7.13** in Hofmann's textbook *Solid state physics: an introduction* [16]

of semiclassical transport theory, such as Boltzmann transport equation [19]. But the semiclassical transport equations cease to be valid once wave characteristics cannot be ignored and they cannot treat the quantum effects accurately. Wave characteristics become important once the de Broglie wavelength of electrons is comparable to the size of the device. Typical de Broglie wave-lengths in Si is several nm, comparable to sub-10 nm MOSFETs. The phase coherence of the electrons must be taken into account in the transport analyzes. As this happens, the use of quantum transport models becomes a necessity. [20]

But this brings about several fundamental problems one must conquer. The ultrasmall devices, where significant quantum effects are anticipated, are inherently three dimensional (3D). The solution of the 3D Schrödinger equation, that governs the wave-function of the electron, are extremely computational heavy. Another challenge is the treatment of the connection between the device region (channel) and the classical reservoirs (source and drain), from where the macroscopic current is obtained. As cited from Sano et. al., "the entire device is intrinsically an open-system and the quantum region and classical reservoirs must be treated on the same physical ground. This is, of course, one of the most difficult problems in basic physics" [20]. In this project we will try and deal with both these problems.

2.2 Transistor modeling

When developing and optimizing devices a modeling tool is essential to better understand its characteristics. The goal is to describe how the electrons move in the semiconductor and its connecting leads and insulators. It is based on a mathematical description of electron transport. As there exist several such descriptions, it can be a challenge to select one. First-principle calculations such as electronic structure calculations can accurately describe the nanoscale physics, but are harder to solve. Other, more well established simulation schemes such as the DD simulation have not been changed for the last decades and is still frequently employed for its success of simulating MOSFET devices down to the sub-100 nm regime [21]. But the underlying equations of the drift-diffusion method cannot describe the nano physics and quantum effects at the sub-10 nm scale, which are of interest in this project. It is a balancing act between sufficient physical descriptions and computational efficiency, where several things have to be considered when choosing a simulation scheme. In this section, we will present the physical effects that we want to capture and the approach chosen to model a transistor.

2.2.1 Quantum effects in nanoscale transistors

The aim in this project is to simulate quantum effects present at the sub-10 nm scale. The quantum effects can roughly be divided into three categories, which are listed below.

- Tunneling
- Confinement
- Scattering

These will be discussed in this section.

2.2.1.1 Tunneling

First of all, quantum tunneling. A true quantum effect where the wave-function of the electron Ψ has a finite probability to propagate through a thin potential barrier, even if the potential barrier is larger than the electron's kinetic energy. It can be explained with the Heisenberg uncertainty principle. Consider the momentum and position uncertainty

$$\Delta x \Delta p \approx \frac{\hbar}{2} \tag{2.1}$$

where Δx is the position uncertainty, Δp the momentum uncertainty and $\hbar$ the reduced Planck constant. If the position of the particle can be pinpointed accurately, the uncertainty in the momentum becomes large and thus also the uncertainty in the kinetic energy. If the uncertainty becomes so large that the kinetic energy overcomes the potential barrier, the electron go over it [22]. Below, in figure 2.3 tunneling through a potential barrier is visualized.

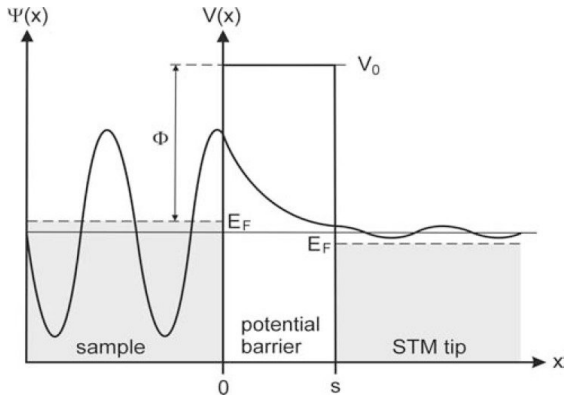


Figure 2.3: Quantum tunneling through a one-dimensional potential barrier of thickness s. In the context of the figure, the potential barrier is a small gap between two electrodes consisting of a sample and the tip of a sampling tunneling microscope (STD). Φ is the tunneling barrier height, E_F are the Fermi levels of the two metal electrodes. The wave function enters the potential barrier at $x = 0$ and then decays exponentially inside the potential barrier. It exists on the other side of the potential barrier with a reduced, although non-zero, amplitude. [3]

Quantum tunneling has been a known phenomenon in transistors for some time, first as a concern for a leakage current through the gate oxide. A well known problem in

[3] Figure taken from F. Trixler's article *Quantum Tunnelling to the Origin and Evolution of Life* [23]

MOSFETs with thin dielectrics for almost as long as they have existed [24]. As the devices continues to shrink, direct tunneling between the source and drain can also be an issue [25] as a result of resonance.

2.2.1.2 Confinement

Next on the list is quantum confinement. As the name describes, quantum confinement is a restriction of the electrons and holes movements. Quantum mechanically, an electron is represented with a wave-packet of size 3-7 nm [26]. When comparing to the classical size of a sub-10 nm MOSFET, that is a significant size. So how can you fit an 5 nm electron into a classical size inversion layer, corresponding to a 1 nm thin potential well? The answer is that you cannot. Confining the electrons to the small inversion layers means that a self-consistent solution of the Poisson-Schrödinger equation must be employed, leading to discretized sets of quantum levels and other quantum effects. In figure 2.4 we see confinement in one-dimension, two-dimensions, and three-dimensions and the resulting quantization of the density of states in the conduction and valence bands.

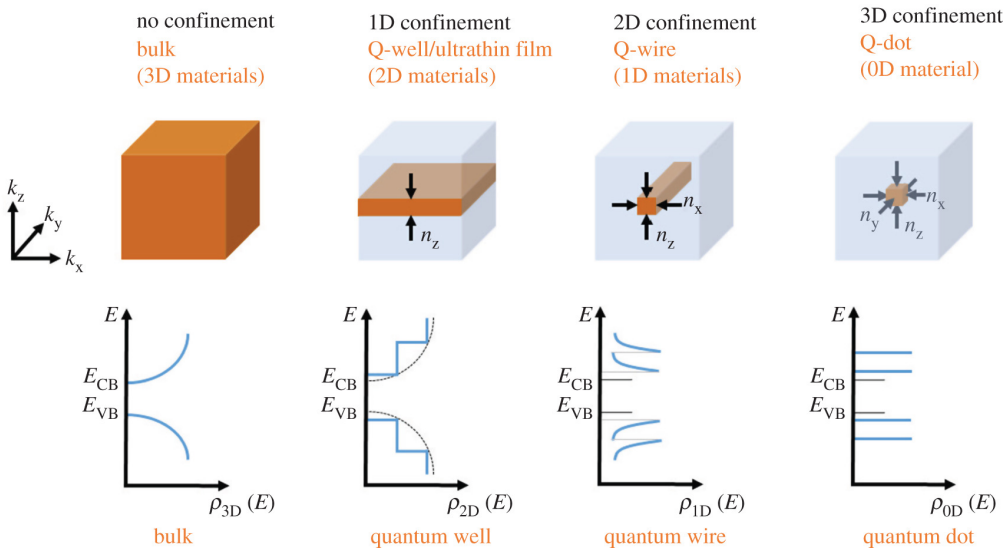


Figure 2.4: Quantum confinement in 0D ,1D, 2D and 3D materials. In the graphs under the material samples the density of states ρ for the valence band (VB) and conduction band (CB) are shown, which emphasise the effects of the quantum confinement. [4]

The quantum effects become present because the electric carriers no longer are simple localized objects, but defined by quantum wave packets, which are deformed when confined to a small volume. One example of such a quantum effect happens at the inversion layer at the $Si/Si0_2$ interface. A classical treatment of a standard MOSFET leads to a carrier density peak at the Si and oxide interface, and an exponential decrease

[4] Figure taken from T. Edvinsson's article *Optical quantum confinement and photocatalytic properties in two-, one- and zero-dimensional nanostructures* [27]

away from the surface. This is a result of the positive gate voltage attracting the negative charge carriers. See figure 2.5.

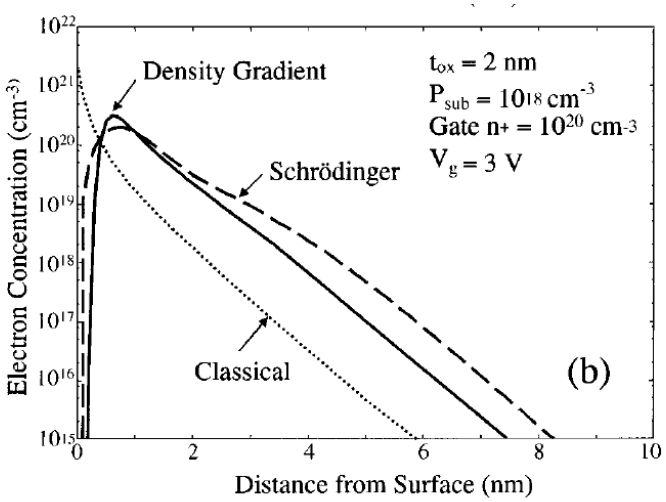


Figure 2.5: Electron charge density in the center of the channel of a n-MOSFET, obtained from the density gradient method (solid line), Schrödinger equation (dashed line), and the classical drift diffusion method (dotted line). [5]

A quantum treatment leads to a opposite result, the charge has to be zero at the $Si/Si0_2$ interface. The local charge density ρ is usually proportional to the wave function Ψ as $\rho = -e|\Psi|^2$ and because of the large offset potential of the oxide, the wave function will vanish at the interface, which means that the charge also vanishes at the interface. [17]

2.2.1.3 Scattering

Lastly, we will talk about scattering, a dynamical quantum effect. As the charge carriers move in a device, they will inevitably scatter on phonons (quantized vibrations of the lattice), interface roughness [28] and impurities (due to doping of the substrates), reducing their mobility. Since the electrons are quantum mechanical (treated as wave packets), the scattering must be treated in a quantum mechanical fashion. This is not an effect which our simulator presently can handle, but by implementing the Pauli Master equation formalism, see section 4.1.2.3, this can be done.

2.3 Schrödinger equation with open boundaries

To capture the quantum tunneling and quantum confinement we wish to solve the Schrödinger equation and Poisson equation self-consistently to determine a device's electrical behaviour, visualized in figure 2.6 below.

[5] Figure taken from N. Sano et. al.'s article *Device Modeling and Simulations Toward Sub-10 nm Semiconductor Devices* [20]

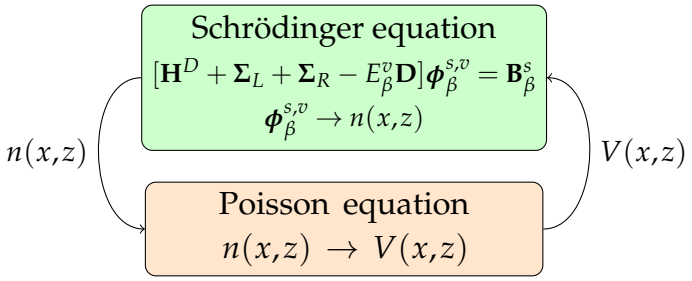


Figure 2.6: The self-consistent Schrödinger-Poisson loop. The $\boldsymbol{\phi}_\beta^{s,v}$ is the electron wave function, $n(x,z)$ the charge density and $V(x,z)$ the potential energy.

The Schrödinger equation is the treaty of this section and the following subsections. To start of, we will point out some of the features of the numerical scheme employed, mainly drawn from work of Laux et. al. [13] and Vyas et. al. [29]. We will set the x, z and y coordinates to the transport, confinement and out-of-plane directions. The simulations are done in the two dimensions (2D) of the x-z plane. We are using the effective-mass approximation, only considering the first conduction band of Si and the band minima is approximated with six equivalent ellipsoidal valleys, see appendix A for a description of these concepts. The potential profile is assumed to be translational invariant in the out-of-plane y direction, a valid approximation for very wide devices, which means that this part can be separated from the Schrödinger equation. We assume ballistic transport because of simplicity, so no scattering is considered. Ballistic transport could be considered as a "best case" analysis of device behaviour, being useful in that regard. With an extension of the model outlined here with the Pauli master equation formulation [30][31][29], it is possible to include scattering into the approach outlined here. See section 4.1 in the Literature review chapter for a more detailed discussion on approaches to simulate scattering.

The goal when solving the Schrödinger equation is to study the system's behavior under an applied bias (drain-source voltage) where particles can be exchanged with the environment through contacts. To treat the interaction with the environment the QTBM is used, which we outline in Appendix B. The source and drain contacts are idealized as semi-infinite long leads (no variation in the potential along its length) connected to the active region of the device. Electron waves are injected from the leads and we want to find their distribution inside the device and calculate the current into the leads.

The solution domain Ω can be divided into a device region Ω_0 and several lead regions $\Omega_1, ..., \Omega_n$ such that $\Omega \equiv \Omega_0 \cup \Omega_1 \cup ... \cup \Omega_n$. See figure 2.7. The boundary of the device Ω_0 not in contact with a lead is named Γ_0 and the shared boundaries of a lead s and Ω_0 are called Γ_s. Associated with each lead is a local coordinate system (η_s, ξ_s), where η_s denotes the lead direction and ξ_s denotes the confinement direction.

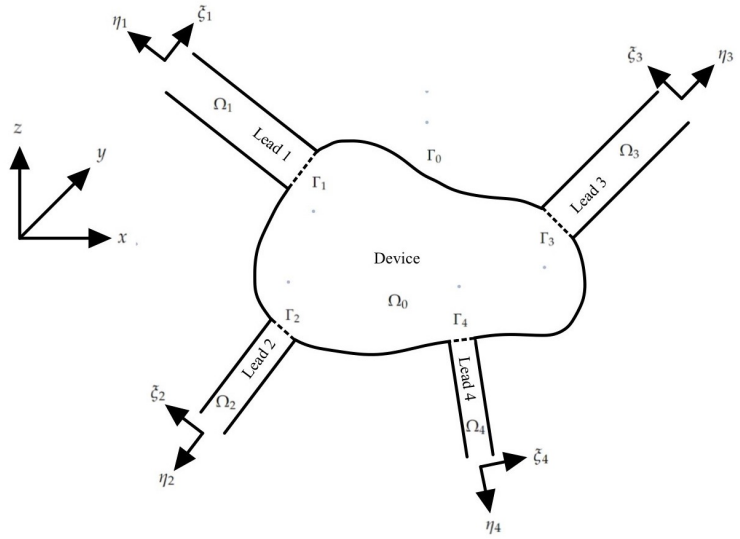


Figure 2.7: Problem geometry

The wave functions inside the leads Ω_s are denoted $\psi^{v}_{\beta,s}(\eta_s,\xi_s)$, where v denotes the valley's in *Si* (see appendix A), β is the injection energy of the electrons injected into the device, s is the lead. The confinement in the ξ_s direction gives wave functions that are the m eigensolutions $\chi^{s,v}_m(\kappa_s)$ (subbands) of the 1D Schrödinger equation with discrete energy spectrum $E^{s,v}_m$ (in the QTBM appendix B, the β, v indices are dropped).

Inside the device region Ω_0 we have wave functions $\phi^{s,v}_\beta(x,z)$. These are obtained by solving the time-independent single-electron Schrödinger equation using the effective-mass approximation in the 2D x-z plane, independently for every lead s. We assume that the real space axes (x,z,y) align with the reciprocal lattice axes (k_x,k_z,k_y), and that the employed simple parabolic energy bands sufficiently describes the conduction band in the whole domain Ω. The resulting Schrödinger equation follows

$$-\frac{\hbar^2}{2}\left[\frac{1}{m^v_x}\frac{\partial^2\phi^{s,v}_\beta(x,z)}{\partial x^2}+\frac{1}{m^v_z}\frac{\partial^2\phi^{s,v}_\beta(x,z)}{\partial z^2}\right]+V(x,z)\phi^{s,v}_\beta(x,z)=E^v_\beta\phi^{s,v}_\beta(x,z) \tag{2.2}$$

V(x,z) is the potential energy distribution, m^v_x and m^v_z are the effective masses in the x respective z direction, $\hbar$ is the reduced Plank's constant. The y direction has been suppressed out using separation of variables. To obtain the full 3D wave function in the entire device $\Psi^{s,v}_\beta(x,y,z)$ the out-of-plane wave functions e^{ik_y} must be multiplied to $\phi^{s,v}_\beta(x,z)$, we get

$$\Psi^{s,v}_\beta(x,y,z)=e^{ik_y y}\phi^{s,v}_\beta(x,z). \tag{2.3}$$

Equation 2.2 is solved for three distinct electron configurations, because of the six valleys in Si, see appendix A.

2.3.1 Discretization of the Schrödinger equation

The Schrödinger equation in 2.2 cannot be solved analytically, so we have to solve it numerically on a computer. To do so, we have to discretize it. Doing a finite-element discretization (see appendix B.2) and using the QTBM boundary conditions (outlined in appendix B), equation 2.2 becomes a linear matrix equation

$$[\mathbf{H}^D + \mathbf{\Sigma}_L + \mathbf{\Sigma}_R - E_\beta^v \mathbf{D}]\boldsymbol{\phi}_\beta^{s,v} = \mathbf{B}_\beta^s. \tag{2.4}$$

$\mathbf{H}^D = \mathbf{T} + \mathbf{V}$ is the $N \times N$ Schrödinger Hamiltonian of the device consisting of a kinetic term $\mathbf{T}$ and a potential term $\mathbf{V}$, $\mathbf{D}$ is a $N \times N$ matrix defined in equation B.34, $\mathbf{B}_\beta^s$ is a $N \times 2$ vector of the injected modes into the device (the 2 stems from the fact that we consider injection from 2 leads independently). $\mathbf{\Sigma}_L$ and $\mathbf{\Sigma}_R$ are $N \times N$ matrices representing the self-energy, which includes reflected and transmitted (traveling- and evanescent-) waves going into and out of the device. These matrices are calculated with the QTBM in Appendix B, section B.1.5.1. The interfaces where no electrons are injected Γ_0 have zero-value Dirichlet boundary conditions implemented (the wave function $\boldsymbol{\phi}_\beta^{s,v}$ are set to zero). Equation 2.4 is solved for the electron wave vectors $\boldsymbol{\phi}_\beta^{s,v}$, for all different injection energies E_β^v, leads s, and conduction-band valleys v. The electron wave vectors is then used to calculate the charge distribution inside the device, we will do this in section 2.3.3. The charge distribution shows the electron occupation in the device and it is used to find a self-consistent solution of the device. [29]

2.3.2 Discretization of the continuous energy spectrum

The energies E_β^v is in reality a continuous energy spectrum, and therefore it needs to be discretized for computational reasons. A convenient discretization preferably samples the states (wave-functions) that matters the most inside the device, since they contribute the most to the charge distribution. It is not straightforward how to perform this sampling, and there exist several approaches to this problem. The approach used in this project is outlined in [13]. The energy spectrum is sampled by imposing two different closed boundary conditions of the Schrödinger equation, either zero-value Dirichlet (the wave function is set to zero)

$$\phi_\beta^{s,v} = 0 \text{ on } \Gamma_s \tag{2.5}$$

or zero-value Neumann boundary conditions (the wave function's normal derivative are set to zero)

$$\nabla\phi_\beta^{s,v}\hat{\eta}_s = 0 \text{ on } \Gamma_s \tag{2.6}$$

on the device-lead interfaces Γ_s, thus double sampling the spectrum. The two resulting eigenvalue problems are solved, yielding standing-wave eigensolutions. The eigenvalues from the solutions make up the energy sampling of the injection energies E_β^v of the open system. This is a good spectrum for obtaining the correct density of states [29].

The eigenfunctions from the eigenvalue problem will have "cosinelike" or "sinelike" behaviour at the Γ_s boundaries. They form a complete orthogonal basis set that spans the Hilbert space of physical solutions in the device, also obeying the injecting boundary conditions of the open system in equilibrium, i.e. the case of no applied bias (no difference in chemical potential between source and drain). Even if the device is driven out equilibrium, the sampling scheme can vigorously discretize the continuous energy spectrum such that accurate device solutions and good convergence are provided [29].

2.3.3 Calculation of charge distribution

Using the discretized energy spectrum obtained from the method mentioned in the above section 2.3.2 when solving the open system Schrödinger equation (see equation 2.4), one gets wave functions $\phi_\beta^{s,v}(x,z)$ which can be "box" normalized to the device volume. This is an advantage of the energy discretization scheme. The normalization is done as follows

$$\int_\Omega dxdz \left(\sum_s |\phi_\beta^{s,v}(x,z)|^2 \right) = \frac{1}{2} \tag{2.7}$$

Instead of $|\sum_s \phi_s|^2$, we have $\sum_s |\phi_s|^2$ because the leads do not inject the traveling modes coherently. The $\frac{1}{2}$ factor arises because the energy spectrum is sampled with two complete sets of device normal modes, the "sinelike and the "cosinelike" closed system eigenfunctions. The wave functions contains the density of states information in the x and z directions, and to obtain the 3D electron density we have to include the out-of-plane y direction. To do this we use the 1D Density Of States (DOS), $D_{1D}^v(E_y)$ representing the continuous energy spectrum along the homogeneous y direction, given by

$$D_{1D}^v(E_y) = \frac{1}{\pi\hbar}\sqrt{\frac{m_y^v}{2E_y}} \tag{2.8}$$

and $E_y = \hbar^2 k_y^2 / 2m_y^v$. In the case of ballistic electron transport the electron density $n(x,y)$ can be calculated by summing all possible traveling modes for the injected states and associating each wave function with an occupancy factor decided by the Fermi-Dirac distribution. I.e., the different current carrying states have an associated energy and the Fermi-Dirac distribution yields a factor for each energy which is multiplied with the current-carrying state. We also include the y direction 1D DOS, where we can integrate out E_y. The following expression is obtained:

$$n(x,z) = \sum_s \sum_v \sum_\beta \int_{-\infty}^{\infty} dE_y \left[\frac{1}{\pi\hbar}\sqrt{\frac{m_y^v}{2E_y}} \times f_{FD}^s(\mathcal{E},T) \right] |\phi_\beta^{s,v}(x,z)|^2, \tag{2.9}$$

where $f_{FD}^s(\mathcal{E},T) \equiv \{1 + exp[(\mathcal{E} - E_F^s)/(k_B T)]\}^{-1}$ is the Fermi-Dirac distribution, E_F^S is the Fermi level of lead s, T the temperature in Kelvin and $\mathcal{E}$ is the total energy given by

$$\mathcal{E} = E_\beta + E_y. \tag{2.10}$$

Using the expression for the Fermi-Dirac integral of order $-\frac{1}{2}$, given by $F_{-\frac{1}{2}}(x) \equiv \int_0^\infty dt\{t^{-\frac{1}{2}}[1+exp(t-x)]^{-1}\}$, we obtain the final expression for the charge distribution

$$n(x,z) = \sum_s \sum_v \sum_\beta \frac{1}{\pi\hbar}\sqrt{\frac{m_y^v k_B T}{2}} \times F_{-\frac{1}{2}}\left(\frac{E_F^s - E_\beta^v}{k_B T}\right)|\phi_\beta^{s,v}(x,z)|^2. \tag{2.11}$$

The charge distribution is calculated separately for different leads s, where each lead has a associated quasi-Fermi-level E_F^s, and then summed together. $p(x,y,z)$, which is the hole charge distribution is calculated semi-classically using the known 3D DOS expression

$$p(x,z) = \frac{1}{2\sqrt{\pi}}\left(\frac{2m_h k_B T}{\pi\hbar^2}\right)^{3/2} \times F_{\frac{1}{2}}\left(\frac{V(x,z) - (E_F + E_G)}{k_B T}\right) \tag{2.12}$$

where $m_h = 0.8m_0$ is the hole effective mass, E_F is a constant Fermi level in the device, E_g is the silicon band-gap energy and $F_{\frac{1}{2}}$ is the Fermi-Dirac integral of order $\frac{1}{2}$.
[29]

2.4 Poisson equation

When the distribution of electrons, holes and ionized dopants (all charge carriers) have been found, the Poisson equation is solved to find the new potential distribution in the 2D (x,z) cross section of the device, V, given by

$$\nabla \cdot [\epsilon(x,z)\nabla V(x,z)] = e\rho(x,z), \tag{2.13}$$

where

$$\rho(x,z) = e[p(x,z) - n(x,z) + N_A(x,z) - N_D(x,z)] \tag{2.14}$$

is the total charge, ϵ is the permittivity, e the electron charge, and N_A (N_D) the acceptor (donor) concentration, characteristics of the simulated transistor. To obtain the conduction band edge potential energy V_C, we have to add a discontinuity term $\Delta E_C(x,z)$ such that $V_C = V(x,z) + \Delta E_C(x,z)$. The discontinuity term describes the discontinuity in the conduction band edge energy between different materials, such as Si and SiO_2 [13].

A gate potential V_{GS} is included by imposing Dirichlet boundary conditions at the domain edge representing the oxide-metal interface, see figure 2.1 or 2.2 to see the oxide placement. Zero-normal derivative is imposed on all other non-lead boundaries,

$$\nabla V(\mathbf{r}) \cdot \hat{n} = 0 \text{ for } \mathbf{r} \in \Gamma_0 \tag{2.15}$$

where $\hat{n}$ is the unit normal vector. The boundary conditions for the lead-device interface is treated in the next section. To solve the Poisson equation, we discretize it with the

same finite-element mesh as the Schrödinger equation, and we obtained the following linear system of equations

$$\mathbf{PV} = \mathbf{c}, \tag{2.16}$$

where $\mathbf{P}$ is a $N \times N$ matrix representing the discretization of the Laplace operator, $\mathbf{V}$ is a $N \times 1$ column vector representing the unknown potential V and $\mathbf{c}$ is a $N \times 1$ column vector representing the known charge density $e\rho(x,z)$.
[29]

2.4.1 Lead boundary conditions Poisson equation

When solving open systems, it can occur (typically at high V_{DS}) unphysical electron depletion or accumulation at the lead-device interface, which is a known problem. Methods to tackle this include

- adjusting Fermi levels at the leads to maintain charge neutrality [32].
- A Drifted-Fermi distribution (physically consistent but expensive) [13].
- Zero normal derivative boundaries at the contacts [29].

Here the zero normal derivative is used since it is computational less expansive. Its major limitation, outlined in [13], is that it forces the electric field to be zero at the contacts, even for biased devices. This is not a physical behaviour. However, tests done in [29] showed that the zero normal derivative produced similar results to the more physically consistent drifted Fermi distribution.

2.5 Self-Consistent scheme

The self-consistency scheme can be presented as a step-by-step process:

1. Set up the problem (geometri, material, etc) and conventions used. Here two dimensions, Si, etc are used.
2. Calculate the discretized energy spectrum by solving the eigenvalue problems corresponding to the standing-wave boundary conditions.
3. Find the electron wave functions in the device domain, using the Quantum Tansmitting Boundary Method (QTBM) and Schrödinger equation. It is solved by using the finite-element method with the QTBM boundary conditions, equation 2.4.
4. Once the wave functions are found, the ballistic charges can be calculated with equation 2.11.
5. Using the distribution of carriers, the Poisson equation 2.16 is solved to find the new potential V.

6. Checking for convergence by comparing the old and new potential. If convergence is achieved the self-consistent solution is obtained and we can calculate the current. Else, we repeat steps 2. to 5., but with the new potential.

Once the self-consistent solution of the open system is obtained, the different current transport parameters, which is, the DOS, the current-density distribution, and the total drain current can be calculated.

2.6 Calculation of current

As said, once the self-consitent solution of the Schrödinger and Poisson equation is obtained, the current through the device can be calculated. First, define the current at lead s as [13]

$$\begin{aligned}\vec{J}^s_{dev} = \sum_{v,\beta} e \int_0^{d_s} d\xi_s \Bigg\{ \frac{1}{\pi\hbar}\sqrt{\frac{m^v_y k_B T}{2}} F_{-\frac{1}{2}}\left(\frac{E^s_F - E^v_\beta}{k_B T}\right)\left(-\frac{i\hbar}{2m^{*v}_{\eta_s}}\right) \\ \times \left[\left(\nabla\phi^{s,v}_\beta(\eta_s,\xi_s)\right)^* \phi^{s,v}_\beta(\eta_s,\xi_s) - \nabla\phi^{s,v}_\beta(\eta_s,\xi_s)\left(\phi^{s,v}_\beta(\eta_s,\xi_s)\right)^*\right]\Big|_{\eta_s\in\Gamma_s}\Bigg\}\end{aligned} \tag{2.17}$$

where d_s is the width of lead s and * denotes the complex conjugate. To get the current intensity I in the device we have to sum all contributions from the leads, $I = \sum_s \vec{J}^s_{dev}$. When no bias voltage is applied, the device is at equilibrium and all Fermi levels E^s_F have the same value E_F. When an applied bias V_A is applied at lead s, the Fermi level shifts at lead s and becomes

$$E^s_F = E_F - V_A \tag{2.18}$$

and the current will be non-vanishing.

2.7 Quantum Computing

Quantum transport algorithms comes with several numerical challenges. Matrix operations such as matrix inversions, eigenvalue problems, and matrix products are very expensive. Solving for the current-voltage characteristics of a Si nanowire FinFET with a 3nm diameter and a length of 20nm (10 bands tight binding model) using 10 bias points requires 100,000s [33]. With the emerging Quantum Computer (QC) it is hopefully possible to speed up these operations/simulations with use of parallelization and quantum algorithms.

2.7.1 Quantum annealers

One of the two main quantum computing architectures is the quantum annealer (an additional architecture to the quantum annealer and gate-based QC is the adiabatic

QC, which is closely related to the quantum annealer), intended as a tool for tackling optimization problems. The quantum annealer processors naturally finds and returns minimum energy solutions using quantum tunneling between meta-stable states and the ground state (also called quantum fluctuations). States are represented by quantum bits, or qubits. Just like regular bits they can take on binary values, but in a quantum state of superposition. One of the most common and popular models of a optimization problem which can be implemented on the quantum annealer is the Ising model [34], a mathematical model in statistical mechanics for ferromagnetism. The energy Hamilton of the Ising model is formulated below

$$H(\vec{\sigma}) = -\sum_{i=1}^{N} h_i \sigma_i - \sum_{i<j}^{N} J_{i,j} \sigma_i \sigma_j \tag{2.19}$$

where $\vec{\sigma} = (\sigma_1, ..., \sigma_N)^T$ are spin variables and $\sigma_i \in \{-1, +1\}$. The σ_i represent the qubits. Closely related to the Ising model, as we will see in the next section, is the QUBO problem, which is a combinatorial optimization problem. QUBO problems are ideal for the implementation on the quantum annealer. In the next section the QUBO problem is introduced and a formulation is shown that can solve a linear system of equations, which will be used in this project. [35]

2.7.2 Quadratic Unconstrained Binary Optimization

When formulating the QUBO problem, you start of with creating a cost function f, that maps from an n-dimensional binary vector space $\mathcal{B}^n$ onto $\mathcal{R}$. f is defined as [35]

$$f(\vec{q}) = \vec{q}^T \boldsymbol{Q} \vec{q} \tag{2.20}$$

where $\vec{q} = (q_1, ..., q_N)^T$, $q_i \in \{0, 1\}$, and $\boldsymbol{Q}$ is an upper diagonal matrix containing the QUBO coefficients. The problem consist of finding the $\vec{q}_{min}$ that minimizes the cost function f. Following from binary arithmetic $q_i^2 = q_i$ (idempotency condition), the cost function can be written as

$$f(\vec{q}) = \sum_{i=1}^{N} Q_{i,i} q_i + \sum_{i<j}^{N} Q_{i,j} q_i q_j \tag{2.21}$$

where $Q_{i,i}$ are the diagonal elements in $\boldsymbol{Q}$ and $Q_{i,j}$ are the off-diagonal elements. The $Q_{i,i}$ represents linear terms in the cost function and the $Q_{i,j}$ represents quadratic terms. As we can see, the cost function in the QUBO model, eq 2.21, and the Hamiltonian of the Ising model, 2.19, are very similar. They differ in their unknows $\sigma \in \{-1, 1\}$ and $q \in \{0, 1\}$ which are related by the following linear mapping

$$\sigma = 2q - 1 \tag{2.22}$$

In the next section a QUBO problem will be formulated from the Least Linear Squares (LLS) problem to find the solution to a system of linear equations.

2.7.3 QUBO formalism for a LLS problem

The LLS problem can be formulated as follows: Find the vector $\vec{x} \in \mathcal{R}^N$ that minimize $||A\vec{x} - \vec{b}||$ where $A \in \mathcal{R}^{N \times N}$ is a matrix and $\vec{b} \in \mathcal{R}^N$ a column vector. Writing out the $A\vec{x} - \vec{b}$ expression

$$A\vec{x} - \vec{b} = \begin{bmatrix} a_{1,1} & a_{1,1} & ... & a_{1,N} \\ a_{2,1} & a_{2,2} & ... & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & ... & a_{N,N} \end{bmatrix} \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^N \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \tag{2.23}$$

and then taking the L2-norm square we obtain the following expression

$$||A\vec{x} - \vec{b}||_2^2 = \sum_{k=1}^{N} \left[\sum_{i=1}^{N} a_{k,i}x^i - b_k \right]^2 \tag{2.24a}$$

$$= \sum_{k=1}^{N} \left[\left(\sum_{i=1}^{N} a_{k,i}x^i \right)^2 - 2b_k \sum_{i=1}^{N} a_{k,i}x^i + b_k^2 \right] \tag{2.24b}$$

$$= \sum_{k=1}^{N} \left[\sum_{i=1}^{N} \sum_{j=1}^{N} a_{k,i}a_{k,j}x^i x^j - 2b_k \sum_{i=1}^{N} a_{k,i}x^i + b_k^2 \right] \tag{2.24c}$$

$$= \sum_{k=1}^{N} \left[\sum_{i=1}^{N} (a_{k,i}x^i)^2 + 2\sum_{i<j} a_{k,i}a_{k,j}x^i x^j - 2b_k \sum_{i=1}^{N} a_{k,i}x^i + b_k^2 \right]. \tag{2.24d}$$

Solving this problem in a binary format, each x^i would have a binary representation of R bits $q_r \in \{0,1\}$ in the following form

$$x^i \approx c^i \sum_{r=0}^{R-1} 2^{-r} q_r^i - d^i \tag{2.25}$$

such that $x^i \in [-d^i, 2c^i - d^i)$ [5]. d^i and c^i specifies the believed domain that contain x^i and are very important for finding a good solution. After inserting the expression in equation 2.25 into equation 2.24 we get our QUBO model $f(\vec{q})$ for the LLS problem. To

find the QUBO coefficients we have to expand the terms. We begin by expanding the first term inside the square brackets in equation 2.24d

$$\sum_{i=1}^{N}(a_{k,i}x^i)^2 \approx \sum_{i=1}^{N}\left[a_{k,i}^2\left(c^i\sum_{r=0}^{R-1}2^{-r}q_r^i - d^i\right)^2\right] \tag{2.26}$$

$$= \sum_{k=1}^{N}\sum_{i=1}^{N}\left[a_{k,i}^2\left(c^i\sum_{r=0}^{R-1}2^{-r}q_r^i - d^i\right)\left(c^i\sum_{r'=0}^{R-1}2^{-r'}q_{r'}^i - d^i\right)\right] \tag{2.27}$$

$$= \sum_{i=1}^{N}\left[a_{k,i}^2\left((c^i)^2\sum_{rr'}2^{-(r+r')}q_r^iq_{r'}^i - 2c^id^i\sum_r 2^{-r}q_r^i + (d^i)^4\right)\right] \tag{2.28}$$

$$= \sum_{i=1}^{N}\left[a_{k,i}^2\left((c^i)^2\sum_{r\neq r'}2^{-(r+r')}q_r^iq_{r'}^i + (c^i)^2\sum_r 2^{-2r}q_r^i - 2c^i\sum_r 2^{-r}q_r^i + (d^i)^2\right)\right]. \tag{2.29}$$

In the last equality we used the idempotency condition $(q_r^i)^2 = q_r^i$. Now, we expand the second term inside the square brackets in equation 2.24

$$2\sum_{i<j}a_{k,i}a_{k,j}x^ix^j \approx 2\sum_{i<j}a_{k,i}a_{k,j}\left(c^i\sum_{r=0}^{R-1}2^{-r}q_r^i - d^i\right)\left(c^j\sum_{r'=0}^{R-1}2^{-r'}q_{r'}^j - d^j\right) \tag{2.30}$$

$$= 2\sum_{i<j}a_{k,i}a_{k,j}\left(c^ic^j\sum_{rr'}2^{-(r+r')}q_r^iq_{r'}^j - c^id^j\sum_r 2^{-r}q_r^i - c^jd^i\sum_r 2^{-r}q_r^j + d^id^j\right) \tag{2.31}$$

Finally, we insert the binary representation in the third term inside the square brackets in equation 2.24

$$-2b_k\sum_{i=1}^{N}a_{k,i}x^i \approx -2b_k\sum_{i=1}^{N}a_{k,i}\left(c^i\sum_{r=0}^{R-1}2^{-r}q_r^i - d^i\right) \tag{2.32}$$

$$= -2\sum_{ir}b_ka_{k,i}c_i2^{-r}q_r^i + 2\sum_i b_ka_{k,i}d^i \tag{2.33}$$

We get our QUBO model by inserting equations 2.29, 2.31 and 2.33 back into 2.24. We do this to obtain the QUBO coefficients, but instead of equations 2.29 and 2.31 we use a binary expansion of the first term in equation 2.24c.

$$f(\vec{q}) = \sum_k\sum_{ir}\sum_{jr'}a_{k,i}a_{k,j}c^ic^j2^{-(r+r')}q_r^iq_r^j - 2\sum_k\sum_{ijr}a_{k,i}a_{k,j}c^id^j2^{-r}q_r^i + \sum_{ij}a_{k,i}a_{k,j}d^id^j \tag{2.34}$$

$$- 2\sum_k\sum_{ir}b_ka_{k,i}c_i2^{-r}q_r^i + 2\sum_k\sum_i b_ka_{k,i}d^i + \sum_k b_k^2 \tag{2.35}$$

and the QUBO coefficients are

$$Q^{r}_{i,i} = -2\sum_{k}\sum_{j} a_{k,i}a_{k,j}c^{i}d^{j}2^{-r} - 2\sum_{k} b_k a_{k,i}c_i 2^{-r} \tag{2.36}$$

$$Q^{r,r'}_{i,j} = \sum_{k} a_{k,i}a_{k,j}c^{i}c^{j}2^{-(r+r')} \tag{2.37}$$

Note that some terms in equation 2.35 are just constants and will therefor not affect the minimization.

2.8 Domain Decomposition Methods

To be able to compute the linear system of equations on a quantum annealer a DDM is used to reduce the size of the linear system of equations corresponding to the discretized Schrödinger equation with open boundary condition, see equation 2.4. DDMs are also useful when utilizing the parallel capabilities of computers. The choice of algorithm fell on the BCR method, a 1D DDM along the electron transport direction outlined in [2] [36] [37] [38].

2.8.1 Block Cyclic Reduction

First introduced in [39] and also known as "renormalization" in [40], the idea in the BCR method is to exploit the structure of the system of linear equations to use Gaussian elimination to decouple matrix blocks. The matrix blocks corresponds to different layers, and in this way decouple the different layers (see figure 2.8) until only the first and the last layers are connected. To make it a bit clearer, let us look at a rewritten version of equation 2.4 where we have employed a rectangular mesh elements and a "vertical" ordering (see section 3.2), and only injection from the left is considered:

$$\underbrace{\begin{bmatrix} \mathbf{H}^D_{1,1} + \mathbf{\Sigma}_L - E_\beta \mathbf{D} & \mathbf{H}^D_{1,2} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{H}^D_{2,1} & \mathbf{H}^D_{2,2} - E_\beta \mathbf{D} & \mathbf{H}^D_{2,3} & \ddots & \vdots \\ \mathbf{0} & \ddots & \ddots & \ddots & \mathbf{0} \\ \vdots & \ddots & \ddots & \ddots & \mathbf{H}^D_{n_x-1,n_x} \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{H}^D_{n_x-1,n_x} & \mathbf{H}^D_{n_x,n_x} + \mathbf{\Sigma}_R - E_\beta \mathbf{D} \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} \boldsymbol{\phi}_1 \\ \boldsymbol{\phi}_2 \\ \vdots \\ \vdots \\ \boldsymbol{\phi}_{n_x} \end{bmatrix}}_{\boldsymbol{\phi}} = \underbrace{\begin{bmatrix} \mathbf{B}_1 \\ \mathbf{0} \\ \vdots \\ \vdots \\ \mathbf{0} \end{bmatrix}}_{\mathbf{B}} \tag{2.38}$$

The elements in the matrix $\mathbf{H}$ are in themselves matrices of size $n_z \times n_z$ corresponding to layers, which can be seen in figure 2.8, and the elements of $\boldsymbol{\phi}$ and $\mathbf{B}$ are column vectors of size $n_z \times 1$. The injection vector $\mathbf{B}$ has zero entries everywhere, except at the layer corresponding to the left lead, where we have an injection of electrons. The v, s and β indices which is seen in equation 2.4 are not written explicitly for clarity. The matrices are obtained from the finite element method with uniform rectangular

mesh elements, for details see B.2. The diagonal elements in equation 2.38 describe interactions within a layer and off diagonal elements describe interactions between adjacent layers. For convenience, diagonal elements in $\mathbf{H}$ will be denoted $\mathbf{H}_{i,i}$, for example $\mathbf{H}_{1,1} = \mathbf{H}^D_{1,1} + \mathbf{\Sigma}_L - E_\beta \mathbf{D}$, and off-diagonal elements are denoted $\mathbf{H}_{i,\pm i}$, for example $\mathbf{H}_{1,2} = \mathbf{H}^D_{1,2}$.

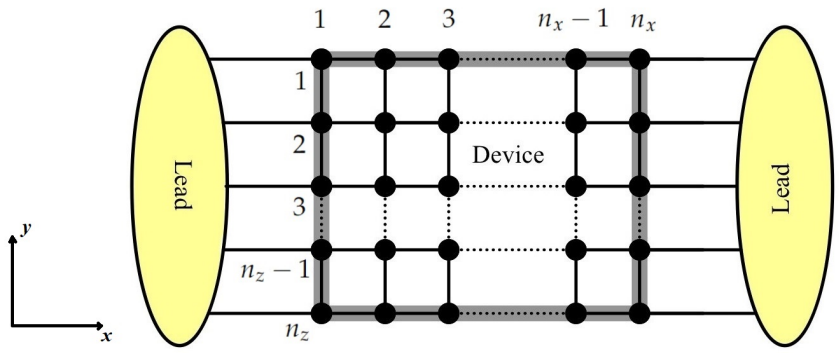


Figure 2.8: Device after discretization. Each dot represents a nodal point which we solve the Schrödinger and Poisson equation for. The vertical nodal points lying under the numbers $1, 2, 3, ..., n_x$ represents the layers.

For the BCR to work properly a number of properties must be satisfied for the system of linear equation $\mathbf{H}$ that we want to solve: (see [2] page 4)

(a) It is block tri-diagonal where each block represents a layer. In our case this means that all nodal points in a layer have the same coordinate in the electron transport direction x.

(b) Following from the first property, the diagonal blocks $\mathbf{H}_{i,i}$ contain energies and only interactions within the same layer.

(c) Off-diagonal blocks $H_{i,i\pm1}$ connect one layer to its left and right neighbor, they are sparse, and $H_{i,i+1} = H^\dagger_{i+1,i}$. In our case $H_{i,i\pm1}$ will be real and diagonal, so these properties are satisfied.

(d) The open boundary conditions, $\mathbf{\Sigma}_L$ and $\mathbf{\Sigma}_R$, and the injection vector $\mathbf{B}$, affects only the first and last blocks.

The problem studied in [2] is an atomistic simulation of high electron mobility transistor (HEMT) and tunneling field-effect transistor (TFET) structures. It uses a tight-binding model as a basis for the Schrödinger equation. In this work, finite element shape functions are employed. If electrons flow along the [1 0 0] crystal axis, the atomic layers (as the layers are called in [2]) will not have interactions between its atoms and the diagonal blocks in equation 2.38 will be quasi-diagonal, which is ideal for the BCR method as we will see. Our discretization do not lead to the same nice structure, our diagonal blocks

are sparse tri-diagonal matrices, but it will not pose a problem to the implementation since it satisfies all the properties listed above. Now, let us turn our attention to the implementation of the BCR method.

As mentioned in the beginning of the section, the BCR method uses Guassian elimination to decouple layers until only the first and last layer are coupled to each other. If we want to decouple layer i the blocks $\mathbf{H}_{i-1,i-1}$, $\mathbf{H}_{i+1,i+1}$, $\mathbf{H}_{i-1,i+1}$ and $\mathbf{H}_{i+1,i-1}$ will be "renormalized" as follows:

$$X_i = \mathbf{H}_{i,i}^{-1}\mathbf{H}_{i,i-1} \tag{2.39}$$

$$Y_i = \mathbf{H}_{i,i}^{-1}\mathbf{H}_{i,i+1} \tag{2.40}$$

$$\hat{\mathbf{H}}_{i-1,i-1} = \mathbf{H}_{i-1,i-1} - \mathbf{H}_{i-1,i}X_i \tag{2.41}$$

$$\hat{\mathbf{H}}_{i+1,i+1} = \mathbf{H}_{i+1,i+1} - \mathbf{H}_{i+1,i}Y_i \tag{2.42}$$

$$\hat{\mathbf{H}}_{i-1,i+1} = -\mathbf{H}_{i-1,i}Y_i \tag{2.43}$$

$$\hat{\mathbf{H}}_{i+1,i-1} = \hat{\mathbf{H}}_{i-1,i+1}^{\dagger} \tag{2.44}$$

The last equation follows from the fact that $H_{i,i+1} = H_{i+1,i}^{\dagger}$. The $\hat{\mathbf{H}}$ denotes the matrix $\mathbf{H}$ after renormalizaton. These six operations suppresses the atomic layer i from $\mathbf{H}$. It is possible to suppress all layers except the first and last layer in this fashion. After this is done, only the first and last layer will be connected and we are left with a $2n_z \times 2n_z$ linear system of equations. So instead of solving a $n_x n_z \times n_x n_z$ system of linear equations, we solve $n_x - 2$ inverses of size $n_z \times n_z$ and a $2n_z \times 2n_z$ linear system of equation. Reducing the size of the linear system of equation we want to sovle, which was exactly what we wanted! But the BCR method also comes with other advantages, and to exploit them fully we have to look at the ordering with which we decouple layers. The renormalization can be devided into three stages:

1. Blocks with even indices are decoupled (2, 4, 6, ...), with exception of the last layer if that has an even index. This requires the inversion of symmetric tri-diagonal real matrices and multiplication of dense and diagonal matrices.

2. Half of the odd indexed blocks are decoupled (3, 7, 11, ...). This includes inversion and multiplication of real dense matrices.

3. The remaining half of blocks with odd indices are removed (5, 9, 13, ...). All matrices in this process are dense. This process is repeated until only the $\hat{\mathbf{H}}_{1,1}$, $\hat{\mathbf{H}}_{n_x,n_x}$, $\hat{\mathbf{H}}_{1,n_x}$, and $\hat{\mathbf{H}}_{n_x,1}$ matrix blocks are left.

After all the steps above are completed, the 2 block $\times$ 2 block remaining system of equations is solved and we find $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_{n_x}$. The rest of the solution vector $\boldsymbol{\phi}$ is reconstructed with the following equation:

$$\boldsymbol{\phi}_i = -X_i\boldsymbol{\phi}_{i-l} - Y_i\boldsymbol{\phi}_{i+r}. \tag{2.45}$$

Before decoupling the layer i, it was connected to layers $i - l$ and $i + r$. So, layer i is reconstructed from the two layers it was connected to before decoupling. In this way we can recursively reconstruct the full solution $\boldsymbol{\phi}$.

3 Methodology

The device simulator can be visualized with a flowchart, which can be seen on the following page in figure 3.1. In the upcoming sections in this Methodology chapter, the different nodes of the flowchart are described. The goal is to explain how one obtains the I-V curve for a given device. A lot of focus will be put on the "Solving the Schrödinger eq." node, since it includes the implementation of the BCR method and the use of the quantum annealer to compute the matrix inversions in the BCR method. We start of with a short section on the conventions and assumptions used.

3.1 Conventions and assumptions

The x-, z-, and y-coordinates denotes the transport, confinement, and out-of-plane directions respectively. Simulations are done in the 2D x-z plane of the device with the effective-mass approximation, limiting calculations to the first conduction band of Si, approximating the conduction band minima with six equivalent ellipsoidal valleys (see appendix A). Translational invariance is assumed in the y-direction, which is a god assumption in wide devices. The Schrödinger equation and Poisson equation are solved self-consistently to determine the electrical behavior of the device. The programming language used is Python. Devices with only one drain and one source are studied.

3.2 Setting up matrices

After discretization of the Schrödinger equation and imposing the open-boundary condition with the QTBM (see appendix B.1.5), you end up with a sparse linear system of equation $\mathbf{H}\boldsymbol{\phi} = \mathbf{B}$, seen in equation 2.4 and also re-written again below

$$\underbrace{[\mathbf{H}^D + \mathbf{\Sigma}_L + \mathbf{\Sigma}_R - E_\beta^v \mathbf{D}]}_{\mathbf{H}} \underbrace{\boldsymbol{\phi}_\beta^{s,v}}_{\boldsymbol{\phi}} = \underbrace{\mathbf{B}_\beta^s}_{\mathbf{B}}.$$

But before you can set up equation 2.4, you first have to construct the $\mathbf{H}^D$ matrix. The $\mathbf{H}^D$ matrix is used to compute the energy spectrum and after that, the open-boundary condition can be obtained. The finite element method, using rectangular mesh elements (see appendix B.1.5 and B.1.5.1) is used to construct the $\mathbf{H}^D$ matrix. Each row in $\mathbf{H}^D$ has at most five nonzero elements and it can be written in the following form

$$(\mathbf{H}^D\boldsymbol{\phi})_{ij} = a_{ij}\phi_{ij} - b_{ij}\phi_{i-1j} - c_{ij}\phi_{i+1j} - d_{ij}\phi_{ij-1} - e_{ij}\phi_{ij+1} \tag{3.1}$$

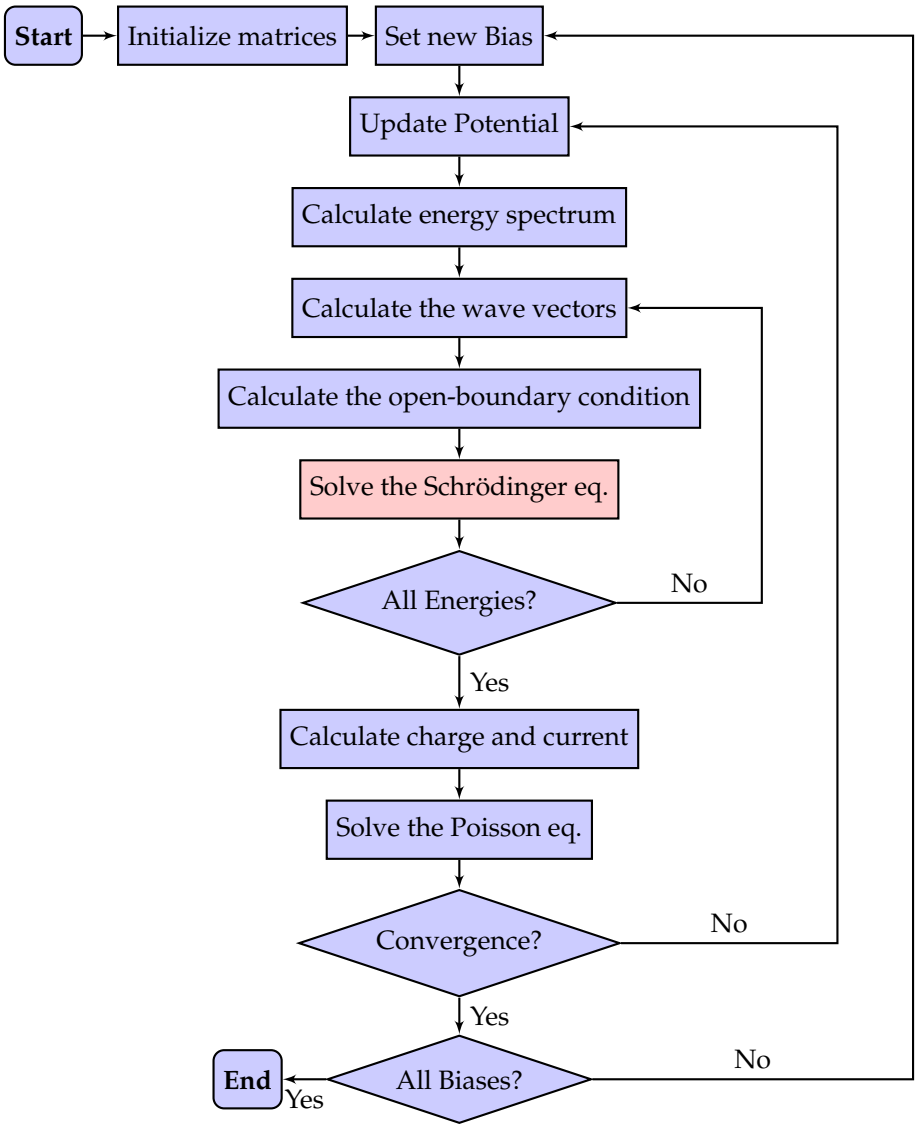


Figure 3.1: Flowchart of the device simulator.

where $i = 1,...,n_z$ and $j = 1,...,n_x$ denotes the 2D spacial position in the discretization mesh, ϕ_{ij} is the wave function in position ij, a_{ij} describes on site energies, b_{ij}, c_{ij} describes interactions with neighbouring sites in the z-direction, and d_{ij}, e_{ij} describes interactions with neighbouring sites in the x-direction. When implementing this on the computer, you have to go from the 2-dimensional indices ij to a 1-dimensional index l. There exist several different such mappings (orderings), for instance natural ordering and red-black ordering, which will give different structures to the $\mathbf{H}^D$ and $\mathbf{H}$ matrices. Here, the following ordering is employed to allow us to use the BCR method and write equation 3.2 in the block tri-diagonal form seen in equation 2.38

$$l(i,j) = i + (j-1) \cdot n_z. \tag{3.2}$$

We are basically going one layer at the time, see figure 2.8, from the top to the bottom of the device. We will call this the "vertical" ordering. This ordering will give the following structure to the $\mathbf{H}^D$ matrix, looking at a arbitrary row $r(i,j)$

$$\mathbf{H}_r^D = [\vec{0} \quad \underbrace{-d_r}_{r-nz} \quad \vec{0} \quad \underbrace{-b_r}_{r-1} \quad \underbrace{a_r}_{r} \quad \underbrace{-c_r}_{r+1} \quad \vec{0} \quad \underbrace{e_r}_{r+nz} \quad \vec{0}] \tag{3.3}$$

where the column indices are written in the underscores. In a matrix format, neglecting the open-boundary conditions, it can be written as

$$\begin{bmatrix}
a_1 & -c_1 & 0 & ... & -e_1 & 0 & ... & & 0 \\
-b_2 & a_2 & \ddots & 0 & ... & \ddots & 0 & \ddots & \vdots \\
0 & \ddots & \ddots & -c_{n_z-1} & 0 & ... & -e_{n_z-1} & \ddots & \\
\vdots & \ddots & -b_{n_z} & a_{n_z} & 0 & \ddots & & \ddots & \\
-d_{n_z+1} & 0 & & 0 & a_{n_z+1} & -c_{n_z+1} & & & \\
0 & \ddots & \ddots & & -b_{n_z+2} & \ddots & \ddots & & \\
\vdots & \ddots & -d_{n-2} & ... & 0 & \ddots & a_{n-2} & -c_{n-2} & 0 \\
\vdots & & 0 & -d_{n-1} & ... & 0 & -b_{n-1} & a_{n-1} & -c_{n-1} \\
0 & ... & & 0 & -d_n & ... & 0 & -b_n & a_n.
\end{bmatrix} \tag{3.4}$$

This matrix is block-tridiagonal and can be written as in equation 2.38. The different block matrices seen in equation 2.38 will have the following structure

$$\mathbf{H}^D_{r,r} = \begin{bmatrix} a_{1+(r-1)\cdot n_z} & -c_{1+(r-1)\cdot n_z} & 0 & 0 & ... & 0 \\ -b_{2+(r-1)\cdot n_z} & a_{2+(r-1)\cdot n_z} & -c_{2+(r-1)\cdot n_z} & 0 & \ddots & \vdots \\ 0 & -b_{2+(r-1)\cdot n_z} & a_{2+(r-1)\cdot n_z} & -c_{2+(r-1)\cdot n_z} & 0 & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & ... & 0 & -b_{r\cdot n_z-1} & a_{r\cdot n_z-1} & -c_{r\cdot n_z-1} \\ 0 & ... & & 0 & -b_{r\cdot n_z} & a_{r\cdot n_z} \end{bmatrix}, \tag{3.5}$$

$$\mathbf{H}^D_{r,r-1} = \begin{bmatrix} -d_{1+(r-1)\cdot n_z} & 0 & ... & & 0 \\ 0 & -d_{2+(r-1)\cdot n_z} & 0 & ... & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & ... & 0 & -d_{r\cdot n_z-1} & 0 \\ 0 & ... & & 0 & -d_{r\cdot n_z} \end{bmatrix}, \tag{3.6}$$

$$\mathbf{H}^D_{r,r+1} = \begin{bmatrix} -e_{1+(r-1)\cdot n_z} & 0 & ... & & 0 \\ 0 & -e_{2+(r-1)\cdot n_z} & 0 & ... & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & ... & 0 & -e_{r\cdot n_z-1} & 0 \\ 0 & ... & & 0 & -e_{r\cdot n_z} \end{bmatrix}, \tag{3.7}$$

The values of the matrix elements are calculated following section B.2. The open boundary conditions, $\mathbf{\Sigma}_L$ and $\mathbf{\Sigma}_L$, will be full $n_z \times n_z$ matrices which are calculated with equation B.41 from the QTBM appendix B.

As stated in section 2.3.1, zero-value Dirichlet boundary condition are employed on the Γ_0 boundary, i.e. the wave-function is vanishing outside the device. To enforce this boundary condition one can set the rows r in $\mathbf{H}$ corresponding to this boundary to zero, with exception to the diagonal elements, which we set to 1. The injection vector $\mathbf{B^m}$ will be zero everywhere except for the rows that pertains to the device-lead boundary, so no modification is necessary there. This procedure leads to the insertion of the trivial equation $1 \cdot \phi_r = 0$ in the system at every boundary index. However, it is often desirable to remove these trivial equations before solving the system of equations, for two reasons. For starters, our matrix H will no longer be symmetric because of the existence of the trivial equations, and secondly, why solve for known wave-vectors as if they were unknown? We can remove these trivial equations in the following way. First, we delete the rows corresponding to the boundary in the matrix $\mathbf{H}$ and the rows in the injection vector $\mathbf{B^m}$. We also want to remove the columns that connect to the boundary

wave-function, but before doing this we have to transfer these contributions over to the injection vector. Since our boundary wave-functions are zero we do not have to add anything to the injection vector. After this step we can finally delete the columns related to the boundary in $\mathbf{H}$ and these trivial equations are removed. We are left with a symmetric system of equations with reduced size, going from a $N \times N$ system to a $N_r \times N_r$ system where $N - N_r$ is equal to the number of nodal points on the boundary where the wave-function already is known. [13]

3.3 Bias and potential

The next step is to pick for which applied bias we want to simulate the device. If the goal is to do a I-V (current and voltage) curve, we have to calculate the resulting currents for many applied biases, as we can see in the flowchart in figure 3.1. This process can be sped up through parallelization, computing the different bias points simultaneously.

Once we have chosen an applied bias, we can set the initial potential distribution V. This is done either by using a potential distribution from an old run with a similar setup, or by calculating the charge distribution from a classical expression, imposing charge neutrality [41], and then solving the Poisson equation, see equation 2.13. The resulting potential distribution is then the initial potential fed to the rest of simulator, including calculating the energy spectrum, wave vectors and then solving the Schrödinger equation for all energies. The potential is embedded into the $\mathbf{H}^D$ matrix.

3.4 Discretization of the energy spectrum

In section 2.3.2, the method of choice to discretize the continuous energy spectrum E^v_β was briefly outlined. Using two different closed boundary conditions at the lead-device boundaries, called standing wave boundary conditions in [13] (they are basically the zero-value Dirichlet conditions and Neumann boundary conditions), for the Schrödinger equation, we obtain two eigenvalue problems of the form

$$\begin{aligned} \mathbf{H}^D\psi_\beta &= E^v_\beta \mathbf{D}\psi_\beta \\ \Rightarrow \mathbf{D}^{-\frac{1}{2}}\mathbf{H}^D\mathbf{D}^{-\frac{1}{2}}\mathbf{D}^{\frac{1}{2}}\psi_\beta &= E^v_\beta \mathbf{D}^{\frac{1}{2}}\psi_\beta \end{aligned} \tag{3.8}$$

where $\mathbf{D}^{\frac{1}{2}}$ is defined such that $\mathbf{D} = \mathbf{D}^{\frac{1}{2}}\mathbf{D}^{\frac{1}{2}}$ and $\mathbf{D}^{-\frac{1}{2}}$ is its inverse. In the second line, we have a standard eigenvalue problem $Ax = \lambda x$, where $A = \mathbf{D}^{-\frac{1}{2}}\mathbf{H}^D\mathbf{D}^{-\frac{1}{2}}$, $x = \mathbf{D}^{\frac{1}{2}}\psi_\beta$ and $\lambda = E^v_\beta$. The standing-wave eigensolutions (E_β, ψ_β) discretizes the continuous energy spectrum of the lead+device system. The Dirichlet boundary conditions are imposed following the last paragraph in section 3.2 and the Neumann boundary conditions does not need any modifications to be implemented since they are satisfied when $\Sigma_L = 0$ and

$\Sigma_R = 0$ (see equation B.20). The eigenvalue problem is solved with the *scipy.linalg.eigh* method, but might also be solvable on a quantum computer, see section 4.2.2. We do not need all the energies obtained from equation 3.8, energies much larger than the Fermi level will be suppressed when calculating the charge distribution. We chose a cut-off energy at $5k_bT$ above the Fermi level [41].

For a more detailed description of the standing-wave solutions see Appendix A, section **1. Finding the device normal modes** in reference [13].

3.5 Wave vectors and open-boundary conditions

The wave vectors k_m^i are calculated using equation B.12 in section B.1.2. Using a infinite-well potential when solving equation B.9 we obtain the mode energies E_m^s, given by

$$E_m^s = \frac{\hbar}{2}\frac{m^2\pi^2}{m_{\zeta_s}^* d_s^2}, \tag{3.9}$$

where $m = 1,2,3,....$ The open-boundary conditions are calculated with equation B.41. The integral in equation B.37 are calculated with the Python function *scipy.integrate.simps*. For a description on how the open boundary conditions are obtained with the QTBM, see section B.1.5 in Appendix B.

3.6 Solving the Schrödinger equation

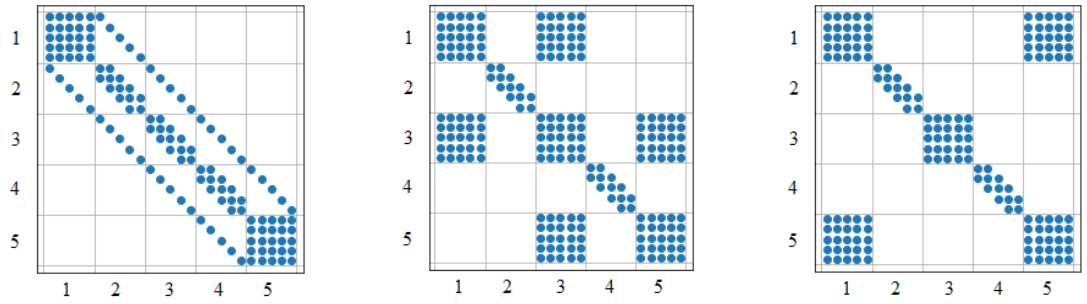
Once we have a potential, calculated the corresponding energy spectrum, the wave vectors and the open-boundary conditions, the next step is to solve the Schrödinger equation. After the discretization, we are left with a system of linear equations, see equation 2.4. The linear system of equations can be solved with an available Python solver, such as the *numpy.linalg.solve* which uses the LAPACK routine `_gesv` [42]. It is based on LU decomposition to solve the system in a reasonable time, but as you have to solve the system for several energies, bias points and iterations to find convergence, the total simulation time will go up a lot. As more nodal points are used in the discretization the system becomes larger and the time to solve the system increases. It is of great interest to speed up this process, which is where the BCR method comes into the picture. The linear system of equations is also too big to solve on present quantum computers, but the BCR method reduces the problem size such that it can be solved on a quantum annealer.

3.6.1 Implementation of the BCR method

The BCR method can be divided into two stages; first one decoupling stage and then one re-coupling stage. The two stages are self-explanatory and are described in the Background, section 2.8.1. The BCR method was developed for solving a similar system

of linear equation to the one that we are left with, there are however some things that differs and needs a special treatment.
One difference between the implementation of the BCR method in reference [36] and in this project is the structure of the matrix which is solved. Their matrix will have diagonal matrix blocks which in themselves are diagonal. (In **Fig. 1.** in reference [37] we can see the structure of their matrix). This is because they do not have connections between elements in each layer. This makes the BCR method super sufficient for their matrix structure since when you compute the X_i and Y_i matrices in equations 2.39 and 2.40 you have to do a matrix inversion of a diagonal matrix, which is trivial. As one can see in equation 3.5 and in figure 3.2, we do not have the same nice matrix structure. This means that the BCR method will not be as efficient for the matrix structure we study, but it will still work (see section 2.8.1). We will have to compute the inverse of tridiagonal matrices instead in the first step, see figure 3.2.

Continuing on the matrix structure, it can be of interest to see how the structure changes when performing the BCR method. This can be seen in figure 3.2 below. What we also can notice in figure 3.2 is that we are dealing with full matrices in all decoupling stages, except the first one. Whereas in [37], full matrices does not appear until in the third decoupling stage.



(a) Before decoupling.

(b) After stage 1. Layer 2 and 4 decoupled.

(c) After stage 2. Layer 3 decoupled.

Figure 3.2: Matrix structure. Non-zero matrix elements are market with blue dots.

A note on the implementation of the BCR method should also be made on the decoupling order, which is very important for the BCR method. First of, we keep the first and last block until the end because those blocks are the only blocks that are full and complex due to the open-boundary conditions. Decoupling these layers in an earlier stage would introduce complex numbers in the decoupling process, and complex arithmetic is 4 times slower than real arithmetic. So, by leaving these layers till the end, we only have to deal with real arithmetic in the decoupling process which is much faster. Also, one of the first or last layer will have a nonzero injection vector, so we do not want to decouple that layer for that reason.

The decoupling order was divided into three stages in section 2.8.1. The first stage only deals with the inversion of sparse tridiagonal matrices. Since it is easier to invert a sparse matrix than a full matrix, (which we are left with in the other decoupling stages, see figure 3.2) we want to decouple as many layers as possible in the first stage. Which is why we decouple every second layer (when decoupling a layer we introduce full matrices in the adjacent layers). Removing every second layer also makes it possible for the decoupling in stage 1. to be run in parallel, this is also true for the decoupling in stage 2.. However, in stage 3. the blocks have to be removed one at the time since the removal of one block will affect the next block in the decoupling order. The layers in the third stage are connected, this is not the case for the layers in the first and second stage which makes it possible to run these stages in parallel.

3.6.1.1 Extra layer BCR

In the re-coupling stage, the whole solution is constructed in a recursive manner where each layer is created from the two layers it was coupled to before removal with equation 2.45. This is not a flawless process since it is not possible to create a nonzero solution if the first and last layer, $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_{n_x}$, which are the two layers that you start the re-coupling stage from, are equal to zero. It can also affect the accuracy of the BCR method if the $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_{n_x}$ layers are much smaller than the rest of the solution that we want to recreate, something we will see in the Results chapter 5. There are however ways to go around this problem.

By keeping one extra layer in the decoupling stage that is likely to be similar in magnitude to the rest of the solution, we can create the solution from this layer instead and achieve a much better accuracy. Several criteria is applicable for the extra layer, outlined below

- It should not be negligible.
- It should correspond to the magnitude of the device wave function.
- The wave function of the extra layer should belong to the modes of the wave function.
- It should correspond to the magnitude of one of the modes of the device wave function.

Preferably, the extra layer should be a layer from the third decoupling stage, since we want to start the re-coupling from that layer. When keeping an extra layer in the decoupling stage you have to solve a $3n_z \times 3n_z$ system of equations instead of a $2n_z \times 2n_z$, which is a drawback of the approach. But you have to compute one less matrix inverse.

The time-consuming part of the BCR method is the inversion of the diagonal blocks in equations 2.39 and 2.40. To try and accelerate these operations a quantum algorithm is developed in the next sections.

3.6.2 Quantum assisted BCR method

To solve the inverses in the BCR method, the quantum assisted QUBO formalism is used, introduced in the Background in section 2.7.2. To simplify the implementation, we employed a Python package called *pyQUBO* [43] [44]. *pyQUBO* provides tools to easily create QUBO models (or Ising) from mathematical expressions, which then can be solved with different samplers. In our case we are interested in the Advantage quantum annealer [4], developed by D-Wave Systems Inc. It is currently the most capable quantum annealer available for use in D-Wave's cloud service Leap, with 5000+ available qubits.

In section 2.7.2 it was shown how a QUBO model can be constructed from the linear system of equations, but now we are interested in the inverse of a matrix. To see how the inverse can be computed, we will look at two different objective functions.

3.6.2.1 The matrix inversion objective function

First of, it is possible to create an objective function based on the following expression

$$||A \cdot B - I_N||, \tag{3.10}$$

where A is a non-singular $N \times N$ matrix, B its inverse and I_N the identity matrix of size $N \times N$. Here we want to find B which constitutes of N^2 different unknown entries. This means that we have to represent B in a binary format and construct our QUBO problem in a similar fashion as in section 2.7.3. We can turn the matrix problem into a linear system of equations

$$A \cdot B = \begin{bmatrix} a_{11} & \dots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \dots & a_{NN} \end{bmatrix} \begin{bmatrix} b_{11} & \dots & b_{1N} \\ \vdots & \ddots & \vdots \\ b_{N1} & \dots & b_{NN} \end{bmatrix} = I_N = \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix} \tag{3.11a}$$

$$\Rightarrow \underbrace{\begin{bmatrix} A & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & A \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} b_{:1} \\ \vdots \\ b_{:N} \end{bmatrix}}_{\mathbf{b}} = \underbrace{\begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix}}_{\mathbf{e}} \tag{3.11b}$$

where $b_{:i}$ denotes the ith column of B and e_i denotes the ith column of I_N. $\mathbf{A}$ is a $N^2 \times N^2$ matrix. Next, we can just follow the procedure of section 2.7.3 to find the QUBO matrix. We will call this the whole matrix approach (or whole matrix solver). This approach is

not optimal when implemented on present day quantum annealers since it requires a binary representation of many unknowns (if we want an accuracy of R qubits, we need a total of $R \cdot n^2$ logical qubits) and we are limited by the number of qubits.

When implementing the whole matrix approach in Python, it is not efficient to turn it into a linear system of equation because it necessitates the creation of a huge matrix **A** with almost all elements equal to zero. Instead, what we did in the project was that we created a general binary model of the B matrix with *pyQUBO*, where each element in B is a binary representation of R binary unknown. An objective function is then created from equation 3.10 with a norm defined as

$$||A|| = \sum_{i}^{N}\sum_{j}^{N}|a_{ij}|^2. \tag{3.12}$$

This approach is preferred since it neglects the creation of the large **A** matrix.

We can also find the inverse of a $N \times N$ matrix A by solving a linear system of equation $Ax = m$ where m is a unit vector, for example $m = e_1 = [1,0,...,0]^T$. This is just a decomposed version of the linearization above. The solution to this linear system of equation with the right-hand-side $m = e_1$ will be the first column of $B = A^{-1}$. So, by solving the linear system of equation for all unit vectors we obtain the inverse of A. This approached is suggested in [5]. We have to solve the linear system of equations N times, where in every solve we are looking for N unknowns. This approach needs a lot less qubits (if we want an accuracy of R qubits, we need a total of $R \cdot n$ logical qubits) and is thus preferred over the approach discussed above. We will call this the unit vector approach (or unit vector solver). The QUBO formalism for the unit vector approach is set up and implemented following section 2.7.3.

3.6.2.2 Choosing the domains

When creating a binary representation of an unknown value x, you have to specify a domain $[-d, 2c - d)$ in which x is believed to reside in, see equation 2.25. The smaller the domain, the more accurate solution you can obtain with fewer qubits. You also have to include the correct solution in your domain for the quantum annealer to be able to find it. Beforehand, you obviously do not know the value of x, so it can be very hard to select a good domain. Especially if the dynamic range is very large. So how do you chose a good domain? You could obviously chose a very large domain and sacrifice the accuracy of the solution. However, from the annealing solution you can specify a new smaller domain around your obtained solution, and compute a new solution with the new domain and get better accuracy. Doing this in iterations you achieve a satisfactory solution.

Another viable approach is to compute a fast, low accuracy classical solution and using that solution as a initial guess for further iterations. In this project, we have the luxury of being able to compute the correct solution to the matrix inverse using the *numpy.linalg.inv* solver, and using this result to specify the domains to the QUBO solver. Setting the correct value v_{cor} in the middle of the domain, and the domain size to the absolute value of the correct value multiplied with a factor f. The resulting domain is the following $[v_{cor} - abs(v_{cor})f, v_{cor} + abs(v_{cor})f]$. In the results, we use a factor of $f = 0.5$.

3.6.2.3 Implementation on the D-Wave annealer

Once the QUBO model is created, it has to be mapped onto the topology of the quantum annealer. The qubits in the D-Wave chip (called physical qubits) are not fully connected (because of fabrication challenges), but are only connected to a limited number of neighbouring qubits [45]. These connections are represented with graphs that makes up the topology of the chip. As an example, we can look at the Chimera topology of the D-Wave 2000Q system, seen in figures 3.3 and 3.4. Here, each physical qubit is connected to four horizontal qubits in each unit cell. The degree of the Chimera topology is 6 (each qubit is connected to 6 different qubits). The Advantage system, employed in this project uses a different topology called Pegasus, that has an improved connectivity with a degree of 15.

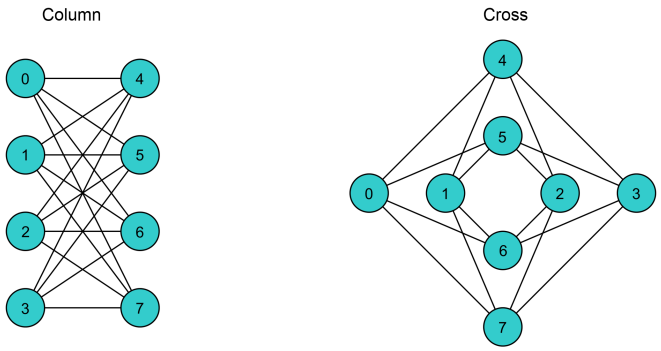


Figure 3.3: Chimera unit cell[1]

[1]Figure is from the Ocean documentation, URL: https://docs.ocean.dwavesys.com/en/stable/concepts/topology.htmltopology-sdk [45]

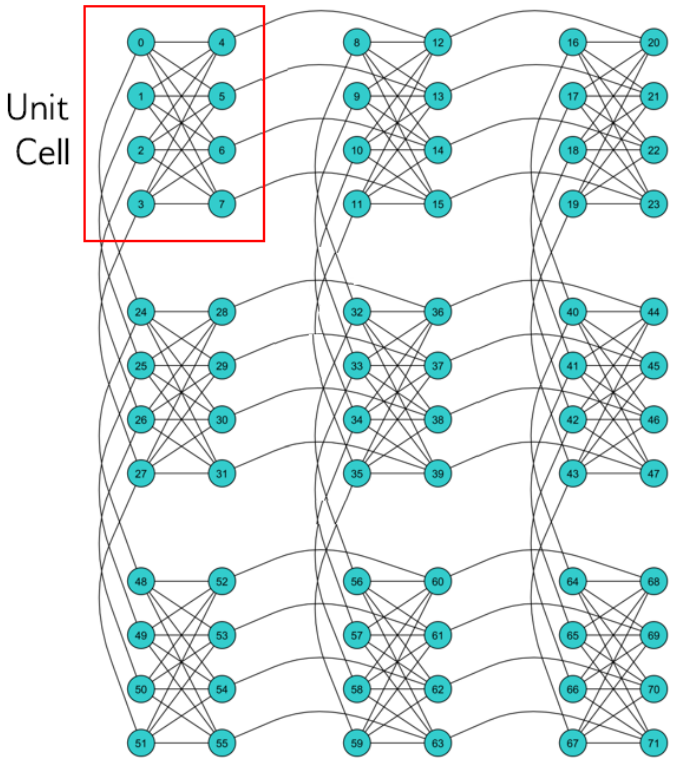


Figure 3.4: A 3×3 Chimera graph[2]

Our QUBO model might have a need for a higher connectivity of the qubits, than that provided by the QPU topology. To solve this problem, one logical qubit (the qubits of our qubo model) can be represented by a chain of physical qubits. There usually exist several ways to create chains of physical qubits and embed the QUBO model onto the QPU topology. To find the most efficient use of the physical qubits when creating an embedding can be a hard problem, and it is called *minor embedding*. D-Wave uses an heuristic tool called minorminer [46], which is also employed when finding the embedding for our QUBO models.

Once the embedding has been found we can use the samplers provided by the D-Wave Ocean software [47] to sample solutions to the QUBO model on the Advantage annealers.

3.7 Charge and current

The charge distribution is calculated after we have found the wave-functions $\phi_\beta^{s,v}$ for all the energies using equation 2.11. First, we normalize the wave-functions using equation 2.7. The integral is calculated with the Simpson method. The Fermi-Dirac integral is calculated with the open source Python package FDINT, created by Scott J. Maddox, based on the work by Fukushima [48] [49] [50]. If the potential has converged we can also calculate the current with equation 2.17 for the device.

[2]Figure is from the Ocean documentation, URL: https://docs.ocean.dwavesys.com/en/stable/concepts/topology.htmltopology-sdk [45]

3.8 Solving the Poisson equation

The Poisson equation 2.13 is solved in the same way as the Schrödinger equation, i.e. we discretize the PDE using the finite element method with the same rectangular mesh as with the Schrödinger equation. A linear system of equations is obtained, see equation 2.16. **P** in this linear system of equations will have the same structure as the matrix in equation 3.4 and is obtained in a very similar way. The elements in the matrix are calculated with equation B.32, where we replace the reciprocal mass $\mathbf{M}^{-1}$ with the permittivity $\epsilon(x,z)$ and remove the $\frac{\hbar}{2}$ factor. The boundary conditions are zero normal derivative as mentioned in section 2.4.1 and no modifications are needed to incorporate them. The linear system of equations can be solved with the BCR method, but as it is only necessary to solve the Poisson equation once every convergence iteration, it will not lead to a massive speed up. Therefore we solve the Poisson equation with the *numpy.linalg.solve* solver.

3.9 Self consistency

When the new potential distribution V is calculated, we have completed one inner iteration. The next step is to check for convergence. If the error between the "old" potential V_{old} fed to the inner iteration and the "new" potential V_{new} obtained in the previous step (section 3.8) is larger than some predefined value ϵ, the inner iteration is repeated with the "new" potential as the "old" potential. Otherwise, the "new" potential is the self-consistent solution that describes the device correctly with the applied bias is found. When this happens, we calculate the current and go on to the next bias point, using the previous self-consistent potential as the initial guess for the potential. This continues until all our bias points have been simulated and we can finally plot our I-V curves, which marks the end for the Methodology chapter. It should be mentioned once again here in the methodology that we currently have not succeeded with finding self consistent solutions, so this part is left as a future work of the project.

4 Literature Review

Modeling of a device can usually be divided into three stages:

- Selection of an physical model. The physical model is often a simplified version of a full description of the problem, but important physics is kept that is relevant for the simulation. Complexity of the model is usually reduced with assumptions or empirical parameters.
- Discretization and mathematical models transforms the physical equations into linear algebra problems.
- The resulting linear algebra problems are solved with computing and numerical algorithms, such as domain decomposition methods and high-performance computers (quantum computers).

In this literature review we will go through physical models, discretization schemes and numerical algorithms considered in the making of this project. Starting out with the nanoscale models.

4.1 Nanoscale models

The challenge of simulating nanoscale transistors have been approached by several different methods. Simulating methods can be divided into macroscopic and microscopic simulation schemes, the difference being the primitive element that forms the theory. Macroscopic models treats *electron populations* whereas the microscopic models considers *single electrons* [18]. Additional subdivisions can also be made. The microscopic models can be divided into classical and quantum models, the distinction being if the individual electrons can be localized in phase space or not. As the size of the electron's wave function becomes comparable to the device size, the electrons are described with their wave functions (quantum) instead as point particles (classical). The macroscopic models can in turn be separated into lumped and continuum theories, depending on the size of the electron population. The methods division can be seen in figure 4.1.

Table 4.1: Table of different electron transport methods. Table is copied from table 1 in [18].

Macroscopic		Microscopic	
Lumped	Continuum	Classical	Quantum
Equivalent circuits, transmission lines	Drift-Diffusion, density-gradient	Semi-classical electron dynamics, Boltzmann transport	Schrödinger, density-matrix, Wigner function, NEGF, PME

The choice of model is dependent on the validity of the model at the considered length scale, and the computational complexity of the model. In figure 4.1, a number of different models are entered into a flowchart to visualize the range of validity and complexity. Model selection is also a question about the physical effects that is under investigation. At the sub-10 nm we enter the quantum mechanical regime, where we have quantum effects such as tunneling, confinement and scattering (see section 2.2.1). These can be modeled with true quantum transport models, or with modified classical models. In the following sections, we outline some of the approaches to model quantum effects in transistors.

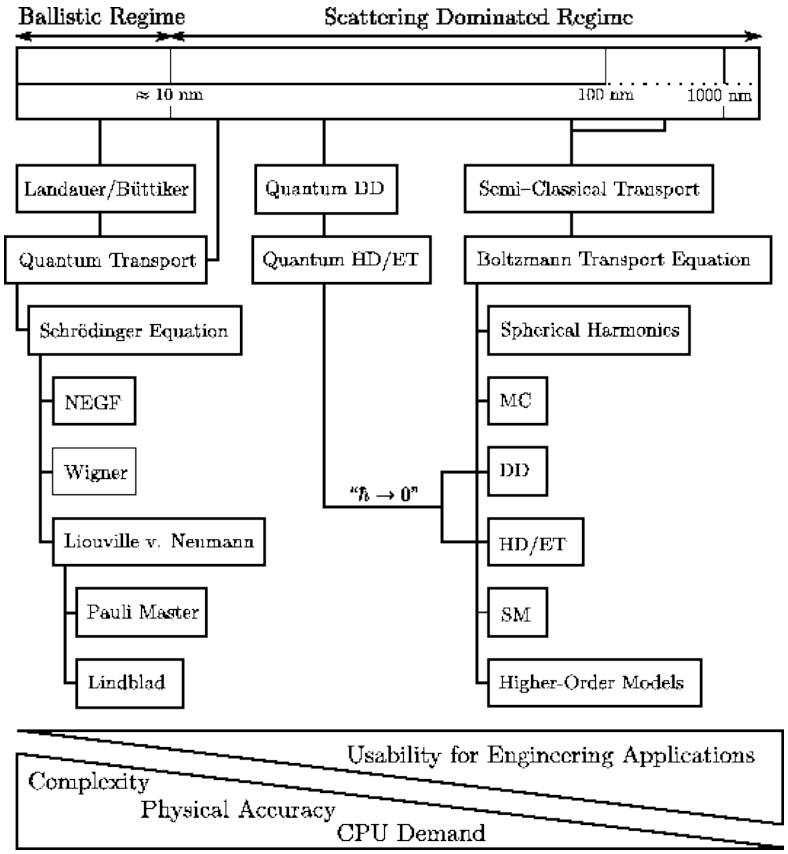


Figure 4.1: Model hierarchy. See [51] for a description of the models not discussed in this thesis. [1]

[1] Figure 1.1 in M. Vasicek dissertation *Advanced Macroscopic Transport Models*. URL: https://www.iue.tuwien.ac.at/phd/vasicek/node3.html

4.1.1 Macroscopic Approaches

One of the oldest and most used macroscopic approaches is the DD method, formulated in the beginning of the 1950s by Shockley and Van Roosbroeck [52] [53]. It is based on the moments of the Boltzmann Transport Equation (BTE). The DD simulation, equipped with the full-band Monte Carlo (FBMC), can successfully be used for characterization of sub-100 nm MOSFETs [54] [21]. However, it is a classical model and cannot be used for quantum modeling.

Efforts have been made to implement quantum effects in macroscopic device simulators as the device size is shrinking. One approach is based on the moment equations derived from a closed-system quantum corrected Wigner distribution function and is called the DG method [18]. The Wigner distribution satisfies a microscopic transport equation derived from the quantum Liouville equation, where a quantum correction term is added. Calculating the first-order moment of this transport equation, we get a macroscopic current continuity equation which is employed for the density gradient method. The DG method is embedded into the DD simulation to compensate for the statistical quantum effects, such as quantum confinement. See figure 2.5 for an example when the density gradient method is used to model quantum confinement. [20]

4.1.2 Microscopic Approaches

The most used microscopic transport model is the BTE. The solution of the BTE is a carrier distribution function $f(\mathbf{r}, \mathbf{p}, t)$ for position $\mathbf{r}$, momentum $\mathbf{p}$, and time t, satisfying the following microscopic equation [51]

$$\frac{\partial f}{\partial t} + \frac{\mathbf{p}}{m^*} \cdot \nabla f - \nabla V \frac{\partial f}{\partial \mathbf{p}} = Q_{coll}(f) \tag{4.1}$$

where m^* is the electron effective mass, V the is the electrostatic potential, and $Q_{coll}(f)$ is a collision term accounting for scattering. At equilibrium, considering fermions, it takes the form of the Fermi-Dirac function. The distribution function is classical since the Heisenberg's uncertainty principle is neglected, which means that position and momentum are known at the same time. Therefore, it is not suitable for the treatment of quantum transport. [51]

In quantum theory a physical state of a closed system is specified by a wave function describing the probability of measuring the state. An alternate description of quantum theory is with the use of a quantum mechanical potential, introduced by Bohm [55]. The quantum potential is a less popular theory in the physics society as it leads to nonlocality, but favorable from a practical point of view since it can easily be implemented in the standard Monte Carlo technique [20]. The electron is thus guided with both the classical electrostatic potential and the quantum potential. This technique can account for static

quantum interference effects due to the nonlocal wave nature of the particle.

Depending on the assumptions and approximations used in the its derivation, many different varieties of quantum potential can be obtained, such as the DG method we saw in the previous section, the quantum moment method [56] and smoothed effective potential [57]. As stated, static quantum affects caused by the potential profile inside the device can be accounted for with the quantum potential, and this is natural because it leads to the closed system Schrödinger equation for electrons [20]. But because it assumes a closed system it is not easy to account for dynamical quantum effects associated with phase randomizing scattering processes.

To capture the dynamical quantum effects it is necessary to solve the quantum transport equation for the entire system and its environment, such as interactions with the other electrons, phonons, and surface roughness. In figure 4.2 below the microscopic approaches are divided into ballistic and diffusive transport, and a more general regime able to handle both.

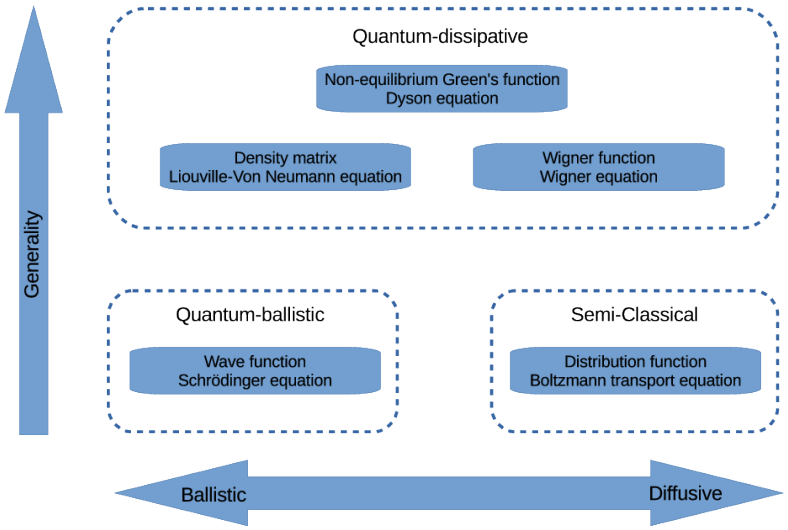


Figure 4.2: The microscopic transport models. Distinctions are made between models that only can describe ballistic or diffusive transport, the top box contains the general models which can describe transport in both regimes. [2]

Before going into the methods that can handle the dynamical quantum effects we should mention the quantum model used in this project, namely the Schrödinger equation with open-boundary conditions that is outlined in section 2.3 and reference [13]. We will call this approach the Wave Function (WF) formalism. The open-boundary conditions permits the modeling of current flow in the device when driven far from equilibrium, exchanging electrons with the environment. This physical model assumes ballistic transport and neglects scattering. However, the model is readily extendable with one of the

[2] Figure 1.1 in P. Ellinghaus dissertation *Two-Dimensional Wigner Monte Carlo Simulation for Time-Resolved Quantum Transport with Scattering*. URL: https://www.iue.tuwien.ac.at/phd/ellinghaus/html_diss_cse2.html#x11-100001.2

dynamical quantum models which soon will be touched upon, called the Pauli master equation [29].

There are a couple of methods which can handle the dynamical quantum effects, for example the density matrix (Wigner distribution[3]) approach, master equation approach, and non-equilibrium Green's function approach. They are essential tools for identifying the shortcomings of the semi-classical BTE approach, and the foundation for the next generation of simulation instruments. But as they are much more physically involved, the computational complexity increases.

4.1.2.1 Wigner function

First off, we have the Wigner distribution function $f_W(\mathbf{r}, \mathbf{p}, t)$. Used in areas such as quantum transport, quantum chemistry (for calculating static and dynamical effects in many-body systems), and signal processing (for investigating waves passing through different media). For non-dissipative transport (no dissipative scattering) the Wigner distribution function satisfies the following equation of motion

$$\frac{\partial f_W}{\partial t} + \frac{\mathbf{p}}{m^*}\nabla f_W - \frac{1}{h^3}\int d^3p' W(\mathbf{r}, \mathbf{p}') f_W(\mathbf{r}, \mathbf{p} + \mathbf{p}') = 0 \tag{4.2}$$

where $W(\mathbf{r}, \mathbf{p}')$ is the Wigner potential, defined in [58], h the Planck constant. The Wigner function f_W is derived from the Liouville-Von Neumann equation for the density matrix using the Wigner-Weyl transformation [59]. The Wigner distribution function is a quantum mechanical description in phase space, hence a particle cannot be localized in phase space which means that the position and momentum cannot be known at the same time (compare to BTE). Because of this the distribution is not positive definite, and can therefor not be regarded as a distribution function in the classical sense. Instead observables are derived from the Wigner distribution function [51]. Because it is based on quantum mechanics it can be used as a quantum transport model.

The transport equation (equation 4.2) is either solved deterministically through discretization (finite-difference-, finite-element-, or spectral-methods) or stochastically with the kinetic Monte Carlo methods [58]. The Wigner function can also be used to treat dissipative scattering, such as electron-phonon interactions [60], with an inclusion of the scattering term in the Hamiltonian. That being said, it is far from straightforward and remains a challenging task to include scattering into the Wigner function formalism. For an overview of the use of Wigner functions and recent advances, see [58].

[3]This Wigner distribution is for an open system, not the closed system associated with the quantum potential.

4.1.2.2 Non-equilibrium Green's function (NEGF)

The Non-Equilibrium Green's Function (NEGF) formalism was developed independently by Kadanoff-Baym [61] and Keldysh [62] in the 1960s. Since the 1990's it has been one of the most efficient techniques to simulate quantum transport at the nanoscale. It presents an alternative to the OBC Schrödinger to describe the non-equilibrium transport. Several Green's functions exist, all related to each other, including the retarded and advanced Green's function. The retarded Green's function $\mathbf{G}^R$ can be computed from the following equation

$$[\mathbf{H}^D + \Sigma_L + \Sigma_R - E\mathbf{D}]\mathbf{G}^R = \mathbf{I} \tag{4.3}$$

where $\mathbf{I}$ is the identity matrix. Comparing this equation with equation 2.4, we see that the retarded Green's function acts as the inverse of the []-factor multiplied with the wave vector $\boldsymbol{\phi}$. So the retarded Green's function and the wave vector $\boldsymbol{\phi}$ are related as [63]

$$\mathbf{G}^R\mathbf{B} = \boldsymbol{\phi} \tag{4.4}$$

Additional Green's functions are the lesser- $\mathbf{G}^<$ and greater- $\mathbf{G}^>$ Green's functions. They describe correlations between different states of the electrons and are used to account for scattering and the open boundary conditions. They are obtained from the following equation [63]

$$\mathbf{G}^{\gtrless} = \mathbf{G}^R\Sigma^{\gtrless}\mathbf{G}^A \tag{4.5}$$

where $\Sigma^{\gtrless}$ are the lesser and greater self-energies that describe the probability for in-scattering ($<$, unoccupied state is filled) and out-scattering ($>$, occupied state is emptied). The open boundary conditions and scattering are including in the self-energies. $\mathbf{G}^A$ is the advanced Green's function, equal to $\left(\mathbf{G}^R\right)^\dagger$.

The advantage of the NEGF formalism is that it is possible to compute the charge density n and current I directly from the Green's functions, without having to evaluate $\boldsymbol{\phi}$ as an intermediate step[63].

4.1.2.3 Pauli master equation (PME)

The Pauli Master Equation (PME) is a less conventional approach to threat scattering, outlined in [30], [31], [29]. It is a Markovian class of master equations that describes transition between quantum states, thereby, the evolution in time of an irreversible open system. Similarly, it can be used to describe the time evolution of the density matrix corresponding to a device connected to external reservoirs. If the coherence length of the injected electrons are longer than the device size, the electrons are highly delocalized and the off-diagonal elements of the density matrix, corresponding to electron coherence, can be neglected [30]. This assumption simplifies the computational complexity of the PME, allowing a treatment of relatively large realistic open systems in an efficient way,

but is also a limitation of the method.

The scattering states that diagonalize the density matrix is determined from the Schrödinger equation in the 2D plane of the device using the effective mass approximation[29], which is the physical model used in this project, allowing us to include scattering into our model with the extension of the PME. The use of a natural basis of open-system wave functions in the PME allows an inclusion of quantum effects directly, such as tunneling and confinement, unlike the BTE approach.

Transition probabilities between quantum states, obtained with Fermi's golden rule, are incorporated into the PME to obtain the steady-state distribution of electrons away from the purely ballistic picture. The system is considered to be in a steady state, even if an applied bias is causing an exchange of particles with the contact reservoirs. The final state of the system is obtained by self consistently solving the Schrödinger equation, PME and Poisson equation. In [29], the impact of electron-phonon and surface-roughness scattering in silicon ultrathin-body double gate FETs are analyzed theoretically with the PME. The results show that electron transport is mainly dissipative even in small devices. The main source of dissipation being the surface roughness, severely reducing the ballistic source-drain current.

4.1.3 Comparisons quantum microscopic approaches

Three true quantum models have been presented in the above section, the Wigner distribution function formalism, the NEGF formalism, and the PME, an extension of the WF formalism used in this project. But how does the methods compare?

In the ballistic picture, it is more computational efficiently to use the WF formalism than the NEGF [64]. It is easier to solve a linear system of equations with N unknowns than to compute a matrix inverse with $N \times N$ unknowns, see equations 2.4 and 4.3. Next, we can consider the open boundary conditions. In the PME approach used in [29] contacts between the open system and the reservoirs are modeled using the QTBM. The QTBM is preferred over the NEGF-based approaches for two reasons: (1) QTBM yields the eigenstates of our open system directly (2) exploiting the eigenstates of the contact, QTBM is numerically more efficient than NEGF.

Considering dissipative scattering, the PME provides an efficient approach to account for scattering processes at steady state, even including a tranfer-wave-vector-dependence. However, because it uses Fermi's golden rule, it is only valid when the perturbation is considered weak. Next, the off-diagonal elements of the density matrix can only be neglected for short devices. Although, a treatment of off-diagonal elements is possible, this leads to an intractable numerical problem with similar complexity as the inversion

of the scattering self-energy matrix in the NEGF formalism [29].

The NEGF is increasingly applied and the standard approach for quantum transport. However, in Ferry et al.'s review of quantum transport in field-effect transistors [17] the preference is the Wigner function formalism. The similarities between the classical particle approach and the Monte Carlo simulation of the quantum Wigner function provides an efficient method to explicitly study the importance of quantum effects. It presents a clear study of each process and its importance in the behaviour of the overall system. As cited from [17], "the Wigner function allows one to clearly identify the quantum effects". Critique of the NEGF partly stems from the treatment of the source and drain contacts. The approximation of only treating the device region quantum mechanically and coupling it to simple self-energies corresponding to the leads ignores many important effects. This can lead to simulations that are far from reality. More critique of the NEGF formalism is outlined in section 3.3 in [17].

4.1.4 Discretization techniques and basis sets

Next, we will consider different discretization techniques. To solve the physical models numerically, the mathematical equation must be discretized by expanding the wave functions $\phi(r)$ in a basis set μ_σ.

$$\phi(r) = \sum_\sigma c^\sigma \mu_\sigma(r) \tag{4.6}$$

The unknowns are the coefficients c^σ that we want to find by solving either a linear system of equations or an eigenvalue problem. In this project a real-space discretization with rectangular mesh elements where used, so the basis set is equal to the shape functions introduced in section B.1.5 and defined in B.2. But it is also possible to discretize the problem in another basis, such as the mode-space expansion used in [65]. There, the real-space discretization in the confined z-direction is replaced with subband eigenfunctions (modes) of a 1D z-directed effective mass Schrödinger equation, replacing the 2D real-space domain with a 1D eigenvalue problem in the confined z-direction and a system of coupled 1D Schrödinger equations in the transport direction. In very thin devices, only a few of these subbands will be occupied, reducing the size of the problem from $(n_x \times n_z)^2$ to a $(n_x)^2$ problem for every subband. A similar approach is employed in [66]. In an extension of the work in [66], by Abdallah et al. [67], WKB techniques is used in addition to the subband decomposition, reducing the problem size further. WKB denotes oscillating shape functions, applied for the 1D Schrödinger equation in the transport direction and allowing for a coarser mesh size.

Other basis functions used in the fields of quantum chemistry and solid-state physics are [68]:

- linear combinations of atomic orbitals (LCAO), along with the popular Gaussian local basis set
- plane wave expansion
- real-space mesh techniques, based on finite-difference, finite-element etc.

4.1.5 First-principle calculations

So far in this literature review, we have seen different physical models that attempts to model the quantum effects in nano-scale devices, employing different simplification to the real physical picture. But they suffer from the need of parametrization (empirical models), transferability of the parameters from bulk materials to nanostructures, treatment of heterostructure and the absence of atomic resolution [64].

As an emerging next step in nanoscale modeling are the first-principles calculations. Starting from the underlying mathematical equations that governs the physical laws, instead of using empirical parameters, we can obtain a fundamental and fully comprehensive picture of the system. These first-principles calculations are based on *electronic structure calculations* that decides most of the physical properties of matter through chemical bonding [68]. The most widely used being the density functional theory (DFT), based on the Kohn-Sahm equations [69].

Ever since the development of quantum mechanics, the fundamental laws governing physics has been known (at least at the nm scale). Thus, the difficulty of first-principles calculations does not lie in the problem formulation, but the problem solution. Simulating atom-by-atom in devices with several thousands of atoms using first-principle calculations is tremendously computational heavy. A full many-body Schrödinger equation, treated numerically, leads to a deceptively simple linear eigenvalue problem. However, it scales exponentially with the number of particles, making it intractable [68]. To make first-principle simulations tractable, present massive parallel capabilities of modern high-performance computing (HPC) architecture must be fully exploited. [68]

4.2 Quantum Computing

Quantum computing is an area of research that is currently on a surge. Quantum computers have the potential to outperform classical computers at certain tasks. From scientific computing [70] to encryption (Shor's algorithm [71]). In this project the quantum annealer was used to solve matrix inverses, that resulted from a linear system of equations. Although, no quantum advantage [72] was found in this project, there exist algorithms that exploit the properties of quantum computers to outperform classical algorithms.

4.2.1 Linear system of equation solvers

The HHL algorithm, named after its inventors Harrow, Hassidim and Lloyd [10], efficiently finds a quantum state $|x\rangle$ that is proportional to a vector $\vec{x}$ that is the solution to a linear system of equations $A\vec{x} = \vec{b}$. The size of the matrix A is $N \times N$ and it has condition number κ (the ratio of A's biggest and smallest eigenvalue). Since we find the quantum state $|x\rangle$ of $\vec{x}$, we cannot find $\vec{x}$ explicitly unless we perform N measurements on $|x\rangle$. However, we are often not interested in $\vec{x}$ itself, but some expectation value of $|x\rangle$. The HHL algorithm can be as much as exponentially faster that classical algorithms, when the condition number κ and one over the error $1/\epsilon$ scales as poly $\log(N)$.

The HHL algorithm uses a gate-based quantum computer. The gate-based quantum computers are sensitive to noise and require error correction to run fault intolerant quantum algorithms. The gate-based quantum computer's sensitivity is also limited by the number of qubits, and algorithms that can be implemented are limited by the number of gates applicable in sequence, before suffering of decoherence occurs. Therefore, because of today's limitations of gate-based quantum computers the quantum annealer where used instead in this project. [73]

4.2.2 Eigenvalue solver

A part of the device simulator was to solve two eigenvalue problems to find the discretized energy spectrum, equation 3.8. The eigenvalue problem can also be implemented on the QC. In [74], Wang and Xiang present a new quantum computing algorithm that efficiently can solve eigenvalueproblems for tridiagonal symmetric matrices, with ability to extend to other matrix structures (in our project we have five diagonals in a symmetric matrix that we want to find the eigenvalues for). Provided a good initial guess for the eigenstates of the system, the algorithm cost scales as *poly*(log N), where N is the dimension of the matrix. The cost of classical solvers scales at best as $O(N^2)$.

4.3 Domain Decomposition Methods

The BCR method used in this project and outlined in 2.8.1 is obviously not the only DDM. However, it was the one that we thought was most suitable to this project. In this section we will outline two of the other DDM considered and additional benefits of the BCR method.

In Sho and Odanaka [75], a parallel DDM based on the restricted additive Schwarz (RAS) method is presented for a quantum-corrected drift-diffusion (QCDD) model. The discretization of the QCDD equations leads to a linear system of equations, which is sought to be solved with multiple computing nodes. In the article, two parallelization schemes are outlined, one intranode splitting-up operator and a internode DDM based

on an overlapping Schwarz method. The overlapping Scwarz method decompose the solution domain Ω into a set of m overlapping subdomains $\left\{\Omega_i^\delta\right\}_{i=1}^m$, where δ is the number of overlaps. Each subdomain is solved for with artificial boundary conditions, which are updated iteratively in a block Jacobi fashion. An inter-node parallel speed up of 35.2 was obtained for 64 decompositions of the QCDD problem. The overlapping Schwarz method was an interesting DDM for our project, however it was unclear how the open boundary conditions should be treated. It is also an iterative method, which is not very suitable to our problem where we deal with several right-hand-sides.

The next DDM considered was the SplitSolve algorithm, developed by Calderara et al. [64], with the purpose to accelerate a first-principles quantum transport simulation based on DFT. The SplitSolve algorithm was added as an extension to the OMEN simulator, which also incorporates the BCR method. Similarly to our project they want to solve a linear system of equations $Tc = Inj$ with a right-hand-side injection vector Inj that is zero almost everywhere, except for the parts corresponding to injection from the leads. This means that you do not have to compute the inverse of the entire matrix T^{-1}, it is sufficient to only compute the first and last columns of T^{-1} that is multiplied to Inj. The SplitSolve also decouples the calculations of the open boundary conditions, which are very computational heavy, from the solution of T^{-1}. Additional partitioning of the problem is done with a modified and optimized version of the SPIKE algorithm [76], which act as spatial decomposition. Since it is a DFT simulation, the Hamiltonian matrix will have a different more dense structure compared to our Hamiltonian matrix (see equation 3.4 and figure 3.2), and therefor the Split-Solve algorithm seemed ill fitted for our problem.

4.3.1 BCR

As already mentioned, we chose the BCR method as the DDM of choice. The reasons being many, such as it was easily adaptable to our problem and it is an exact method. More befinits are listed in the Background 2.8.1 and Methodology 3.6.1. The main usage of the BCR method in this project was to reduce the size of the problem such that it could be solved with a quantum annealer. But BCR also have additional capabilities and use cases.

The BCR method provides good scalability and is easily parallelizable. Each CPU only stores a part of the matrix $\mathbf{H^D}$ in equation 2.38, decouples all local layers until only the first and last layer are connected, and then exchange information with neighbouring CPUs to remove the remaining layers. It also allows for computational interleaving. Since the first and last layer in the BCR method are removed at the end, the open boundary condition (OBC) matrices, Σ_L and Σ_R, can be computed at the same time as the renormalization of $\mathbf{H}$. Something that's not possible with a linear solver which needs the whole matrix $\mathbf{H}$, including the OBCs. The OBCs can often be hard to compute, which makes the BCR preferable over other linear solvers such as SuperLU_dist [77] and

MUMPS [77]. This is a point pointed out in [2].

Other computational methods have also been tested with the BCR method, such as the mixed precision scheme to speed up the matrix operations [2]. Instead of using double precision when computing equations 2.39 - 2.44, a single precision was employed to speed up the calculations. However, it turned out that equation 2.39 and 2.40 must be computed in double precision, otherwise rounding errors are propagated through with the repeated usage of X_i and Y_i. The single precision did not offer the required accuracy. It is possible to use single precision in the matrix multiplications of equations 2.41 - 2.43, with a constraint. Double precision must be used for the last iteration of the Schrödinger-Poisson self-consistent loop to obtain accurate results.

5 Results

In this Results chapter we present the results for the first iteration of the of the Schrödinger-Poisson convergence loop for a Resonant Tunneling Device (RTD). A self-consistent solution was never achieved in the creation of the simulator, and remains a task for future work. Instead the results will focus on the performance of the BCR method and its suggested improvement of an extra layer approach, which has been one of the main contributions of this work.

This chapter also includes the performance of the quantum annealer matrix inversion approaches that was outlined in section 3.6.2.1, as well as the results for one injection energy of the quantum assisted BCR method, where all the matrix inversions are computed on an actual quantum annealer. The D-Wave Systems Inc. quantum annealers are used to conduct the experiments. D-Wave also provides a quantum-classical hybrid solver which we will compare to our quantum assisted BCR method.

5.1 Resonant tunneling device

Using an initial potential (no self-consistent solution was achieved), see figure 5.2, for a RTD when solving the Schrödinger equation, we can compare the BCR method to a solution of the whole linear system of equation. The RTD is 13 nm long and 2 nm high, see figure 5.1. Two SiO_2 layers with a thickness 0.5 nm define a 2 nm $\times$2 nm undoped well region. The 5 nm long cathode and anode are doped with a doping concentration $N_D = 10^{20}$ cm^3. We set the number of nodal points to $n_x = 130$ and $n_z = 20$. So a total of $n_x \times n_z = 2600$ nodal points are used. In all of the computations we only consider a single valley configuration of the effective masses. They are set accordingly:

$$m_x = 0.92m_e, \quad m_z = 0.19m_e, \quad m_y = 0.19m_e \tag{5.1}$$

where m_e is the free electron mass. Additional parameter values are presented in table 5.1.

Table 5.1: Parameter values

n_z	n_x	N_D	# injection energies	# lead nodal points	E_G of Si	E_G of SiO_2	T
20	130	10^{20} cm^3	32	$n_z = 20$	1.12 eV	7.62 eV	300 K

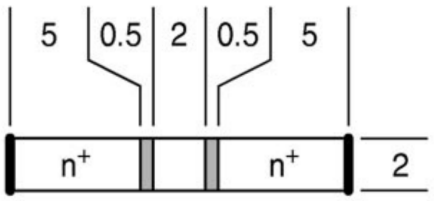


Figure 5.1: [nm] The resonant tunneling device considered. [1]

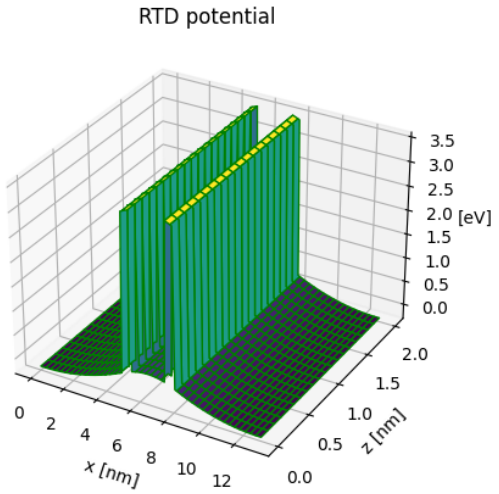


Figure 5.2: An initial potential distribution for a RTD device. Used to compare the wave functions computed with a whole linear system solver, the regular BCR method and the extra layer BCR method.

5.2 Bottleneck analysis

The first aspect studied was the time complexity of the simulator, and where the bottleneck of the simulation scheme resides for one convergence loop. The most time consuming parts of the simulator is the calculation of the energy spectrum and solving the Schrödinger equation for all the injection energies. For the parameter values in table 5.1 and the potential in figure 5.2, the following CPU times was achieved using the processor: Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz 2.11 GHz.

[1] Figure taken from S. E. Laux et. al.'s article *Analysis of quantum ballistic electron transport in ultrasmall silicon devices including space-charge and geometric effects* [13]

Table 5.2: CPU times (Wall times) of different parts of the simulator. The wall time is shorter because the LAPACK drivers used by the Python linear system solver and eigenproblem solvers uses parallel processing. The "Others" contribution is mainly the total time of the calculation of the open boundary conditions, the charge distribution and the solution of the Poisson equation.

$n_z \times n_x$	Energy spectrum [s]	Schrödinger equation [s]	Others [s]
20×130	36.7 (10.4)	179 (53.7)	1.2 (0.7)
20×200	130 (31.3)	666 (173)	3.4 (1.6)
30×130	132 (32.2)	734 (211)	5.0 (2.1)
30×200	508 (99)	2100 (461)	11.3 (2.7)

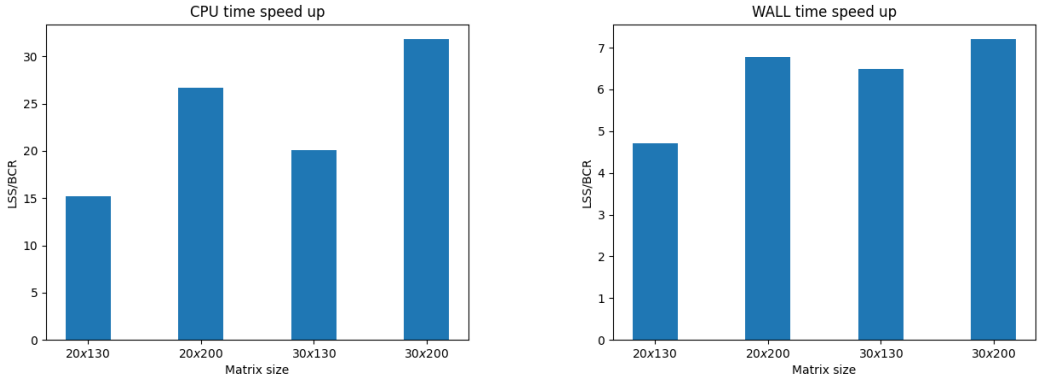
The energy spectrum is calculated by finding the eigenenergies and eigenvectors of two different eigenvalue problems, a time consuming process. Right now all the eigenvalues are found, which are $(n_z - 2)n_x$ many for the "sinus-like" solutions and $(n_z - 2)(n_x - 2)$ many for the "cosinus-like solutions (the minus 2 term stems from the implementation of the Dirichlet boundary conditions), but only the first 32 are used. The Schrödinger equation consist of solving the $(n_z - 2)n_x \times (n_z - 2)n_x$ complex valued linear system of equations 32 times with the Python function *numpy.linalg.solve*.

This process can be sped up with the regular BCR method (the CPU time is about the same for the extra layer BCR). We look at the same configuration of nodal points as in table 5.2. The CPU (Wall) time for the BCR method can be seen in table 5.3.

Table 5.3: CPU times (Wall times) of the BCR method compared to the linear system solver *numpy.linalg.solve*.

$n_z \times n_x$	# injection energies	Schrödinger eq. [s]	Schrödinger eq. with BCR [s]
20×130	32	179 (53.7)	11.8 (11.4)
20×200	32	666 (173)	25 (25.5)
30×130	32	734 (211)	36.6 (32.5)
30×200	32	2100 (461)	66 (64)

We can also plot the speed up of the BCR method, which we have done in figure 5.3.



(a) Speed up CPU

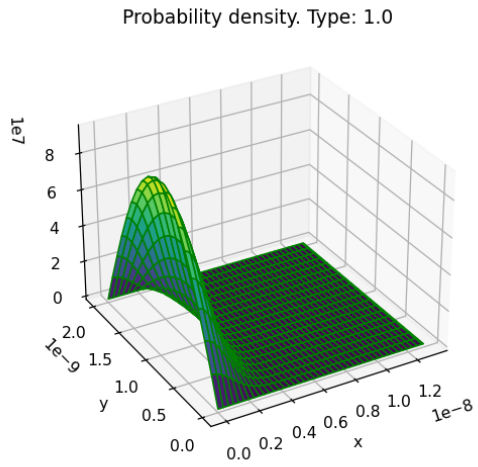
(b) Speed up walltime

Figure 5.3: The speed up of the BCR method compared to the whole linear system solver (called LSS). It is calculated by dividing the LSS time with the BCR time.

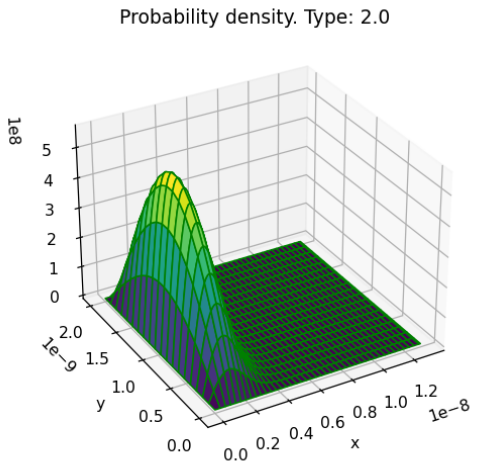
So just by using the BCR method, without any parallelization, we have a substantial speed up. As we see, the BCR method outperforms the full linear equation solver.

5.3 Error analysis: BCR vs Whole system solver

As a measure of the error, we will compute the root mean square (RMS) of the wave functions $\boldsymbol{\phi}_{\beta}^{s,v}$, where we use the whole linear system solution as the "correct" solution. All results are computed in double-precision. We only consider injection from the left lead and the cut-off energy is set to about 1.25 eV, which results in 32 injection energies. We examine the regular BCR method, along with the extra layer approach proposed in section 3.6.1, which should improve accuracy. The 5th layer is the extra layer kept at the end of the decoupling stage. The two standing-wave eigenenergies which discretizes the energy spectrum are looked at separately. We will call the "cosine-like" eigenenergies for Type: 1 energies and the "sine-like" eigenenergies for Type: 2 energies, see section 2.3.2. The wave-functions have not been normalized. To see the difference between the two injection energy types, we plot two of these. We pick the 2nd and 3rd injection energy of both types.

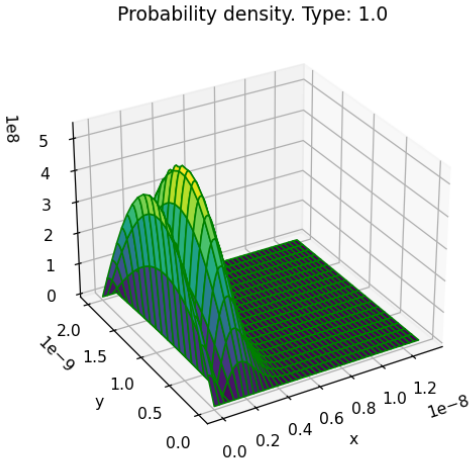


(a) Type: 1 energy. One mode can be seen.

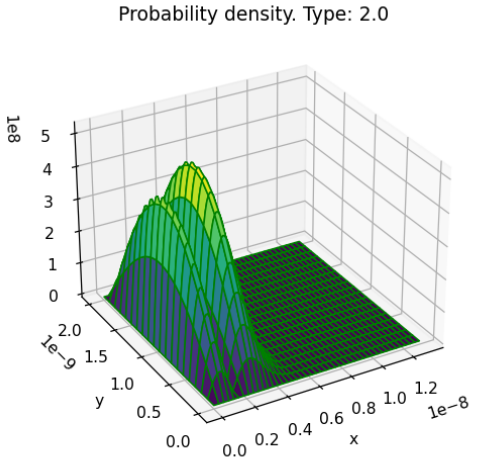


(b) Type: 2 energy. One mode can be seen.

Figure 5.4: Wave function of the 2nd Type: 1 energy and the 2nd Type: 2 energy.



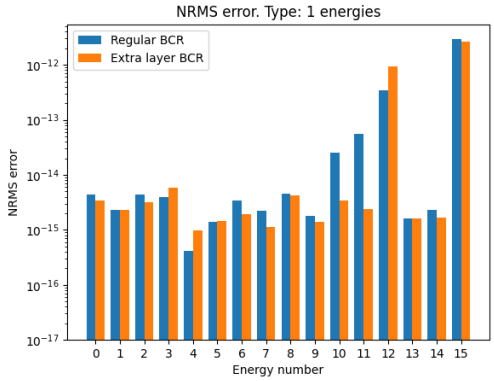
(a) Type: 1 energy. Two modes can be seen.



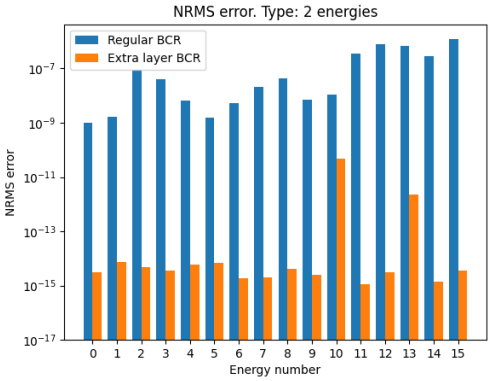
(b) Type: 2 energy. Two modes can be seen.

Figure 5.5: Wave function of the 3rd Type: 1 energy and the 3rd Type: 2 energy.

The Type: 1 energies yields "cosine-like" behaviour at the lead-device boundary, i.e. the wave function's normal derivative is small at the lead-device boundary. Whereas the Type: 2 energies behave "sine-like" at the lead device boundary, i.e. the wave function is small at the lead-device boundary. Next, we will look at the normalized RMS error in figure 5.6 for the the two different injection energy types.



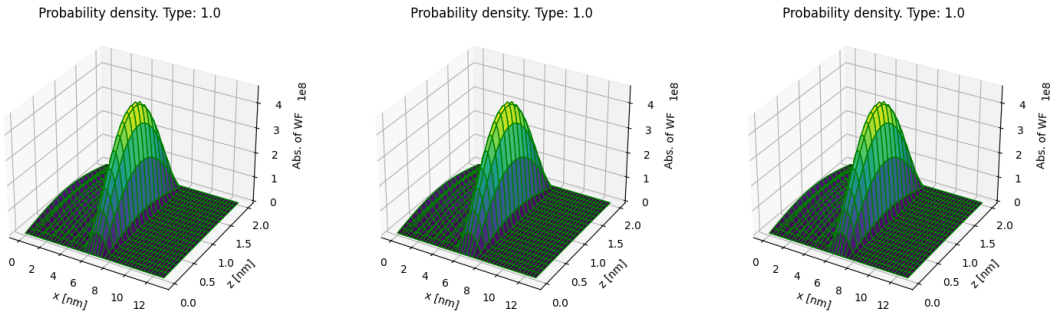
(a) Normalized root mean square error for different Type: 1 energies.



(b) Normalized root mean square error for different Type: 2 energies.

Figure 5.6: Normalized RMS error. The error axis is logarithmic. The normalization is performed by dividing the RMS with the maximum value of the wave function for each energy.

As we can see in figure 5.6a, the error is very similar for almost all Type: 1 energies and the regular BCR and the extra layer BCR perform similarly. However, there are two exceptions, energy number 12 and 15. We plot the 12th energy wave function in figure 5.7 for the whole linear system solution, the regular BCR and the extra layer BCR to see what the solutions looks like.



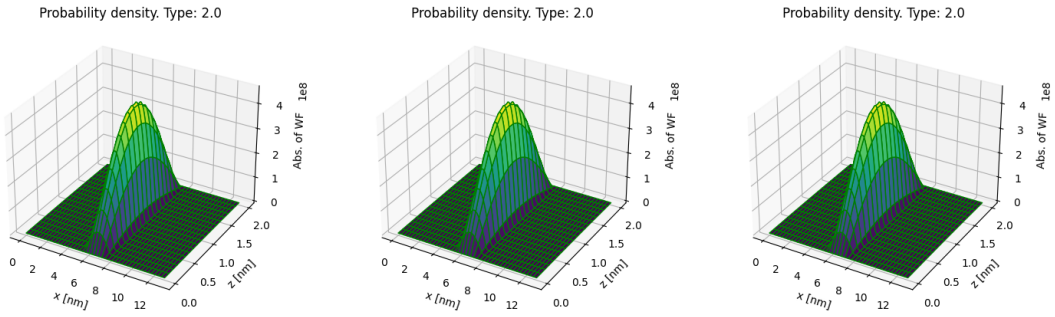
(a) Whole linear system solver. (b) Regular BCR solver. (c) Extra layer BCR solver.

Figure 5.7: Wave function for the 12th Type:1 energy. The value of the injection energy is 0.88911170 eV.

No real difference between the figures can be noticed, but as we saw in figure 5.6a, there is a noticeably larger error for this injection energy. We should remark that the 12th energy leads to tunneling into the well-region of the device, we see a tunneling mode. The same thing happens for the 15th Type:1 energy.

Now, looking at the errors for the Type:2 energies in figure 5.6b, there is more of a difference between the regular BCR and the extra layer BCR, with the extra layer BCR performing much better for most of the energies. The regular BCR is struggling to find

good solutions. For the 10th and the 13th Type:2 energy there is a noticeably larger error compared to the rest for the extra layer BCR. We can look at the wave function for the 10th Type: 2 energy, computed with the three different methods.



(a) Whole linear system solver. (b) Regular BCR solver. (c) Extra layer BCR solver.

Figure 5.8: Wave function for the 10th Type:2 energy. The value of the injection energy is 0.88906483 eV.

Also here, no difference can be seen between the figures because the error is too small. As we can see, the 10th Type:2 energy injection energy leads to a tunneling mode. It appears that both the regular BCR solver and the extra layer BCR struggles with the tunneling modes.

5.4 Quantum effects

One of the specific goals of this project was to simulate the quantum effects of nano-scale transistors. As listed in the section 2.2.1, the relevant quantum effects in nano-scale transistors can be divided into tunneling, confinement and scattering. In this project we considered tunneling and confinement, simulated by solving the open boundaries Schrödinger equation. In figures 5.4, 5.5, 5.7, and 5.8 we have the wave functions (solutions of the Schrödinger equation), and we can clearly see that tunneling and confinement are captured.

In figures 5.7 and 5.8 we have a tunneling mode. The electron has tunneled through the potential barriers, even though the injection energy is smaller than the barrier height. The value of the injection energies are written in the figure caption and the barrier height is about 3.5 eV, see figure 5.2. This behaviour is quantum mechanical since classically the electron should not be able to tunnel through the barrier.

Quantum confinement is also indicated by these figures. The wave function is confined to the device region with zero probability of existing at the device boundary not in contact with the lead. To describe confined electrons you have to solve the Schrödinger equation, which we do here. The confinement leads to discretization of the attainable

energy levels of the electrons and the wave function have mode-like solutions, which we especially can see in figures 5.4 and 5.5. Through our sampling technique of the continuous energy spectrum we find the discrete energy levels that the electrons can attain, thus the energy discretization scheme handles this effect.

5.5 Matrix inversion on the quantum annealer

As proposed in section 3.6.2.1, the objective function for the matrix inverse can be constructed in two different ways, either as a whole matrix objective function or as a LLS with the right hand side being the unit vectors. In these results we will compare these two approaches when computing the matrix inverse of a tri-diagonal matrix A, where A is given by

$$A = \begin{bmatrix} -2 & 1 & 0 & ... & 0 \\ 1 & -2 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 1 & -2 & 1 \\ 0 & ... & 0 & 1 & -2 \end{bmatrix}. \tag{5.2}$$

This matrix has the same structure as the matrices of the 1st decoupling stage of the BCR method. The matrix A is sparse and because of this, the embedding of the Binary Quadratic Model (BQM) on the quantum annealer do not need the same connectivity as if the matrix A was dense, reducing the number of physical qubits needed. This property is used when creating the embedding for the whole matrix solver, which needs a lot more qubits than the unit vector solver.

In the unit vector approach, we sample a solution for the BQM created from the LLS objective function for different right-hand side unit vectors. Since the problem basically is the same for all these different objective functions (the same matrix A), we can use the same embedding. The only thing that differs is the weights, which are adjusted when we call the sampler with a specific unit vector BQM. So, we can use the same embedding for all the unit vectors. This is preferred since the embedding is a time consuming task. However, since we have to solve for all the unit vectors in the unit vector approach, we will have several sampling times. The total sampling time from all samplings is the value presented in the tables 5.4, 5.5 and 5.6. The embedding is performed with *minorminer.find_embedding()*, which is based on a heuristic algorithm described in [46].

In tables 5.4, 5.5 and 5.6 we have tabulated the embedding time, sampling time, logical qubits, physical qubits and the RMS error of both methods. The Dwave quantum annealer chip ADVANTAGE_SYSTEM4.1 is used as the sampler. 100 annealing circles

is performed with a qubit accuracy of 3 qubits. We have also tabulated the maximal attainable error given the search domains we chose.

Table 5.4: Embedding and sampling times for matrix inversion on the quantum annealer, using 3 qubits accuracy and 100 annealing circles. Qubit usage is also included. The RMS error is computed for the lowest energy sample. WHS = Whole Matrix Solver. UVS = Unit Vector Solver.

Matrix size:	5×5	6×6	7×7	8×8	9×9	10×10
Embedding time WMS [s]	3.787108	4.375526	4.093422	8.597509	7.391740	8.927392
Embedding time UVS [s]	3.515022	3.569846	3.640435	4.929300	6.760475	9.809062
Sampling time WMS [ms]	32.917	29.067	46.346	62.530	76.841	74.717
Tot. sampling time UVS [ms]	97.065	100.659	207.332	247.830	268.645	296.733
Logical qubits WMS	75	108	147	192	243	300
Logical qubits UVS	15	18	21	24	27	30
Physical qubits WMS	156	227	353	456	569	740
Physical qubits UVS	34	47	60	82	98	124
RMS error WMS	0.155065	0.183223	0.192029	0.239608	0.282303	0.349393
RMS error UVS	0.163246	0.197081	0.160062	0.240180	0.274895	0.364064
Max RMS error	0.391932	0.440959	0.490990	0.541667	0.592781	0.644205

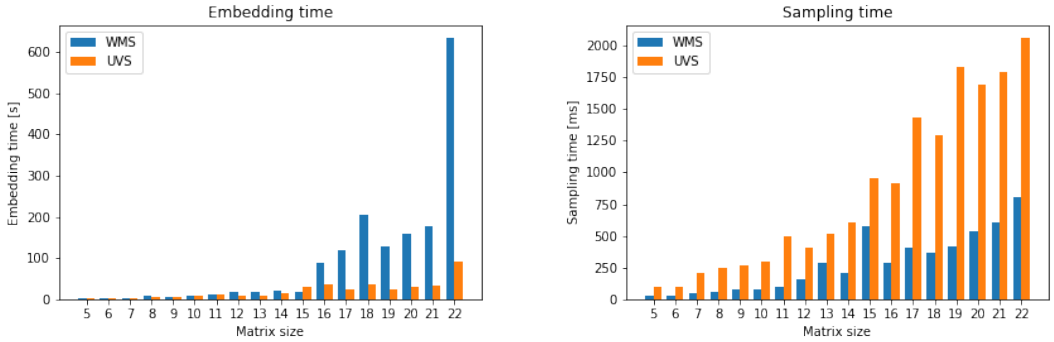
Table 5.5: Continuation. WHS = Whole Matrix Solver. UVS = Unit Vector Solver.

Matrix size:	11×11	12×12	13×13	14×14	15×15	16×16
Embedding time WMS [s]	12.559210	16.898327	17.102410	20.886657	18.220348	88.843074
Embedding time UVS [s]	12.267178	8.180394	9.763127	14.062302	29.533599	35.546644
Sampling time WMS [ms]	96.006	161.985	286.886	212.740	579.663	285.967
Tot. sampling time UVS [ms]	496.056	406.117	516.697	610.163	955.900	914.677
Logical qubits WMS	363	426	498	576	660	750
Logical qubits UVS	33	36	39	42	45	48
Physical qubits WMS	911	1076	1366	1587	1824	2198
Physical qubits UVS	147	191	211	248	254	326
RMS error WMS	0.300349	0.366480	0.421162	0.450764	0.440341	0.487875
RMS error UVS	0.373167	0.315537	0.424001	0.400299	0.448990	0.590392
Max RMS error	0.695857	0.747682	0.799639	0.851702	0.903850	0.956066

Table 5.6: Continuation. WHS = Whole Matrix Solver. UVS = Unit Vector Solver

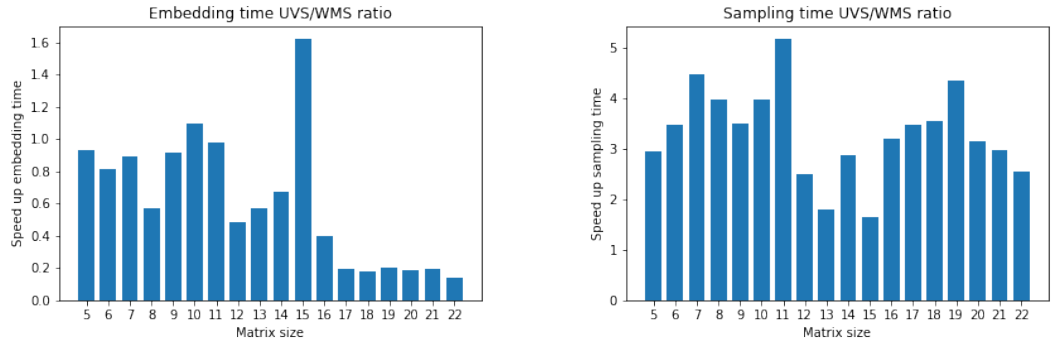
Matrix size:	17×17	18×18	19×19	20×20	21×21	22×22
Embedding time WMS [s]	120.761445	206.347493	127.960260	158.455197	175.970814	633.731196
Embedding time UVS [s]	23.671692	36.688154	25.563607	30.106782	34.316238	90.679036
Sampling time WMS [ms]	409.993	363.156	420.914	537.626	602.084	806.876
Tot. sampling time UVS [ms]	1426.280	1287.725	1826.708	1689.135	1785.174	2057.731
Logical qubits WMS	846	948	1056	1170	1290	1386
Logical qubits UVS	51	54	57	60	63	66
Physical qubits WMS	2502	2807	3016	3312	3681	4029
Physical qubits UVS	354	398	409	566	494	568
RMS error WMS	0.597358	0.525439	0.589361	0.666625	0.750231	0.724590
RMS error UVS	0.581689	0.590786	0.504415	0.663391	0.564284	0.633990
Max RMS error	1.008339	1.060660	1.113021	1.165416	1.217840	1.270290

We also plot the different results in bar graphs, which can be seen below.



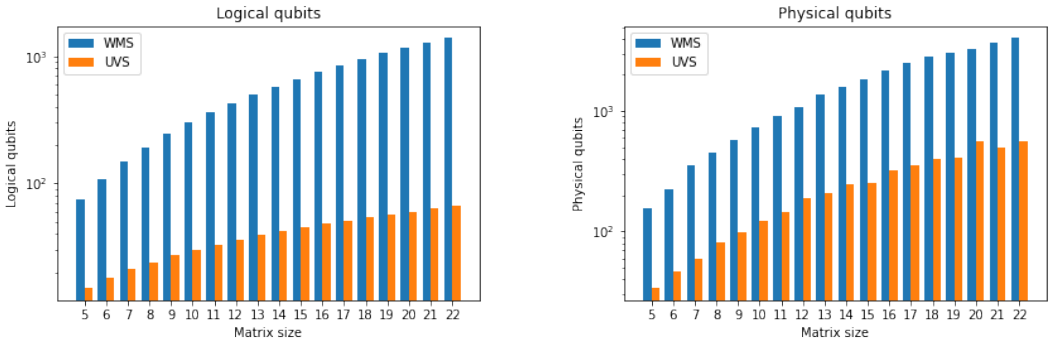
(a) Embedding time (b) Sampling time

Figure 5.9: Sampling and embedding time. The sampling is performed with the Dwave annealer ADVANTAGE_SYSTEM4.1.



(a) Embedding time ratio (b) Sampling time ratio

Figure 5.10: Ratio sampling and embedding time. The sampling is performed with the Dwave annealer ADVANTAGE_SYSTEM4.1.



(a) Logical qubits needed to represent the BQM with the given specifications.

(b) Actual physical qubits used on the quantum annealer.

Figure 5.11: Logical and physical qubits for the two inversion methods. The Dwave annealer ADVANTAGE_SYSTEM4.1 is used, which has 5627 available physical qubits.

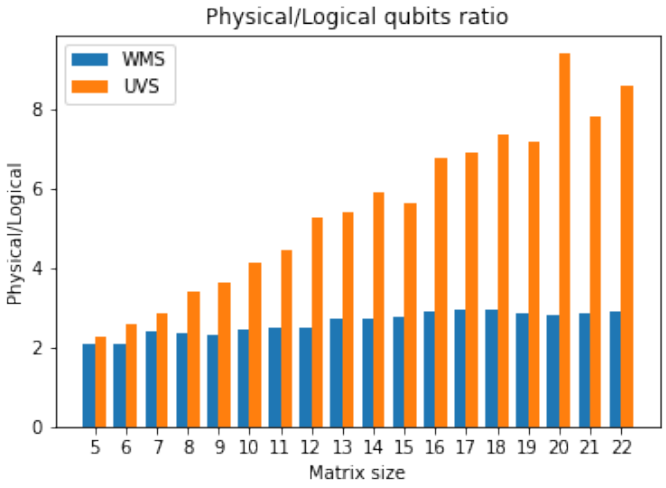


Figure 5.12: The ratio of physical and logical qubits for the two inversion methods.

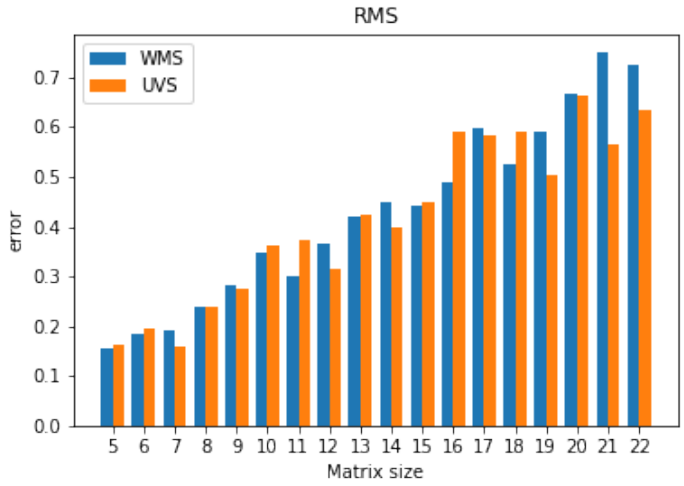


Figure 5.13: The RMS error of the lowest energy sample for the two methods using 100 samplings.

Starting with figure 5.9a, we see that the embedding time for both approaches increases as the matrix size increase, which is expected since the increased matrix size leads to more qubits that needs to be mapped onto the QPU Topology. Because the whole matrix solver needs a lot more logical qubits, see figure 5.11a, it will also be harder to embed.

Next, we can look at the sampling time in figure 5.9b. Also here, the time increases as the matrix size increases, which is expected. The sampling time is larger for the unit vector approach. An average of the individual sampling time for each unit vector can be obtained by dividing the total sampling time with the matrix size. Some fluctuations can be seen, caused by the stochastic nature of the quantum annealer.

The ratio of the sampling and embedding time is plotted in figure 5.10. We begin to look at the embedding time in figure 5.10a. Up until the 15×15 matrix the embedding time is similar for both approaches, with some oscillations. However, for larger matrix sizes the whole matrix solver is noticeably slower than the unit vector solver. Considering the sampling time next figure 5.10b, the whole matrix solver is on average about 3 times faster than the unit vector solver.

Now, turning our attention to the qubit usage in figure 5.11. The logical qubits scale as expected, with the whole matrix solver scaling quadratic and the unit vector solver scaling linear in the amount of logical qubits. The scaling seems to be similar for the physical qubits in figure 5.11b. Looking at the ratio of physical and logical qubits in figure 5.12, we get a better grasp of the physical qubit usage. The ratio for the whole matrix solver is close to constant for the different matrix sizes, whilst the unit vector solver shows a linear increase in the physical qubit usage. It is expected that the whole matrix solver has a better ratio usage of physical qubits since we have removed zero

interactions and thus reduced the necessary connectivity.

Lastly, we will examine the RMS error in figure 5.13. We have a similar error for both inversion methods for all matrix sizes. This is a bit surprising since the unit vector solver should perform better with the same amount of samplings since it is probing a smaller state space. It could be that the number of samplings is to small for any of the methods to perform good. The errors can be compared to the "Max RMS error" in tables 5.4, 5.5 and 5.4, and we see that the errors are fairly large. So, a higher number of annealing cycles should be employed in further experiments.

5.6 Quantum assisted BCR

In this final section we will combine the BCR method with the matrix inversion on the quantum annealer. We will use the unit vector approach, since it is more capable than the whole matrix solver, given the fact that we are limited by the connectivity and the amount of qubits. We will only consider the solution of one (the first) injection energy. Because of the limited free time (5 min) provided by D-Wave using their quantum annealers, the number of nodal points will be greatly reduced when computing these results. We use 100 annealing cycles and 1 qubit accuracy. The Max RMS error in table 5.7 is the error of the wave function obtained when the maximal attainable error for the matrix inverse, given the domain, is used.

Table 5.7: The sampling time and embedding time of the quantum annealer used in the BCR method. The RMS error is the error of the full wave function when compared to when the inverses are computed with the *np.linalg.inv* solver, called the classical inverse. The matrix size of the inverses computed are $(n_z - 2) \times (n_z - 2)$ because of the Dirichlet boundary conditions.

$n_z \times n_x$:	10×20	12×25	14×30
Embedding time [s]	4.23	3.67	4.8
Avg. sampling time per layer [ms]	153	181	215
Tot. sampling time [s]	2.6	4.0	5.8
Tot. classical inverse time [ms]	7.2	9.4	12.17
Logical qubits	8	10	12
Physical qubits	12	18	24
RMS error	16.42	71.49e+3	84.08e+6
Max RMS error	16.30e+7	17.43e+7	18.47e+7

We can also look at the total time consumption of the whole procedure of solving the Schrödinger equation, presented in table 5.8.

Table 5.8: Time consumption of the whole procedure of solving the Schrödinger equation. CPU (Wall) time. The wall time is very long because we have to access the quantum annealer from the Leap cloud service. The CPU time is also longer than the combined total sampling time and embedding time in table 5.7, that is because the objective function has to be compiled and transformed into a BQM, which can take quite some time.

$n_z \times n_x$	Quantum assisted BCR	Regular BCR	Whole linear system solver
10×20	8.73 s (1min 52s)	15.6 ms (11 ms)	359 ms (81.2 ms)
12×25	20.4 s (3 min 3s)	15.6 ms (9 ms)	484 ms (87 ms)
14×30	40.4 s (4min 46s)	78.1 ms (24.2 ms)	391 ms (107 ms)

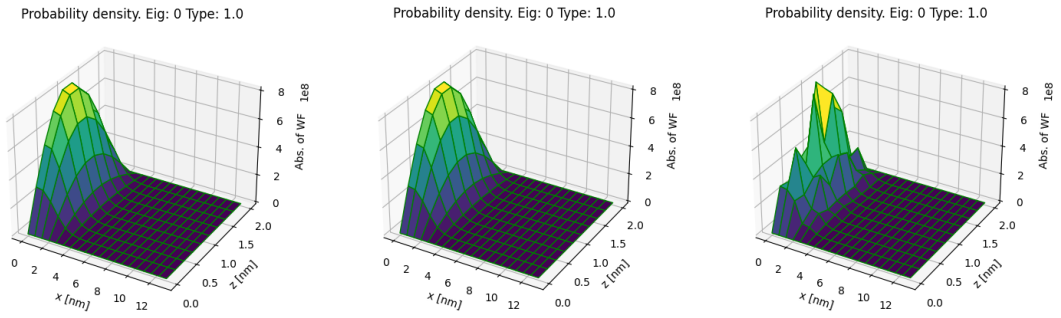
5.6.1 Comparison to D-Wave's hybrid solver

D-Wave provides hybrid solvers that accepts an arbitrary quadratic model and uses state-of-the-art classical algorithms together with smart allocation of quantum processing resources to find its solution [14]. Since our quantum assisted BCR method is a form of a hybrid solver, it can be interesting to compare it to D-Wave's hybrid solver. The idea is to create an BQM of the whole discretized Schrödinger equation (equation 2.4), feed it to the D-Waves hybrid solver and compare it to the results from our BCR method. Since the wave functions are complex, we have to split the problem in a real and imaginary part

$$\begin{aligned} &(\mathbf{H}_{re} + i\mathbf{H}_{im})(\boldsymbol{\phi} + i\boldsymbol{\phi}_{im}) = (\mathbf{B}_{re} + i\mathbf{B}_{im}) \\ \Rightarrow &\begin{bmatrix} \mathbf{H}_{re} & -\mathbf{H}_{im} \\ \mathbf{H}_{im} & \mathbf{H}_{re} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}_{re} \\ \boldsymbol{\phi}_{im} \end{bmatrix} = \begin{bmatrix} \mathbf{B}_{re} \\ \mathbf{B}_{im} \end{bmatrix} \end{aligned} \tag{5.3}$$

doubling the size of the original system of linear equations. This linear system of equations is used to create a LLS objective function, see section 2.7.3.

Just as in the previous section, we create a BQM with 1 qubits accuracy, and the domains are selected from the correct solution. The nodal points in the z and x directions are set to $n_z = 10$ and $n_x = 20$. For larger problem sizes, we could not compile the BQM for the hybrid solver. The hybrid solver used was the Leap provided *LeapHybridSampler*, which samples a BQM. The default settings of the sampler was used. In figure 5.14 the resulting wave functions are plotted for the first injection energy of Type: 1.



(a) Whole linear system solver.

(b) Quantum assisted BCR solver.

(c) Leap hybrid solver

Figure 5.14: Comparison between the Leap hybrid solver and the quantum assisted BCR method.

As we can see in the figure 5.14, the Leap hybrid solver cannot find a close resemble to the whole linear system solver, which we use as the reference for the correct solution, whereas the quantum assisted BCR solver have a close resemblance. We also look at the time it takes to solve the Schrödinger equation with the Leap hybrid solver and our quantum assisted BCR in table 5.8.

Table 5.9: Time consumption of the whole procedure of solving the Schrödinger equation for the Leap hybrid solver compared to the quantum assisted BCR solver. CPU (Wall) time. Sampling time for the quantum assisted BCR is the same as the total sampling time in table 5.8.

Solver:	Quantum assisted BCR	Leap hybrid solver
Whole procedure time	8.73 s (1min 52s)	2min 51s (3min 7s)
Sampling time	2.6 s	594 ms

The Leap hybrid solver has to compile, create a BQM and decode a bigger problem than the quantum assisted BCR method, causing a long procedure time. The sampling time is the time it takes for the *LeapHybridSampler* to find a sampleset from the BQM.

6 Discussion

6.1 Discussion of results

We begin the discussion with an analysis of the results presented in chapter 5 Results.

6.1.1 Bottleneck analysis

In the bottleneck analysis it became clear that solving the Schrödinger equation is the most time consuming process, making up approximately 80% of the total simulation time of one convergence loop. See table 5.2. Therefore, the solution of the Schrödinger equation are most interesting to incorporate in a quantum algorithm. As already mentioned, current quantum computers are limited by the size of the problems they can solve, so we first applied the BCR method to reduce the problem size.

The BCR method can also speed up the solution time. Either by allowing parallel processing, or simply by reducing the complexity of the problem. In table 5.3 and figure 5.3 we see that the BCR method reduces the complexity of the problem, leading to at least a 15 times faster speed up (CPU time). The speed up for the wall time is approximately 5 times faster because of the multi-threading employed by the *numpy.linalg.solve* function. The BCR method also allows for parallel processing, not employed in this project, that can speed up the solution time even more.

6.1.2 Error analysis BCR

First, we should discuss our method to compute the error of the BCR method. We assumed that the solution that is obtained by solving the full discretized Schrödinger equation (equation 2.4) with *numpy.linalg.solve* is a good approximate for the "correct" solution. The "correct" solution was then compared to the solutions calculated with the BCR methods and an error was obtained. However, it might be that the "correct" solution calculated with *numpy.linalg.solve* includes errors compared to the true correct solution. So, a better way to compute the error would be to multiply our solution **Œ** with **H** and compare the product to the injection vector **B**, and in this way check our solution.

The errors for the BCR method are very small, as seen in figure 5.6. That is because the BCR method is an exact method. The errors that are present arise because of the

computers precision computing matrix inverses/multiplications. Since many operations are computed, it is important that each operation has a high precision, or an error will accumulate. In the results, all operations are computed in double-precision floating-point format. As mentioned in the Literature review section 4.3.1, a single-precision is not accurate enough when using the BCR method.

If we now turn our attention to the errors of the "extra layer" BCR, looking at figures C.1 and 5.6, we see that the the error is reduced using the same double-precision compared to the regular BCR. So, it seems that the accuracy of the BCR method is dependent on the layers with which the construction stage starts from. This was the hypothesis which the idea of the "extra layer" BCR came from. The reason for the improvement using the "extra layer" BCR is not yet determined, although a speculation is that the decrease in the dynamic range of the matrix operations is the cause for the improvement of the "extra layer" BCR. But a more thorough examination must be conducted to establish a reason. For example, an attempt to improve the accuracy of the tunneling modes (energy 12 and 15 in figure 5.6a and energy 10 and 13 in figure 5.6b) could pose another test for the hypothesis.

Lastly, the significance of the "extra layer" BCR is not established. In the end, the objective of device simulations is to extract I-V curves (or some other characteristics) when the device is driven out of equilibrium. Since we do not have a fully working simulator, it is not clear how the smaller boundaries contribute to the current, charge density, etc, and how the "extra layer" BCR affects the simulations.

6.1.3 Matrix inverses

In section 5.5 the matrix inversion on the quantum annealer was examined as a test before implementing it in the BCR method. The two different approaches, called the WMS and UVS were compared. As expected, the WMS uses more qubits but the embedding time is similar until the 16×16 matrix size, where the WMS shows an increase in embedding time compared to the UVS. See figures 5.9a and 5.10a. At matrix size 22, we have a sharp rise in the embedding time, which could be caused by the fact that we are close to the max capacity of the annealer, making it harder to find an embedding. Even if the ADVANTAGE_SYSTEM4.1 chip has 5627 physical qubits, you cannot necessarily use all of them. Inactive qubits and approaching the boundaries on the chip makes it harder to find an embedding. Some fluctuations can be seen in the embedding times, which is caused by the stochastic nature of the embedding. It can therefor be interesting to do the embedding several times to get a statistical average with deviations.

Looking at the sampling time next. First of, it should be mentioned that the sampling time is equal to the *annealing time* $\times$ *number of annealing cycles*. Both the *annealing time* and *number of annealing cycles* are parameters that can be tuned to find the best annealing solu-

tion. In the results the default annealing time and 100 annealing cycles was used, which necessarily are not the best settings. The default annealing time changes as the size of the problem changes. Looking at the sampling times in figure 5.9b we see that the WMS has a shorter sampling time than the UVS for all matrix sizes, which is mainly because we have to perform a sampling for every unit vector. As we saw in section 3.6.2.1, the UVS is a decomposed version of the linearized WMS, or opposite, the WMS is the parallel version of the UVS. Which makes it reasonable that the sampling time of the WMS is shorter.

However, the comparison between the UVS and the WMS is not quantitative. If we look at the error in figure 5.13, we see that it is similar for both approaches, even though the UVS should have a smaller error because it is probing a smaller state space. Meaning that it should be able to find a better solution than the WMS using the same amount of annealing cycles. Thus, we are probably using to few annealing cycles. We can also look at the qubit ratio in 5.12. There we see that the WMS have a better usage of the physical qubits, which are not strange considering that we have removed the zero interactions for the WMS, see section 5.5, reducing the connectivity. In the UVS we created an embedding with full connectivity, not removing zero interactions. A proper parameter study is needed for a quantitative comparison between the two methods, where the annealing time and annealing cycles are optimised for each method, and a corresponding connectivity is used for both approaches in the embedding.

Lastly, we should comment on the actual use cases of these methods. One of the problems with these annealing problems is that a search domain for the solution must be set. We presented a brief discussion on this problem in section 3.6.2.2. In this project we set our domains knowing the correct solution, which means that we can use few qubits to obtain accurate solutions. Usually, this is not the case and a different approach for finding a good domain is necessary (outlined in 3.6.2.2) if the matrix inverse on the quantum annealer is to replace a classical solver.

6.1.4 Quantum assisted BCR

Finally, the quantum assisted BCR was evaluated in the results, combining the BCR method with the matrix inversion on the quantum annealer. In table 5.7 three different problem sizes are presented. First, we can look at the RMS error, which increases drastically for increased problem size. This is an effect of the many operations in the BCR method and the need for accurate solutions.

Next, we can consider the total sampling time and compare it to the total classical inverse time, which is the time it takes for *numpy.linalg.solve* to compute the same matrix inverses. As we can see, the classical matrix inverse is about 400 times faster than the quantum correspondence. The time consumption comparison continues in table 5.8, where the speed up of the classical algorithm is even greater (500 times faster) because of time

spent creating the annealing problem (compiling the objective function and transforming it into a BQM). This means that the quantum algorithm is a lot slower than the classical algorithm. Even if it is possible to optimize the use of the quantum annealer, considering things such as annealing time, annealing cycles, embedding, objective function, and compiling etc., there currently is a massive gap to the classical algorithm.

In the last section the Leap hybrid solver provided by D-Wave is compared to the quantum assisted BCR. In figure 5.14 the wave functions computed with the Leap hybrid solver, *numpy.linalg.solve* and the quantum assisted BCR are compared. The Leap hybrid solver provides a bad solution compared to the other two methods. The reason for the failure of the Leap hybrid solver is probably caused by the size of the BQM, which is a lot bigger than the BQM of the Quantum assisted BCR. The Leap hybrid solver is also employing a decomposition method, which might not be as accurate as the BCR method.

In table 5.9 the time consumption is compared. As we see, the whole procedure time is longer for the Leap hybrid solver than our quantum assisted BCR. So, in the end the quantum assisted BCR performs better than the Leap hybrid solver, but the matrix inversion on the quantum annealer has a long way to go before being comparable to the classical algorithms. In the Literature Review in section 4.2.1 we outlined the HHL algorithm for solving linear system of equations on a gate-based quantum computer. It has a proven speed up over classical algorithms, and could be a way forward for the quantum assisted BCR once there exist adequate gate-based quantum computers.

6.2 Future works

One obvious future extension of this work is to create a fully functional self-consistent iteration process. The self-consistent iteration process of solving the Schrödinger and Poisson equation seldom converges in practice [29]. So to achieve a solution, you need a good initial guess for the potential. That can either be obtained by the solution of a simpler physical model or a previous self-consistent solution. To speed up the convergence, one can also implement a non-linear solver, such as the Newton method. See Laux et al. [13] for a starting-point.

It can also be interesting to study different types of transistor technologies. In this project we only looked at the RTD, whereas the silicon ultrathin-body double-gate FETs (possibility to use a subband decomposition, see section 4.1.4) and the newer FinFETs and gate-all-around transistors are more interesting to study from a commercial stand point, although this would need an extension of the model to 3D. The inclusion of time-dependence to study switching characteristics of nanoscale transistors, is also of

interest.

Another possibility for future work is to include scattering into the physical model. The PME is a suitable choice of method to model the dissipative scattering in short devices, see sections 4.1.2.3 and 4.1.3. Scattering is an inevitable quantum effect in nanoscale devices and it affects the current flow through the device. It is also possible to extend the single band effective mass model to a more realistic model, such as the full-band atomistic model in [78]. The full-band and first-principles models are more accurate, and therefore also more numerically expensive. But efficient use of discretization techniques, DDM, and HPC/QC allows for a treatment of larger problems. The subband and WKB techniques presented in section 4.1.4 are examples that offer interesting extensions to our model for certain transistor technologies to speed up the simulation time.

Additional use of QC resources is another future work. The QC can assist other parts of the simulator, such as the eigensolver (see secion 4.2.2) for computing the energy spectrum. A quantum algorithm could also be used to solve the linear system of equations that is part of the BCR method, using either a quantum annealing algorithm such as the one presented in section 2.7.3, or the HHL algorithm presented in section 4.2.1 given that the gate-based quantum computers continue to improve. Finally, one could optimize the use of the quantum annealer for the matrix inversion, tuning annealing cycles, annealing time, embeddings, etc. in a future work.

7 Conclusion

In this thesis project, the current capabilities of the quantum annealer were tested in the context of a device simulator, with a focus on quantum effects. The results were computed for a resonant tunneling device. There exist several transistor models with the capability of simulating quantum effects, so a comprehensive literature review was conducted to find a suitable choice of model. The model chosen is based on a self-consistent iterative process of solving the Schrödinger- and Poisson equation. In this quantum model, the solution of the open-boundary condition Schrödinger equation is the most time-consuming part of the simulator, although necessary to capture static quantum effects such as quantum tunneling and quantum confinement. It is not possible to directly assist the solution of the Schrödinger equation with a quantum annealer, because of the limited number of qubits. However, with the use of a domain decomposition method it is possible to reduce the size of the problem such that it is solvable on present quantum annealers. The block cyclic reduction (BCR) method, in addition to reducing the problem size, speeds up the solution of the Schrödinger equation with a factor 15. It was discovered that for certain injection energies, the BCR method had a decrease in accuracy. This inspired a modification called the extra layer BCR. The proposed extra layer BCR improves the accuracy of the BCR method for wave functions with small first and last layers. The significance of this improvement is unknown because of the lack of a fully functioning simulation scheme. In the project we also saw that the quadratic unconstrained binary optimization (QUBO) whole matrix solver (WMS) is faster than the unit vector solver(UVS), but as expected uses more qubits. A more thorough comparison should be conducted for a quantitative advantage of the WMS. Compared to the state of the art classical matrix inverses, the quantum annealer algorithm is a lot slower, suggesting that the usage of a quantum computer is not optimal for solving the linear algebra problems resulting from transistor modeling. Lastly, it was found that the quantum assisted BCR performs better than the D-Wave Leap hybrid solver, both in its accuracy and the time-to-solution.

Literature

[1] C. S. Lent and D. J. Kirkner, "The quantum transmitting boundary method," *Journal of Applied Physics*, vol. 67, no. 10, pp. 6353–6359, 1990. [Online]. Available: https://doi.org/10.1063/1.345156.

[2] M. Luisier, T. B. Boykin, G. Klimeck, and W. Fichtner, "Atomistic nanoelectronic device engineering with sustained performances up to 1.44 pflop/s," in *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, pp. 1–11.

[3] *LAPACK—Linear Algebra PACKage*. [Online]. Available: https://netlib.org/lapack/ (accessed October 14, 2022).

[4] *The Advantage™ Quantum Computer | D-Wave*, en-US. [Online]. Available: https://www.dwavesys.com/solutions-and-products/systems/ (accessed October 3, 2022).

[5] M. L. Rogers and R. L. Singleton, "Floating-point calculations on a quantum annealer: Division and matrix inversion," *Frontiers in Physics*, vol. 8, 2020. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fphy.2020.00265.

[6] G. E. Moore, "Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff.," *IEEE Solid-State Circuits Society Newsletter*, vol. 11, no. 3, pp. 33–35, 2006.

[7] *Samsung Begins Chip Production Using 3nm Process Technology With GAA Architecture*, en. [Online]. Available: https://news.samsung.com/global/samsung-begins-chip-production-using-3nm-process-technology-with-gaa-architecture (accessed August 16, 2022).

[8] W. Fichtner, D. Rose, and R. Bank, "Semiconductor device simulation," *IEEE Transactions on Electron Devices*, vol. 30, no. 9, pp. 1018–1030, 1983.

[9] F.-Q. Xie, L. Nittler, C. Obermair, and T. Schimmel, "Gate-controlled atomic quantum switch," *Phys. Rev. Lett.*, vol. 93, p. 128 303, 12 Sep. 2004. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.93.128303.

[10] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Phys. Rev. Lett.*, vol. 103, p. 150 502, 15 October 2009. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.103.150502.

[11] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, *Quantum computation by adiabatic evolution*, 2000. [Online]. Available: https://arxiv.org/abs/quant-ph/0001106.

[12] *IBM Unveils Breakthrough 127-Qubit Quantum Processor*, en-us. [Online]. Available: https://newsroom.ibm.com/2021-11-16-IBM-Unveils-Breakthrough-127-Qubit-Quantum-Processor (accessed October 11, 2022).

[13] S. E. Laux, A. Kumar, and M. V. Fischetti, "Analysis of quantum ballistic electron transport in ultrasmall silicon devices including space-charge and geometric effects," *Journal of Applied Physics*, vol. 95, no. 10, pp. 5545–5582, 2004. [Online]. Available: https://doi.org/10.1063/1.1695597.

[14] *Hybrid Solvers — Ocean Documentation 5.4.0 documentation*. [Online]. Available: https://docs.ocean.dwavesys.com/en/stable/overview/hybrid.html (accessed October 5, 2022).

[15] M. Riordan, *Transistor - Transistors and Moore's law | Britannica*, en. [Online]. Available: https://www.britannica.com/technology/transistor (accessed Sep. 1, 2022).

[16] P. Hofmann, *Solid state physics: an introduction*, English. Weinheim: Wiley-VCH, 2011.

[17] D. K. Ferry, J. Weinbub, M. Nedjalkov, and S. Selberherr, "A review of quantum transport in field-effect transistors," *Semiconductor Science and Technology*, vol. 37, no. 4, p. 043 001, February 2022. [Online]. Available: https://doi.org/10.1088/1361-6641/ac4405.

[18] M. G. Ancona, "Density-gradient theory: A macroscopic approach to quantum confinement and tunneling in semiconductor devices," *Journal of Computational Electronics*, vol. 10, no. 1, pp. 65–97, 2011.

[19] R. L. Liboff, *Kinetic Theory: Classical, Quantum, and Relativistic Descriptions*, en. Springer Science & Business Media, Sep. 2003, Google-Books-ID: 9_LfoChz3GcC.

[20] N. Sano, A. Hiroki, and K. Matsuzawa, "Device modeling and simulations toward sub-10 nm semiconductor devices," *IEEE Transactions on Nanotechnology*, vol. 1, no. 1, pp. 63–71, 2002.

[21] M. Lundstrom and Z. Ren, "Essential physics of carrier transport in nanoscale mosfets," *IEEE Transactions on Electron Devices*, vol. 49, no. 1, pp. 133–141, 2002.

[22] M. Razavy, *Quantum theory of tunneling*, English. River Edge, NJ: World Scientific, 2003.

[23] F. Trixler, "Quantum tunnelling to the origin and evolution of life," *Current organic chemistry*, vol. 17, pp. 1758–1770, August 2013.

[24] M. Lenzlinger and E. H. Snow, "Fowler-nordheim tunneling into thermally grown sio2," *Journal of Applied Physics*, vol. 40, no. 1, pp. 278–283, 1969. [Online]. Available: https://doi.org/10.1063/1.1657043.

[25] A. B. Fowler, G. L. Timp, J. J. Wainer, and R. A. Webb, "Observation of resonant tunneling in silicon inversion layers," *Phys. Rev. Lett.*, vol. 57, pp. 138–141, 1 Jul. 1986. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.57.138.

[26] D. K. Ferry, "The onset of quantization in ultra-submicron semiconductor devices," *Superlattices and Microstructures*, vol. 27, no. 2, pp. 61–66, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0749603699908005.

[27] T. Edvinsson, "Optical quantum confinement and photocatalytic properties in two-, one- and zero-dimensional nanostructures," *Royal Society Open Science*, vol. 5, no. 9, p. 180 387, 2018. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rsos.180387.

[28] S. M. Goodnick, D. K. Ferry, C. W. Wilmsen, Z. Liliental, D. Fathy, and O. L. Krivanek, "Surface roughness at the si(100)-SiO_2 interface," *Phys. Rev. B*, vol. 32, pp. 8171–8186, 12 December 1985. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.32.8171.

[29] P. B. Vyas, M. L. Van de Put, and M. V. Fischetti, "Master-equation study of quantum transport in realistic semiconductor devices including electron-phonon and surface-roughness scattering," *Phys. Rev. Applied*, vol. 13, p. 014 067, 1 January 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevApplied.13.014067.

[30] M. V. Fischetti, "Theory of electron transport in small semiconductor devices using the pauli master equation," *Journal of Applied Physics*, vol. 83, no. 1, pp. 270–291, 1998. [Online]. Available: https://doi.org/10.1063/1.367149.

[31] ——, "Master-equation approach to the study of electronic transport in small semiconductor devices," *Phys. Rev. B*, vol. 59, pp. 4901–4917, 7 February 1999. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.59.4901.

[32] W. Pötz, "Self-consistent model of transport in quantum well tunneling structures," *Journal of Applied Physics*, vol. 66, no. 6, pp. 2458–2466, 1989. [Online]. Available: https://doi.org/10.1063/1.344257.

[33] Y. He, T. Kubis, M. Povolotskyi, J. Fonseca, and G. Klimeck, "Quantum transport in nemo5: Algorithm improvements and high performance implementation," in

2014 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD), 2014, pp. 361–364.

[34] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse ising model," *Phys. Rev. E*, vol. 58, pp. 5355–5363, 5 November 1998. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.58.5355.

[35] K. Jun, *Qubo formulations for numerical quantum computing*, 2021. [Online]. Available: https://arxiv.org/abs/2106.10819.

[36] T. B. Boykin, M. Luisier, and G. Klimeck, "Multiband transmission calculations for nanowires using an optimized renormalization method," *Phys. Rev. B*, vol. 77, p. 165 318, 16 April 2008. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.77.165318.

[37] M. Luisier, G. Klimeck, A. Schenk, W. Fichtner, and T. B. Boykin, "A parallel sparse linear solver for nearest-neighbor tight-binding problems," in *Euro-Par 2008 – Parallel Processing*, E. Luque, T. Margalef, and D. Benítez, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 790–800.

[38] M. Luisier and G. Klimeck, "Numerical strategies towards peta-scale simulations of nanoelectronics devices," *Parallel Computing*, vol. 36, no. 2, pp. 117–128, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167819110000049.

[39] R. A. Sweet, "A cyclic reduction algorithm for solving block tridiagonal systems of arbitrary dimension," *SIAM Journal on Numerical Analysis*, vol. 14, no. 4, pp. 706–720, 1977. [Online]. Available: http://www.jstor.org/stable/2156489 (accessed Jun. 21, 2022).

[40] G. Grosso, S. Moroni, and G. P. Parravicini, "Electronic structure of the inas-gasb superlattice studied by the renormalization method," *Phys. Rev. B*, vol. 40, pp. 12 328–12 337, 18 December 1989. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.40.12328.

[41] P. B. Vyas, C. Naquin, H. Edwards, M. Lee, W. G. Vandenberghe, and M. V. Fischetti, "Theoretical simulation of negative differential transconductance in lateral quantum well nmos devices," *Journal of Applied Physics*, vol. 121, no. 4, p. 044 501, 2017. [Online]. Available: https://doi.org/10.1063/1.4974469.

[42] *?gesv*, en. [Online]. Available: https://www.intel.com/content/www/us/en/develop/documentation/onemkl-developer-reference-c/top/lapack-routines/lapack-linear-equation-routines/lapack-linear-equation-driver-routines/gesv.html (accessed August 29, 2022).

[43] M. Zaman, K. Tanahashi, and S. Tanaka, "Pyqubo: Python library for qubo creation," *IEEE Transactions on Computers*, 2021.

[44] K. Tanahashi, S. Takayanagi, T. Motohashi, and S. Tanaka, "Application of ising machines and a software development for ising machines," *Journal of the Physical Society of Japan*, vol. 88, no. 6, p. 061 010, 2019.

[45] *QPU Topology — Ocean Documentation 5.4.0 documentation*. [Online]. Available: https://docs.ocean.dwavesys.com/en/stable/concepts/topology.html#topology-sdk (accessed October 3, 2022).

[46] J. Cai, W. G. Macready, and A. Roy, *A practical heuristic for finding graph minors*, 2014. [Online]. Available: https://arxiv.org/abs/1406.2741.

[47] *Ocean™ Developer Tools | D-Wave*, en-US. [Online]. Available: https://www.dwavesys.com/solutions-and-products/ocean/ (accessed November 8, 2022).

[48] T. Fukushima, "Precise and fast computation of inverse fermi–dirac integral of order 1/2 by minimax rational function approximation," *Applied Mathematics and Computation*, vol. 259, pp. 698–707, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0096300315003094.

[49] ——, "Precise and fast computation of fermi–dirac integral of integer and half integer order by piecewise minimax rational approximation," *Applied Mathematics and Computation*, vol. 259, pp. 708–729, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0096300315003033.

[50] ——, "Precise and fast computation of generalized fermi–dirac integral by parameter polynomial approximation," *Applied Mathematics and Computation*, vol. 270, pp. 802–807, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0096300315011509.

[51] M. Vasicek, "Advanced macroscopic transport models," Ph.D. dissertation, TU Wien, 2009. [Online]. Available: https://www.iue.tuwien.ac.at/phd/vasicek/diss.html.

[52] W. Shockley, *Electrons and Holes in Semiconductors*. London: Van Nostrand, 1951.

[53] W. Van Roosbroeck, "Theory of the flow of electrons and holes in germanium and other semiconductors," *The Bell System Technical Journal*, vol. 29, no. 4, pp. 560–607, 1950.

[54] J. Bude, "Mosfet modeling into the ballistic regime," in *2000 International Conference on Simulation Semiconductor Processes and Devices (Cat. No.00TH8502)*, 2000, pp. 23–26.

[55] D. Bohm, "A suggested interpretation of the quantum theory in terms of "hidden" variables. i," *Phys. Rev.*, vol. 85, pp. 166–179, 2 January 1952. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.85.166.

[56] J.-R. Zhou and D. Ferry, "Simulation of ultra-small gaas mesfet using quantum moment equations," *IEEE Transactions on Electron Devices*, vol. 39, no. 3, pp. 473–478, 1992.

[57] D. Ferry, R. Akis, and D. Vasileska, "Quantum effects in mosfets: Use of an effective potential in 3d monte carlo simulation of ultra-short channel devices," in *International Electron Devices Meeting 2000. Technical Digest. IEDM (Cat. No.00CH37138)*, 2000, pp. 287–290.

[58] J. Weinbub and D. K. Ferry, "Recent advances in wigner function approaches," *Applied Physics Reviews*, vol. 5, no. 4, p. 041 104, 2018. [Online]. Available: https://doi.org/10.1063/1.5046663.

[59] E. Wigner, "On the quantum correction for thermodynamic equilibrium," *Phys. Rev.*, vol. 40, pp. 749–759, 5 Jun. 1932. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.40.749.

[60] C. Jacoboni and P. Bordone, "The wigner-function approach to non-equilibrium electron transport," *Reports on Progress in Physics*, vol. 67, no. 7, pp. 1033–1071, Jun. 2004. [Online]. Available: https://doi.org/10.1088/0034-4885/67/7/r01.

[61] L. P. Kadanoff and G. Baym, *Quantum Statistical Mechanics: Green's Function Methods in Equilibrium and Nonequilibrium Problems*. Boca Raton: CRC Press, Jun. 2019.

[62] L. V. Keldysh, "DIAGRAM TECHNIQUE FOR NONEQUILIBRIUM PROCESSES," *SOVIET PHYSICS JETP*, vol. 20, no. 4, pp. 1018–1026, 1965.

[63] M. Luisier, "Atomistic simulation of transport phenomena in nanoelectronic devices," *Chem. Soc. Rev.*, vol. 43, pp. 4357–4367, 13 2014. [Online]. Available: http://dx.doi.org/10.1039/C4CS00084F.

[64] M. Calderara, S. Brück, A. Pedersen, M. H. Bani-Hashemian, J. VandeVondele, and M. Luisier, "Pushing back the limit of ab-initio quantum transport simulations on hybrid supercomputers," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ACM, November 2015. [Online]. Available: https://doi.org/10.11452f2807591.2807673.

[65] R. Venugopal, Z. Ren, S. Datta, M. S. Lundstrom, and D. Jovanovic, "Simulating quantum transport in nanoscale transistors: Real versus mode-space approaches," *Journal of Applied Physics*, vol. 92, no. 7, pp. 3730–3739, 2002. [Online]. Available: https://doi.org/10.1063/1.1503165.

[66] E. Polizzi and N. Ben Abdallah, "Subband decomposition approach for the simulation of quantum electron transport in nanostructures," *J. Comput. Phys.*, vol. 202, no. 1, pp. 150–180, January 2005. [Online]. Available: https://doi.org/10.1016/j.jcp.2004.07.003.

[67] N. Ben Abdallah, M. Mouis, and C. Negulescu, "An accelerated algorithm for 2d simulations of the quantum ballistic transport in nanoscale mosfets," *Journal of Computational Physics*, vol. 225, no. 1, pp. 74–99, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002199910600578X.

[68] J. Kestyn and E. Polizzi, "From fundamental first-principle calculations to nanoengineering applications: A review of the nessie project," *IEEE Nanotechnology Magazine*, vol. 14, no. 6, pp. 52–C3, 2020.

[69] P. Hohenberg and W. Kohn, "Inhomogeneous electron gas," *Phys. Rev.*, vol. 136, B864–B871, 3B November 1964. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.136.B864.

[70] B. Bauer, S. Bravyi, M. Motta, and G. K.-L. Chan, "Quantum algorithms for quantum chemistry and quantum materials science," *Chemical Reviews*, vol. 120, no. 22, pp. 12 685–12 717, October 2020. [Online]. Available: https://doi.org/10.1021%2Facs.chemrev.9b00829.

[71] P. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134.

[72] H.-S. Zhong, H. Wang, Y.-H. Deng, M.-C. Chen, L.-C. Peng, Y.-H. Luo, J. Qin, D. Wu, X. Ding, Y. Hu, P. Hu, X.-Y. Yang, W.-J. Zhang, H. Li, Y. Li, X. Jiang, L. Gan, G. Yang, L. You, Z. Wang, L. Li, N.-L. Liu, C.-Y. Lu, and J.-W. Pan, "Quantum computational advantage using photons," eng, *Science (American Association for the Advancement of Science)*, vol. 370, no. 6523, pp. 1460–1463, 2020.

[73] A. M. Childs, R. Kothari, and R. D. Somma, "Quantum algorithm for systems of linear equations with exponentially improved dependence on precision," *SIAM Journal on Computing*, vol. 46, no. 6, pp. 1920–1950, 2017. [Online]. Available: https://doi.org/10.1137/16M1087072.

[74] H. Wang and H. Xiang, "A quantum eigensolver for symmetric tridiagonal matrices," *Quantum Information Processing*, vol. 18, no. 3, p. 93, February 2019. [Online]. Available: https://doi.org/10.1007/s11128-019-2211-z.

[75] S. Sho and S. Odanaka, "Parallel domain decomposition methods for a quantum-corrected drift–diffusion model for mosfet devices," *Computer Physics Communications*, vol. 237, pp. 8–16, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465518303771.

[76] E. Polizzi and A. H. Sameh, "A parallel hybrid banded system solver: The spike algorithm," *Parallel Computing*, vol. 32, no. 2, pp. 177–194, 2006, Parallel Matrix Algorithms and Applications (PMAA'04). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167819105001353.

[77] P. Amestoy, I. Duff, and J.-Y. L'Excellent, "Multifrontal parallel distributed symmetric and unsymmetric solvers," *Computer Methods in Applied Mechanics and Engineering*, vol. 184, no. 2, pp. 501–520, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S004578259900242X.

[78] M. L. Van de Put, M. V. Fischetti, and W. G. Vandenberghe, "Scalable atomistic simulations of quantum electron transport using empirical pseudopotentials," *Computer Physics Communications*, vol. 244, pp. 156–169, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465519301961.

[79] R. J. Singh, *Solid state physics*, English. New Delhi, India: Pearson, 2011.

Appendices

A Effective mass approximation

In this project the effective mass approximation is used. This allows us to threat the motion of electrons in a solid as if they were free electrons, just with a different mass called the effective mass m^*. So the effect of the solid on the electron is that it changes the effective mass of the electron. When an electron moves in a solid under an applied bias, it will feel repulsive forces from negative charged ions in the solid, slowing down its motion. When the electron moves slower because of its interactions with the solid, it is said to have a heavier mass [79]. We are accounting for the solids impact on the electrons with the effective mass. The effective mass can either be much bigger or much smaller than the free electron mass, depending on the solid.

The electrons that conduct the current all belong to a certain energy band called the conduction band. To understand what an energy band is, we will begin by considering the energy levels of an isolated atom. The electrons in a single atom occupy discrete energy levels called atomic orbitals. Joining two atoms forms a diatomic molecule, where the single atom energy levels are split into two close levels. Forming a larger molecule, or a solid, of say n atoms, each energy level will be split into n new levels. These new levels will be so close to each other that they can be treated as a continuous band, see figure A.1.

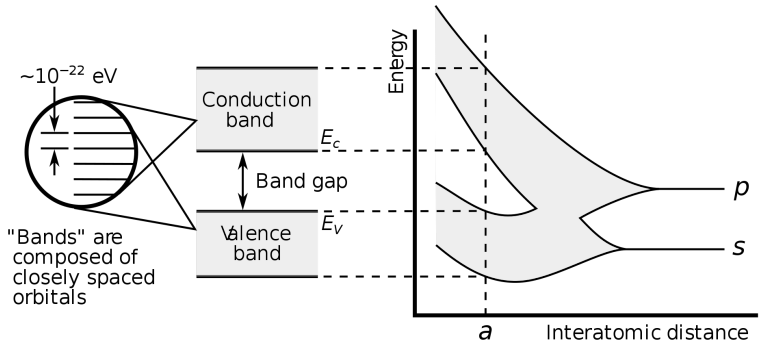


Figure A.1: The electronic band structure of a solid. The graph to the right shows the energy levels of the atoms as a function of the lattice spacing a (i.e. the distance between the atoms in the solid). For a long lattice spacing, each atom will have the single atom p and s orbitals. Bringing the atoms closer to each other, the orbitals begin to overlap. Because of Pauli's exclusion principle, the orbitals have to split such that no two electrons have the same quantum numbers, forming the energy bands. A solid is typically compromised of roughly $n \approx 10^{22}$ atoms, leading to $\sim 10^{22}$ energy levels in each band. [1]

The conducting electrons effective mass m^* are related to the dispersion relation at the conduction band minima E_C of the material by the following equation

$$\frac{1}{m^*} = \frac{1}{\hbar^2}\frac{\partial E_C}{\partial \mathbf{k}} \tag{A.1}$$

where $\hbar$ is the reduced Planck constant, E_C is conduction band energy and $\mathbf{k}$ is the wave vector. In this project, silicon is the semiconductor material in the transistor. Silicon has different effective masses in different directions. The effective mass in the longitudinal direction is $0.92m_e$ and in the transverse directions it is $0.19m_e$. Using the dispersion relation in equation A.1, the conduction band edges can be calculated. They form six ellipsoidal surfaces, called valleys, which can be seen in figure A.2. The effective mass are found empirically with the cyclotron resonance experiment. [79]

[1] Figure is from URL: https://commons.wikimedia.org/wiki/File:Solid_state_electronic_band_structure.svg

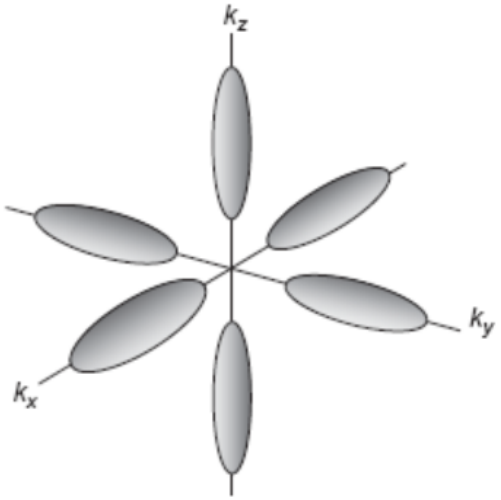


Figure A.2: Silicon conduction band valleys. [2]

B QTBM

B.1 Quantum Transmitting Boundary Method

The Quantum Transmitting Boundary Method (QTBM) is used to solve the 2D effective mass Schrödinger equation. It was developed by Craig S. Lent and David J. Kirkner and their work is presented in [1]. It is a numerical algorithm for finding current carrying states which are solutions to the open boundaries 2D Schrödinger equation with a finite element discretization.

B.1.1 Problem formulation

The problem region which is solved for can be divided into a "device" region Ω_0 and several lead regions extending to infinity, $\Omega_1, \Omega_2, ..., \Omega_n$. All this can be seen in figure B.1. The boundaries between the leads and the device are denoted $\Gamma_1, \Gamma_2, ..., \Gamma_n$ and the boundary of the device that do not share a boundary with a lead is denoted Γ_0. For each lead we introduce a local coordinate system (η_i, ξ_i), where $\hat{\eta}_i$ is parallel to the lead walls and $\hat{\xi}_i$ points away from the device. $\hat{\xi}_i$ is perpendicular to $\hat{\eta}_i$.

[2] Figure is from **Figure 9.42** in R.J Singh's textbook *Solid State Physics* [79]

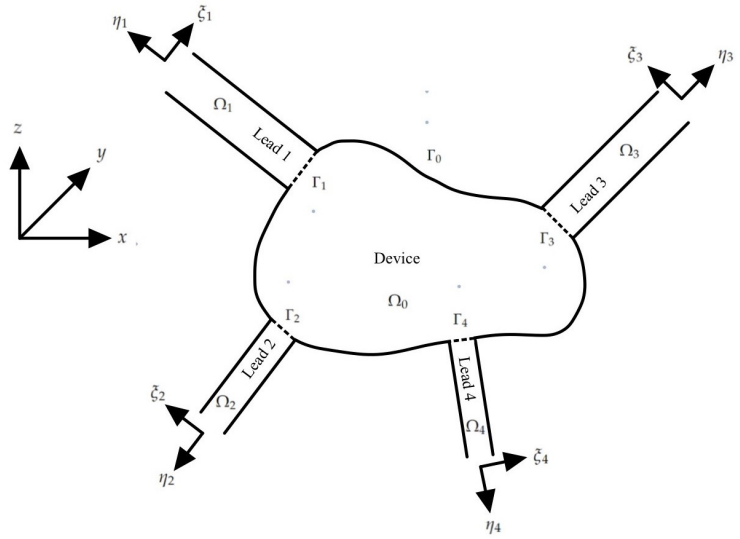


Figure B.1: Problem geometry

The governing equation is the 2D effective-mass Schrödinger equation defined on $\Omega \equiv \Omega_0 \cup \Omega_1 \cup \Omega_2 ... \cup \Omega_n$. The problem can be formulated as follows:
Given a total energy E, the potential energy in every region $V_s(x,z)$, $s = 0,1,2,...,n$, and complex amplitudes corresponding to incoming modes in each lead a_m^s, find:

$$\psi_0 \in \mathbb{C}^2(\Omega_0), \psi_0 \in \mathbb{C}^2(\Omega_1), ..., \psi_n \in \mathbb{C}^2(\Omega_n)$$

(ψ_0 is equal to $\phi_\beta^{s,v}$ in equation 2.2) such that

$$-\frac{\hbar}{2}\left[\frac{\partial}{\partial x}\left(\frac{1}{m_x^*}\frac{\partial \psi_s(x,z)}{\partial x}\right) + \frac{\partial}{\partial z}\left(\frac{1}{m_z^*}\frac{\partial \psi_s(x,z)}{\partial z}\right)\right] + V_s(x,z)\psi_s(x,z) = E\psi_s, \quad (x,z) \in \Omega_s \tag{B.1}$$

and with the following boundary conditions

$$\psi_0 = \psi_s \text{ on } \Gamma_s, \tag{B.2}$$

$$\nabla\psi_0 \cdot \hat{\eta_s} = \nabla\psi_s \cdot \hat{\eta_s} \text{ on } \Gamma_s, \tag{B.3}$$

$$\psi_0 = 0 \text{ on } \Gamma_0 \equiv \partial\Omega_0 - \sum_s \Gamma_s \tag{B.4}$$

$$\psi_s = 0 \text{ on } \Gamma_{s0} \equiv \partial\Omega_s - \Gamma_s \tag{B.5}$$

$$\psi_s \text{ is bounded as } \sqrt{x^2+z^2} \to \infty. \tag{B.6}$$

The ∇ operator is the gradient and $\partial\Omega_i$ denotes the boundary of region Ω_i. The boundary between Ω_0 and Ω_s is defined as

$$\Gamma_s = \{(\eta_s, \xi_s) | \xi_s \in (0, d_s), \eta_s = 0\}, \tag{B.7}$$

where d_s is the width of the lead. The QTBM solution algorithm is valid for any potential $V_0(x,z)$ in the device region. The potential in the leads $V_s(x,z)$ is required to be independent of its η_s coordinate. This follows from it assumed infinite length. The QTBM's goal is to formulate boundary conditions for the lead boundaries Γ_s which allows us to specify a incoming flux in every lead and find a solution of the Schrödinger equation in the device region Ω_0 only.

B.1.2 Solution in the leads

Since we have required the leads to be uniform along their lengths, the potential in the lead vary only across its width, $V_s(\eta_s,\xi_s) = V_s(\xi_s)$. For simple square-well leads, we have $V_s(\xi_s) = 0$, which is used in this project. This means that the Schrödinger equation is separable into several coupled one-dimensional problems and that it can be solved analytically. Its general solution can be written as

$$\begin{aligned}\psi_s(\eta_s,\psi_s) = &\sum_{m=1}^{N^s}\left[a_m^s\chi_m^s(\xi_s)e^{-ik_m^s\eta_s} + b_m^s\chi_m^s(\xi_s)e^{ik_m^s\eta_s}\right] \\ &+ \sum_{m=N^s+1}^{\infty}\left[a_m^s\chi_m^s(\xi_s)e^{k_m^s\eta_s} + b_m^s\chi_m^s(\xi_s)e^{-k_m^s\eta_s}\right]\end{aligned} \tag{B.8}$$

where χ_m^s is the mth eigenstate of the one-dimensional Schrödinger equation in lead s,

$$-\frac{\hbar}{2}\left[\frac{\partial}{\partial\xi_s}\left(\frac{1}{m^*_{\xi_s}}\frac{\partial\chi_m^s(\xi_s)}{\partial\xi_s}\right)\right] + V_s(\xi_s)\chi_m^s(\xi_s) = E_m^s\chi_m^s(\xi_s). \tag{B.9}$$

The a_m^s are the injection amplitudes of the *incoming* traveling-wave states in the first sum and the injection amplitudes of *incoming* evanescent states in the second sum, and is an input to the problem which we will calculate. N_m^s is the number of allowed traveling-wave modes in lead s, defined at the end of this section. The b_m^s's in the first sum are coefficients of the traveling-wave states and the coefficients of the *outgoing* evanescent states in the second sum. The b_m^s's are unknown and must be calculated. We should mention that we use incoming evanescent states, which are a modified version of the traditional QTBM [1], employed by Laux et al. in [13]. The eigenstates χ_m^s are chosen to be orthonormal

$$\int_0^{d_s}[\chi_m^s(\xi_s)]^*\chi_n^s(\xi_s)d\xi_s = \delta_{mn}, \tag{B.10}$$

where δ_{mn} is the Dirac delta. If the potential $V_s(\xi_s)$ takes the form of infinite square-wells and the effective mass $m^*_{\xi_s}$ is constant, we have the following analytical solution to equation B.9

$$\chi_m^s(\xi_s) = \sqrt{2/d_s}sin[(m\pi/d_s)\xi_s], \tag{B.11}$$

and the wave vector for the mth mode in lead s is given by

$$k_m^s = \sqrt{|(2m^*_{\xi_s}/\hbar^2)(E - E_m^s)|}. \tag{B.12}$$

The number of traveling waves N^i is defined as the maximum m such that $E_m^s < E$.

B.1.3 Boundary conditions

Now, let's turn our attention to the boundary condition at the lead-device intersections Γ_s. We require continuity of both the wave function and its normal derivative (see equation B.2 and B.3). Inserting the general solution, equation B.8, into the normal derivative continuity condition we obtain

$$\begin{aligned}\nabla\psi_0(\mathbf{r})\cdot\hat{\eta_s}\Big|_{\mathbf{r}\in\Gamma_s} &= \nabla\psi_s(\mathbf{r})\cdot\hat{\eta_s}\Big|_{\mathbf{r}\in\Gamma_s} = \frac{\partial}{\partial\eta_s}\psi_s(\eta_s,\xi_s)\Big|_{\eta_s=0} \\ &= \sum_{m=1}^{N^s} -ia_m^s k_m^s \chi_m^s(\xi_s) + ib_m^s k_m^s \chi_m^s(\xi_s) \\ &+ \sum_{m=N^s+1}^{\infty} a_m^s k_m^s \chi_m^s(\xi_s) - b_m^s k_m^s \chi_m^s(\xi_s)\end{aligned} \tag{B.13}$$

Setting $\eta_s = 0$ and using the orthogonality of the χ_m^i's, we can compute the b_m^i's

$$b_m^s = \int_0^{d_s} \chi_m^s(\xi_s)\psi_s(\eta_s = 0,\xi_s)d\xi_s - a_m^s. \tag{B.14}$$

Insert this into equation B.13

$$\begin{aligned}\frac{\partial}{\partial\eta_s}\psi_s(\eta_s,\xi_s)\Big|_{\eta_s=0} &= \sum_{m=1}^{N^s} ik_m^s\chi_m^s(\xi_s) \\ &\times\left(-2a_m^s + \int_0^{d_s}\chi_m^s(\xi_s)\psi_s(\eta_s=0,\xi_s)d\xi_s\right) \\ &- \sum_{m=N^s+1}^{\infty} k_m^s\chi_m^s(\xi_s)\times\left(-2a_m^s + \int_0^{d_s}\chi_m^s(\xi_s)\psi_s(\eta_s=0,\xi_s)d\xi_s\right) \\ &= f_s[\xi_s,\psi_s(\eta_s=0,\xi_s)]\end{aligned} \tag{B.15}$$

Now using equation B.2, the continuity of the wave function itself, we can replace the ψ_s's with the ψ_0's since we are only looking at the boundary. Hence, we get an boundary condition for ψ_0 and its normal derivative:

$$
\begin{aligned}
\nabla\psi_0(\mathbf{r})\cdot\hat{\eta}_s &= f_s[\xi_s,\psi_0(\eta_s=0,\xi_s)] \\
&= \sum_{m=1}^{N^s} ik_m^s\chi_m^s(\xi_s) \\
&\times\left(-2a_m^s+\int_0^{d_s}\chi_m^s(\xi_s)\psi_0(\eta_s=0,\xi_s)d\xi_s\right) \\
&-\sum_{m=N^s+1}^{\infty}k_m^s\chi_m^s(\xi_s)\times\left(-2a_m^s+\int_0^{d_s}\chi_m^s(\xi_s)\psi_0(\eta_s=0,\xi_s)d\xi_s\right)
\end{aligned}
\tag{B.16}
$$

The functional f_s, defined in equation B.16, is the boundary condition which will be used to formulate the boundary-value problem for the current-carrying states. Observe that the value of the normal derivative of the wave-function at a certain point relates to all values of the wave-function along the boundary through the integral.

B.1.4 Weak variational form

Before discretizing the problem, we will use a weak variational form of the Schrödinger equation. The goal is to only discretize the wave function in the device domain Ω_0 and using suitable boundary conditions to match the wave function and normal derivative to the solution in the leads, where we have analytical solutions. Starting with the 2D time-independent Schrödinger equation for the wave function in the device region ψ_0

$$
-\frac{\hbar^2}{2}\nabla\left(\mathbf{M}^{-1}\nabla\psi_0(x,z)\right)+V(x,z)\psi_0(x,z)=E\psi_0(x,z). \tag{B.17}
$$

where $\mathbf{M}^{-1}$ is diagonal inverse effective mass matrix, defined as

$$
\mathbf{M}^{-1}=\begin{bmatrix}\frac{1}{m_x^*} & 0\\ 0 & \frac{1}{m_z^*}\end{bmatrix} \tag{B.18}
$$

We multiply equation B.17 with an arbitrary test function $\bar{\psi}$ and integrate over the device region Ω_0

$$
-\frac{\hbar^2}{2}\int_{\Omega_0}\bar{\psi}\nabla\left(\mathbf{M}^{-1}\nabla\psi_0(x,z)\right)+\int_{\Omega_0}\bar{\psi}(V-E)\psi_0=0. \tag{B.19}
$$

Choosing the same boundary conditions for the test function $\bar{\psi}$ as for ψ_0, i.e. $\bar{\psi}=0$ on Γ_0, we can use Green's first identity and obtain the following

$$\frac{\hbar^2}{2}\int_{\Omega_0}\nabla\bar{\psi}\mathbf{M}^{-1}\nabla\psi_0(x,z)+\int_{\Omega_0}\bar{\psi}(V-E)\psi_0 = \frac{\hbar^2}{2}\oint_{\Gamma}\bar{\psi}(\mathbf{M}^{-1}\nabla\psi_0\cdot\hat{n})d\Gamma$$
$$= \frac{\hbar^2}{2}\sum_s\int_{\Gamma_s}\bar{\psi}\left[\frac{1}{m_x^*}\frac{\partial\psi_0}{\partial x},\frac{1}{m_z^*}\frac{\partial\psi_0}{\partial z}\right]\cdot\hat{n}d\Gamma_s \tag{B.20}$$

In the last equality we used the fact that $\bar{\psi}$ is zero on the Γ_0 boundary. The problem formulation in section B.1.1 can now be recast in the form of a weak variational form using the boundary conditions in equation B.16

$$\frac{\hbar^2}{2}\int_{\Omega_0}\nabla\bar{\psi}\mathbf{M}^{-1}\nabla\psi_0+\int_{\Omega_0}\bar{\psi}(V-E)\psi_0 = \frac{\hbar^2}{2}\sum_s\int_{\Gamma_s}\bar{\psi}\frac{1}{m_{\eta_s}^*}f_s[\xi_s,\psi_0]d\Gamma_s \tag{B.21}$$

where

$$\begin{aligned} f_s[\xi_s,\psi_0] = &\sum_{m=1}^{N^s} ik_m^s\chi_m^s(\xi_s) \\ &\times\left(-2a_m^s+\int_0^{d_s}\chi_m^s(\xi_s)\psi_0(\eta_s=0,\xi_s)d\xi_s\right) \\ &-\sum_{m=N^s+1}^{\infty}k_m^s\chi_m^s(\xi_s)\times\left(\int_0^{d_s}\chi_m^s(\xi_s)\psi_0(\eta_s=0,\xi_s)d\xi_s\right) \end{aligned} \tag{B.22}$$

B.1.5 Finite element discretization

The next step is to solve this reformulated problem in equation B.21. We do this with a finite element discretization, using a mesh with N nodal points, $\mathbf{r}_1,\mathbf{r}_2,...,\mathbf{r}_N$. Each nodal point has an associated shape function $\varphi_i(\mathbf{r})$ with the following property

$$\varphi_i(\mathbf{r_j}) = \delta_{i,j} \tag{B.23}$$

where $\delta_{i,j}$ is the Kronecker delta. Using the shape functions as a basis, we can do an approximate expansion of the wave function ψ_0 in the shape function basis

$$\psi_0(\mathbf{r}) = \sum_i\psi(\mathbf{r}_i)\varphi_i(\mathbf{r}) = \sum_i u_i\varphi_i(\mathbf{r}) \tag{B.24}$$

or, in a vector form

$$\psi_0(\mathbf{r}) = \mathbf{N}(\mathbf{r})\cdot\mathbf{u} \tag{B.25}$$

where $\mathbf{N}(\mathbf{r})$ is a $(1\times N)$ row vector of shape functions

$$\mathbf{N}(\mathbf{r}) = [\varphi_1(\mathbf{r}),\varphi_2(\mathbf{r}),...,\varphi_N(\mathbf{r})] \tag{B.26}$$

and $\mathbf{u}$ is a $(N \times 1)$ column vector containing the unknown nodal values of the wave function ψ_0. We can also expand the gradient of the wave vectors in the shape function basis in a similar manner

$$\nabla\psi_0(\mathbf{r}) = \begin{bmatrix} \frac{\partial\psi_0}{\partial x} \\ \frac{\partial\psi_0}{\partial z} \end{bmatrix} = \mathbf{B}(\mathbf{r}) \cdot \mathbf{u} \tag{B.27}$$

where $\mathbf{B}(\mathbf{r})$ is a $(2 \times N)$ matrix of derivatives of the shape functions, given by

$$\mathbf{B}(\mathbf{r}) = \begin{bmatrix} \partial_x\varphi_1(\mathbf{r}) & \partial_x\varphi_2(\mathbf{r}) & ... & \partial_x\varphi_N(\mathbf{r}) \\ \partial_y\varphi_1(\mathbf{r}) & \partial_y\varphi_2(\mathbf{r}) & ... & \partial_y\varphi_N(\mathbf{r}) \end{bmatrix}. \tag{B.28}$$

In a similar fashion we can do a approximate expansion of the test function $\bar{\psi}$ in the shape function basis

$$\bar{\psi} = \bar{\mathbf{u}}^T \cdot \mathbf{N}^T \tag{B.29}$$

$$\nabla\bar{\psi} = \bar{\mathbf{u}}^T \cdot \mathbf{B}^T \tag{B.30}$$

where $\bar{\mathbf{u}}$ is a $(N \times 1)$ column vector of nodal values of $\bar{\psi}$. We can insert these approximate expansions into the Schrödinger equation, see equation B.21

$$\begin{aligned} \bar{\mathbf{u}}^T \left(\int_{\Omega_0} \frac{\hbar^2}{2} \mathbf{B}^T(\mathbf{r}) \mathbf{M}^{-1} \mathbf{B}(\mathbf{r}) d\mathbf{r} \right) \mathbf{u} + \bar{\mathbf{u}}^T \left(\int_{\Omega_0} \left[V(\mathbf{r}) - E \right] \mathbf{N}^T(\mathbf{r}) \mathbf{N}(\mathbf{r}) d\mathbf{r} \right) \mathbf{u} \\ = \frac{\hbar^2}{2} \sum_s \left(\int_{\Gamma_s} \bar{\psi}(\mathbf{r}) \frac{1}{m^*_{\eta_s}} f_s \left[\xi_s, \psi_0(0, \xi_i) \right] d\Gamma_s \right). \end{aligned} \tag{B.31}$$

The right-hand side is not yet discretized, but it is something we will do soon. We define three $(N \times N)$ matrices $\mathbf{T}$, $\mathbf{V}$ and $\mathbf{D}$ as follows

$$\mathbf{T} = \frac{\hbar^2}{2} \int_{\Omega_0} \mathbf{B}^T(\mathbf{r}) \mathbf{M}^{-1} \mathbf{B}(\mathbf{r}) d\mathbf{r} \tag{B.32}$$

$$\mathbf{V} = \int_{\Omega_0} V(\mathbf{r}) \mathbf{N}^T(\mathbf{r}) \mathbf{N}(\mathbf{r}) d\mathbf{r} \tag{B.33}$$

$$\mathbf{D} = \int_{\Omega_0} \mathbf{N}^T(\mathbf{r}) \mathbf{N}(\mathbf{r}) d\mathbf{r} \tag{B.34}$$

where $\mathbf{T}$ represents the kinetic term, $\mathbf{V}$ the potential term and $E\mathbf{D}$ the energy in the Schrödinger equation. The partially discretized Schrödinger can be written as

$$\bar{\mathbf{u}}^T\left(\mathbf{T}+\mathbf{V}-E\mathbf{D}\right)\mathbf{u}=\frac{\hbar^2}{2}\sum_s\left(\int_{\Gamma_s}\bar{\psi}(\mathbf{r})\frac{1}{m^*_{\eta_s}}f_s\left[\xi_s,\psi_0(0,\xi_s)\right]d\Gamma_s\right). \tag{B.35}$$

B.1.5.1 Self-energies

Now, let us discretize the righ-hand side. First, let $\mathbf{u}_s$ be the projection of $\mathbf{u}$ onto Γ_s. $\mathbf{u}_s$ will have length M_s, where M_s is the number of nodal points on the boundary Γ_s. Also, define a row vector $\mathbf{N}_s(\xi_s)$, with shape $(1\times M_s)$, of shape functions on the boundary so that we can expand $\psi_0(\mathbf{r}\in\Gamma_s)$ in the following way

$$\psi_0(\mathbf{r}\in\Gamma_s)=\psi_0(\eta_s=0,\xi_s)=\mathbf{N}_s(\xi_s)\cdot\mathbf{u}_s. \tag{B.36}$$

We also define a $(1\times M_s)$ column vector $\mathbf{N}_{s,m}$, given by

$$\mathbf{N}_{s,m}\equiv\int_0^{d_s}\chi^s_m(\xi_s)\mathbf{N}_s(\xi_s)d\xi_s \tag{B.37}$$

The right-hand side can be written as

$$\begin{aligned}\frac{\hbar^2}{2}\sum_s\left(\int_{\Gamma_s}\bar{\psi}(\mathbf{r})\frac{1}{m^*_{\eta_s}}f_s\left[\mathbf{r},\mathbf{u}\right]d\Gamma_s\right)=\frac{\hbar^2}{2}\sum_s\bar{\mathbf{u}}^T\frac{1}{m^*_{\eta_s}}\Bigg[&\sum_{m=1}^{N_s}-2ia^s_mk^s_m\int_0^{d_s}N_s^T(\xi_s)\chi^i_m(\xi_s)d\xi_s\\&+\sum_{m=1}^{N^s}ik^s_m\left(\int_0^{d_s}\mathbf{N}_s^T(\xi_s)\chi^s_m(\xi_s)d\xi_s\right)\left(\int_0^{d_s}\mathbf{N}_s^T(\xi_s)\chi^s_m(\xi_s)d\xi_s\right)\mathbf{u}_s\\&+\sum_{m=N^s+1}^{\infty}2a^s_mk^s_m\int_0^{d_s}N_s^T(\xi_s)\chi^s_m(\xi_s)d\xi_s\\&-\sum_{m=N^s+1}^{\infty}k^s_m\left(\int_0^{d_s}\mathbf{N}_s^T(\xi_s)\chi^s_m(\xi_s)d\xi_s\right)\left(\int_0^{d_s}\mathbf{N}_s^T(\xi_s)\chi^s_m(\xi_s)d\xi_s\right)\mathbf{u}_s\Bigg]\end{aligned} \tag{B.38}$$

and using equation B.37 we obtain

$$\begin{aligned}\frac{\hbar^2}{2}\sum_s\left(\int_{\Gamma_s}\bar{\psi}(\mathbf{r})\frac{1}{m^*_{\eta_s}}f_s\left[\mathbf{r},\mathbf{u}\right]d\Gamma_s\right)=\frac{\hbar^2}{2}\sum_s\bar{\mathbf{u}}^T\frac{1}{m^*_{\eta_s}}\Bigg[&\sum_{m=1}^{N^s}-2ia^i_mk^i_m\mathbf{N}^T_{s,m}\\+\left(\sum_{m=1}^{N^s}ik^s_m\mathbf{N}^T_{s,m}\mathbf{N}_{s,m}\right)\mathbf{u}_s+&\sum_{m=N^s+1}^{\infty}2a^s_mk^s_m\mathbf{N}^T_{s,m}-\left(\sum_{m=N^s+1}^{\infty}k^s_m\mathbf{N}^T_{s,m}\mathbf{N}_{s,m}\right)\mathbf{u}_s\Bigg].\end{aligned} \tag{B.39}$$

We define a vector $\mathbf{P}_i$ and matrix $\mathbf{C}_i$ as

$$\mathbf{P}_s\equiv-\frac{\hbar^2}{2m^*_{\eta_s}}\left(\sum_{m=1}^{N^s}2ia^s_mk^s_m\mathbf{N}^T_{s,m}-\sum_{m=N^s+1}^{\infty}2a^s_mk^s_m\mathbf{N}^T_{s,m}\right) \tag{B.40}$$

$$\mathbf{C}_s\equiv-\frac{\hbar^2}{2m^*_{\eta_s}}\left(\sum_{m=1}^{N^s}ik^s_m\mathbf{N}^T_{s,m}\mathbf{N}_{s,m}-\sum_{m=N^i}^{\infty}k^s_m\mathbf{N}^T_{s,m}\mathbf{N}_{s,m}\right). \tag{B.41}$$

When actually computing the sums we have to set an upper limit on the infinite sum $\sum_{m=N^s+1}^{\infty}$. Since we would only obtain $M_s - 2$ different eigensolutions when solving the discretized lead Schrödinger equation (equation B.9), we set $M_s - 2$ as the upper limit (the subtraction of 2 is because we have Dirichlet boundary conditions at the ends of Γ_s). Using equation B.40 and B.41 the right-hand side can be written as

$$\sum_s \bar{\mathbf{u}}^T(\mathbf{P}_s - \mathbf{C}_s\mathbf{u}_s). \tag{B.42}$$

When solving the discretized Schrödinger equation we have to embed $\mathbf{P}_s$ into a $(N \times 1)$ column vector $\hat{\mathbf{P}}_s$ and $\mathbf{C}_s$ into a $(N \times N)$ matrix $\hat{\mathbf{C}}_i$. The discretized Schrödinger equation becomes

$$\bar{\mathbf{u}}^T\left(\mathbf{T} + \mathbf{V} - E\mathbf{D}\right)\mathbf{u} = \sum_i \bar{\mathbf{u}}^T\hat{\mathbf{P}}_s - \bar{\mathbf{u}}^T\hat{\mathbf{C}}_s\mathbf{u} \tag{B.43}$$

or

$$\sum_s \bar{\mathbf{u}}^T\left(\mathbf{T} + \mathbf{V} - E\mathbf{D} - \hat{\mathbf{C}}_s\right)\mathbf{u} = \sum_s \bar{\mathbf{u}}^T\hat{\mathbf{P}}_s \tag{B.44}$$

The $\bar{\mathbf{u}}^T$ vector stems from the arbitrary test function $\bar{\psi}$, so we can reduce the Schrödinger equation to

$$\left(\mathbf{T} + \mathbf{V} - E\mathbf{D} - \sum_s \hat{\mathbf{C}}_s\right)\mathbf{u} = \sum_s \hat{\mathbf{P}}_s. \tag{B.45}$$

Comparing to equation 2.4 where two leads is used, $\mathbf{H}^D$ is equal to the $\mathbf{T} + \mathbf{V}$, $\hat{\mathbf{C}}_1$ is equal to the left self-energy Σ_L and $\hat{\mathbf{C}}_2$ is equal to the right self-energy Σ_R, and $\mathbf{B}^m$ is equal to the injection vector $\sum_s \hat{\mathbf{P}}_s$. In the project we only consider injection from one lead at the time, such that we have the wave functions corresponding to one lead only, this is because the leads inject traveling modes independently without coherence. We have the equations to compute the different matrices in equation 2.4, but to actually be able to compute them we have to pick a set of shape functions, which we will do in section B.2. But first, we will compute the injection amplitudes a_m^s.

B.1.6 Injection amplitudes

To compute the injection amplitudes a_m^s we will make use of the standing-wave solutions ψ_β, obtained from equation 3.8. The amplitudes a_m^s can be computed with a similar approach as we computed the b_m^s amplitudes in equation B.14. Depending on the standing wave boundary condition used when computing one of the following equations are used when computing a_m^s. For the "sinus-like" solution we have the following equations

$$a_m^s = i\int_0^{d_s} \chi_m^s(\xi_s)\psi_\beta(\eta_s = 0, \xi_s)d\xi_s, \quad \text{for } 1 \leq m \leq N^s, \tag{B.46}$$

$$= \int_0^{d_s} \chi_m^s(\xi_s)\psi_\beta(\eta_s = 0, \xi_s)d\xi_s, \quad \text{for } m > N^s. \tag{B.47}$$

Since the eigenfunction for the "cosinus-like" solutions are zero at the lead boundary $\eta_s = 0$, we cannot use the expression above to obtain the injection amplitudes. Instead, we will opt for the normal derivative of the eigenfunctions at the lead boundary, which gives the following

$$a_m^s = \frac{i}{2k_m^s}\int_0^{d_s} \chi_m^s(\xi_s)\left.\frac{\partial\psi_\beta(\eta_s, \xi_s)}{\partial\eta_s}\right|_{\eta_s=0} d\xi_s, \quad \text{for } 1 \leq m \leq N^s, \tag{B.48}$$

$$= \frac{1}{2k_m^s}\int_0^{d_s} \chi_m^s(\xi_s)\left.\frac{\partial\psi_\beta(\eta_s, \xi_s)}{\partial\eta_s}\right|_{\eta_s=0} d\xi_s, \quad \text{for } m > N^s. \tag{B.49}$$

For a more detailed description, see Appendix **2. Finding the traveling eigencomponents** in Laux's and et al.'s work in [13].

B.2 Rectangular mesh elements

In this section we will present how the matrices associated with the discretization of the Schrödinger and Poisson equation are formed using the rectangular mesh elements. Consider the mesh element in figure B.2, where we have aligned the mesh rectangle with the x and z axis. The rectangular mesh elements have an area Δ. We have split the rectangle diagonally from the 1 corner to the 4 corner and defined shape functions for the resulting triangular meshes. The upper triangular has an area Δ_U and the lower triangle has an area Δ_L.

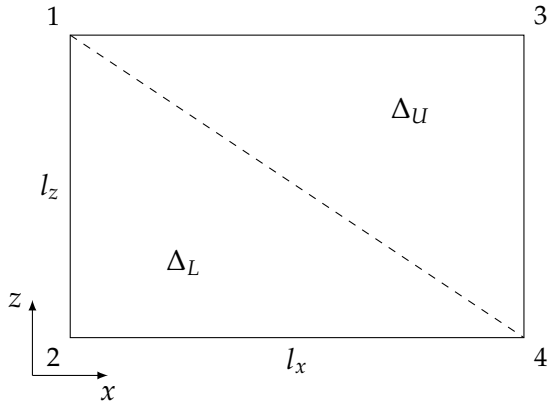
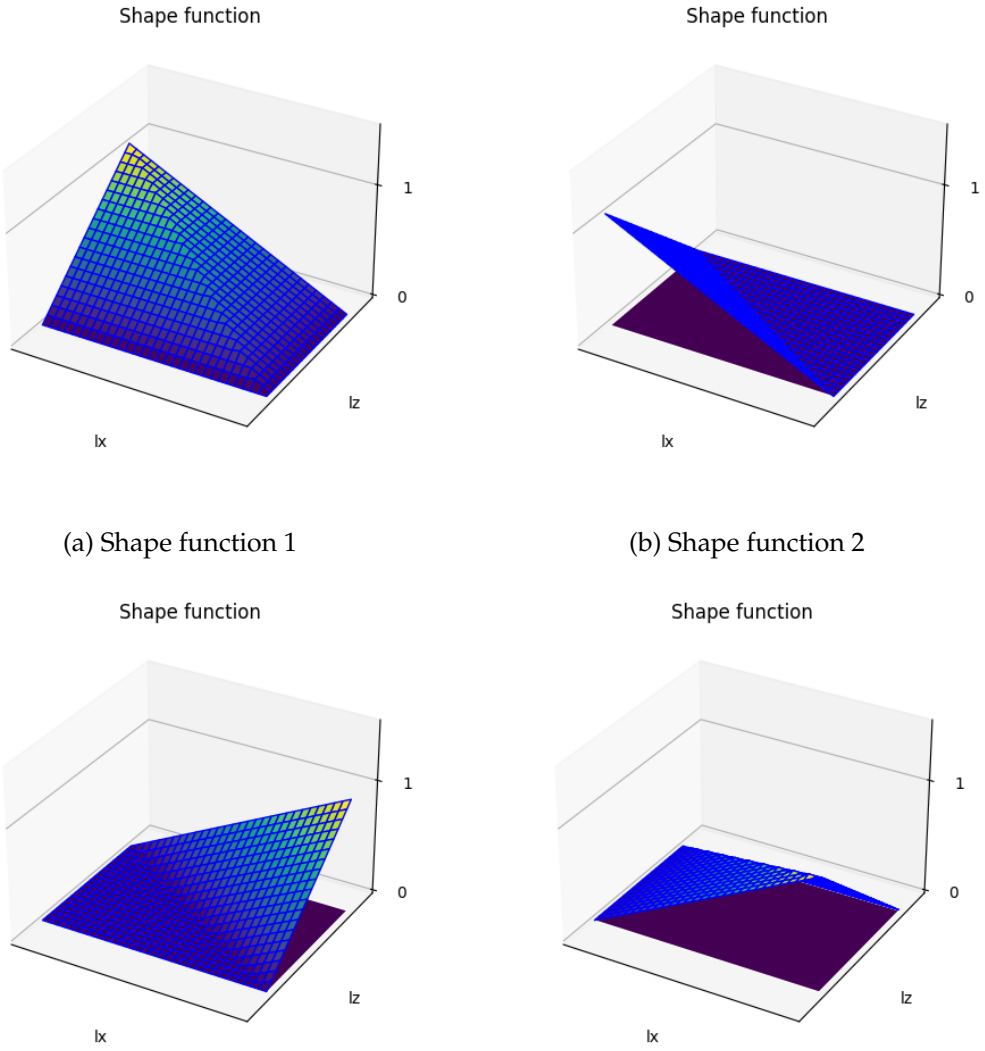


Figure B.2: Rectangular mesh element.

Associated to a nodal point k is a shape function φ_k, where $k = 1,..,4$. The shape function has unity value in node k and zero value in the other nodes by definition, see equation B.23. Since the nodal points 1 and 4 are part of both the upper and the lower triangular, they will have one shape function in each triangular. We use linear shape functions, $\varphi_k = a_k + b_k x + c_k z$, where where a_k, b_k and c_k are constants. The shape functions can be seen in figure B.3.



(a) Shape function 1

(b) Shape function 2

(c) Shape function 3

(d) Shape function 4

Figure B.3: Shape functions.

Setting the effective masses m_x^* and m_z^* to constants in eash mesh element, we can compute equation B.32 using the linear shape functions. This yields the following 4×4 matrix (neglecting all shape functions outside the mesh element, which will be zero)

$$\mathbf{T}_{rec} = \frac{\hbar^2}{2}\int_\Delta \mathbf{B}^T(\mathbf{r})\mathbf{M}^{-1}\mathbf{B}(\mathbf{r})d\mathbf{r}$$
$$= \frac{\hbar^2}{2}\int_\Delta \begin{bmatrix} \frac{\partial_x\varphi_1\partial_x\varphi_1}{m_x^*}+\frac{\partial_y\varphi_1\partial_y\varphi_1}{m_y^*} & \frac{\partial_x\varphi_1\partial_x\varphi_2}{m_x^*}+\frac{\partial_y\varphi_1\partial_y\varphi_2}{m_y^*} & \frac{\partial_x\varphi_1\partial_x\varphi_3}{m_x^*}+\frac{\partial_y\varphi_1\partial_y\varphi_3}{m_y^*} & \frac{\partial_x\varphi_1\partial_x\varphi_4}{m_x^*}+\frac{\partial_y\varphi_1\partial_y\varphi_4}{m_y^*} \\ \frac{\partial_x\varphi_2\partial_x\varphi_1}{m_x^*}+\frac{\partial_y\varphi_2\partial_y\varphi_1}{m_y^*} & \frac{\partial_x\varphi_2\partial_x\varphi_2}{m_x^*}+\frac{\partial_y\varphi_2\partial_y\varphi_2}{m_y^*} & \frac{\partial_x\varphi_2\partial_x\varphi_3}{m_x^*}+\frac{\partial_y\varphi_2\partial_y\varphi_3}{m_y^*} & \frac{\partial_x\varphi_2\partial_x\varphi_4}{m_x^*}+\frac{\partial_y\varphi_2\partial_y\varphi_4}{m_y^*} \\ \frac{\partial_x\varphi_3\partial_x\varphi_1}{m_x^*}+\frac{\partial_y\varphi_3\partial_y\varphi_1}{m_y^*} & \frac{\partial_x\varphi_3\partial_x\varphi_2}{m_x^*}+\frac{\partial_y\varphi_3\partial_y\varphi_2}{m_y^*} & \frac{\partial_x\varphi_3\partial_x\varphi_3}{m_x^*}+\frac{\partial_y\varphi_3\partial_y\varphi_3}{m_y^*} & \frac{\partial_x\varphi_3\partial_x\varphi_4}{m_x^*}+\frac{\partial_y\varphi_3\partial_y\varphi_4}{m_y^*} \\ \frac{\partial_x\varphi_4\partial_x\varphi_1}{m_x^*}+\frac{\partial_y\varphi_4\partial_y\varphi_1}{m_y^*} & \frac{\partial_x\varphi_4\partial_x\varphi_2}{m_x^*}+\frac{\partial_y\varphi_4\partial_y\varphi_2}{m_y^*} & \frac{\partial_x\varphi_4\partial_x\varphi_3}{m_x^*}+\frac{\partial_y\varphi_4\partial_y\varphi_3}{m_y^*} & \frac{\partial_x\varphi_4\partial_x\varphi_4}{m_x^*}+\frac{\partial_y\varphi_4\partial_y\varphi_4}{m_y^*} \end{bmatrix} \tag{B.50}$$

The derivatives of the linear shape functions are given by

$$\partial_x\varphi_1 = \begin{cases} -\frac{1}{l_x}, & \text{if } (x,z)\in\Delta_U, \\ 0, & \text{if } (x,z)\in\Delta_L, \end{cases} \quad \partial_y\varphi_1 = \begin{cases} 0 & ,\text{if } (x,z)\in\Delta_U \\ \frac{1}{l_z} & ,\text{if } (x,z)\in\Delta_L, \end{cases} \tag{B.51}$$

$$\partial_x\varphi_2 = \begin{cases} 0, & \text{if } (x,z)\in\Delta_U, \\ -\frac{1}{l_x}, & \text{if } (x,z)\in\Delta_L, \end{cases} \quad \partial_y\varphi_2 = \begin{cases} 0, & \text{if } (x,z)\in\Delta_U \\ -\frac{1}{l_z}, & \text{if } (x,z)\in\Delta_L, \end{cases} \tag{B.52}$$

$$\partial_x\varphi_3 = \begin{cases} \frac{1}{l_x}, & \text{if } (x,z)\in\Delta_U, \\ 0, & \text{if } (x,z)\in\Delta_L, \end{cases} \quad \partial_y\varphi_3 = \begin{cases} \frac{1}{l_z}, & \text{if } (x,z)\in\Delta_U \\ 0, & \text{if } (x,z)\in\Delta_L, \end{cases} \tag{B.53}$$

$$\partial_x\varphi_4 = \begin{cases} 0, & \text{if } (x,z)\in\Delta_U, \\ \frac{1}{l_x}, & \text{if } (x,z)\in\Delta_L, \end{cases} \quad \partial_y\varphi_4 = \begin{cases} -\frac{1}{l_z}, & \text{if } (x,z)\in\Delta_U \\ 0, & \text{if } (x,z)\in\Delta_L, \end{cases} \tag{B.54}$$

where l_x and l_z are the side lenghts of the regtangular mesh element. Substituting these equations into B.50 and computing the integral, we obtain the following matrix

$$\begin{aligned} \mathbf{T}_{rec} =& \frac{\hbar^2}{2}\int_\Delta \mathbf{B}^T(\mathbf{r})\mathbf{M}^{-1}\mathbf{B}(\mathbf{r})d\mathbf{r} \\ =& \frac{\hbar^2}{2}\begin{bmatrix} \frac{\Delta_U}{m_x^*l_x^2}+\frac{\Delta_L}{m_z^*l_z^2} & -\frac{\Delta_L}{m_z^*l_z^2} & -\frac{\Delta_U}{m_x^*l_x^2} & 0 \\ -\frac{\Delta_L}{m_z^*l_z^2} & \frac{\Delta_L}{m_x^*l_x^2}+\frac{\Delta_L}{m_z^*l_z^2} & 0 & -\frac{\Delta_L}{m_x^*l_x^2} \\ -\frac{\Delta_U}{m_x^*l_x^2} & 0 & \frac{\Delta_U}{m_x^*l_x^2}+\frac{\Delta_U}{m_z^*l_z^2} & -\frac{\Delta_U}{m_z^*l_z^2} \\ 0 & -\frac{\Delta_L}{m_x^*l_x^2} & -\frac{\Delta_U}{m_z^*l_z^2} & \frac{\Delta_L}{m_x^*l_x^2}+\frac{\Delta_U}{m_z^*l_z^2} \end{bmatrix} \\ =& \frac{\hbar^2}{2}\begin{bmatrix} \frac{l_z}{2m_x^*l_x}+\frac{l_x}{2m_z^*l_z} & -\frac{l_x}{2m_z^*l_z} & -\frac{l_z}{2m_x^*l_x} & 0 \\ -\frac{l_x}{2m_z^*l_z} & \frac{l_z}{2m_x^*l_x}+\frac{l_x}{2m_z^*l_z} & 0 & -\frac{l_z}{2m_x^*l_x} \\ -\frac{l_z}{2m_x^*l_x} & 0 & \frac{l_x}{2m_x^*l_x}+\frac{l_x}{2m_z^*l_z} & -\frac{l_x}{2m_z^*l_z} \\ 0 & -\frac{l_z}{2m_x^*l_x} & -\frac{l_x}{2m_z^*l_z} & \frac{l_z}{2m_x^*l_x}+\frac{l_x}{2m_z^*l_z} \end{bmatrix} \end{aligned} . \tag{B.55}$$

In the last equality we used the fact that $\Delta_L = \Delta_U = \frac{l_x l_z}{2}$. To get the full $\mathbf{T}$ matrix, we have to embed all the smaller $\mathbf{T}_{rec}$ matrices, corresponding to a rectangular mesh element, into the full $\mathbf{T}$ matrix of all mesh elements. In a similar fashion the 4×4 contributions

to the **D** and **V** matrices can be calculated with equations B.34 and B.33, yielding the following matrices

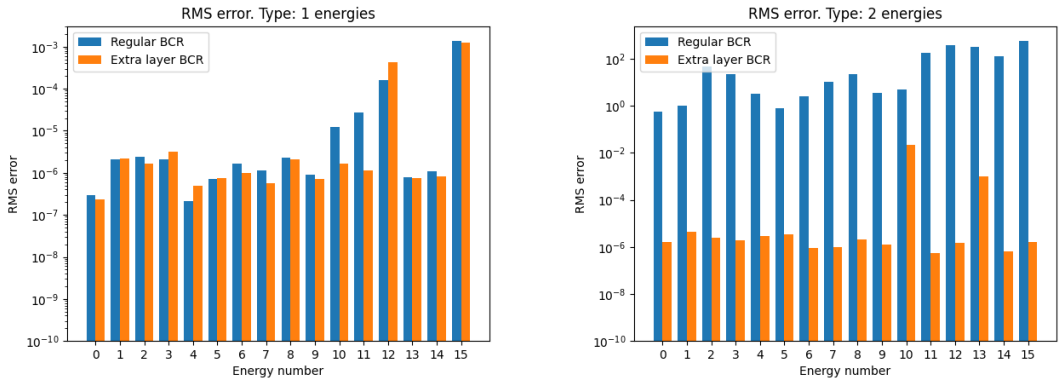
$$\mathbf{D}_{rec} = \begin{bmatrix} \frac{l_x l_y}{4} & 0 & 0 & 0 \\ 0 & \frac{l_x l_y}{4} & 0 & 0 \\ 0 & 0 & \frac{l_x l_y}{4} & 0 \\ 0 & 0 & 0 & \frac{l_x l_y}{4} \end{bmatrix} \tag{B.56}$$

$$\mathbf{V}_{rec} = \begin{bmatrix} V_1 \frac{l_x l_y}{4} & 0 & 0 & 0 \\ 0 & V_2 \frac{l_x l_y}{4} & 0 & 0 \\ 0 & 0 & V_3 \frac{l_x l_y}{4} & 0 \\ 0 & 0 & 0 & V_4 \frac{l_x l_y}{4} \end{bmatrix} \tag{B.57}$$

where V_1, ..., V_4 are the values of $V(\mathbf{r})$ at the element nodes. If the edge $1 \leftrightarrow 2$ of the mesh element is on the boundary of lead s, this element contributes the 1×2 matrix to $\mathbf{N}_{s,m}$

$$\begin{bmatrix} \chi_m^s(1)\frac{l_y}{2} & \chi_m^s(2)\frac{l_y}{2} \end{bmatrix} \tag{B.58}$$

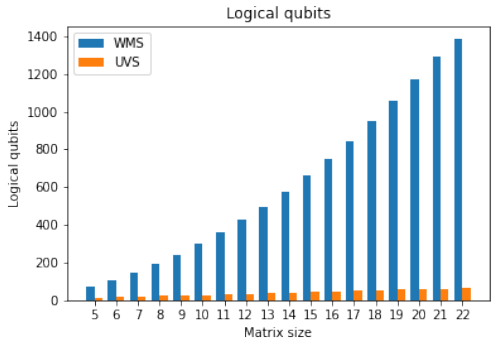
C Extra Results



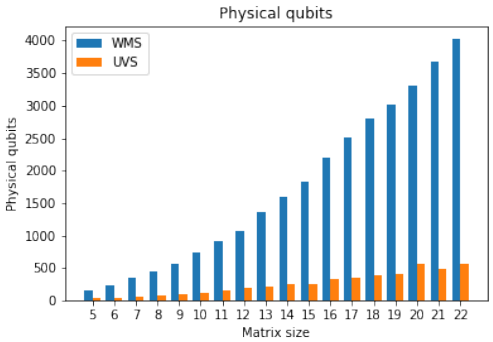
(a) Root mean square error for different Type: 1 energies.

(b) Root mean square error for different Type: 2 energies.

Figure C.1: RMS error. The error axis is logarithmic.



(a) Logical qubits needed to represent the BQM with the given specifications.



(b) Actual physical qubits used on the quantum annealer.

Figure C.2: Logical and physical qubits for the two inversion methods. The Dwave annealer ADVANTAGE_SYSTEM4.1 is used, which has 5627 available physical qubits.