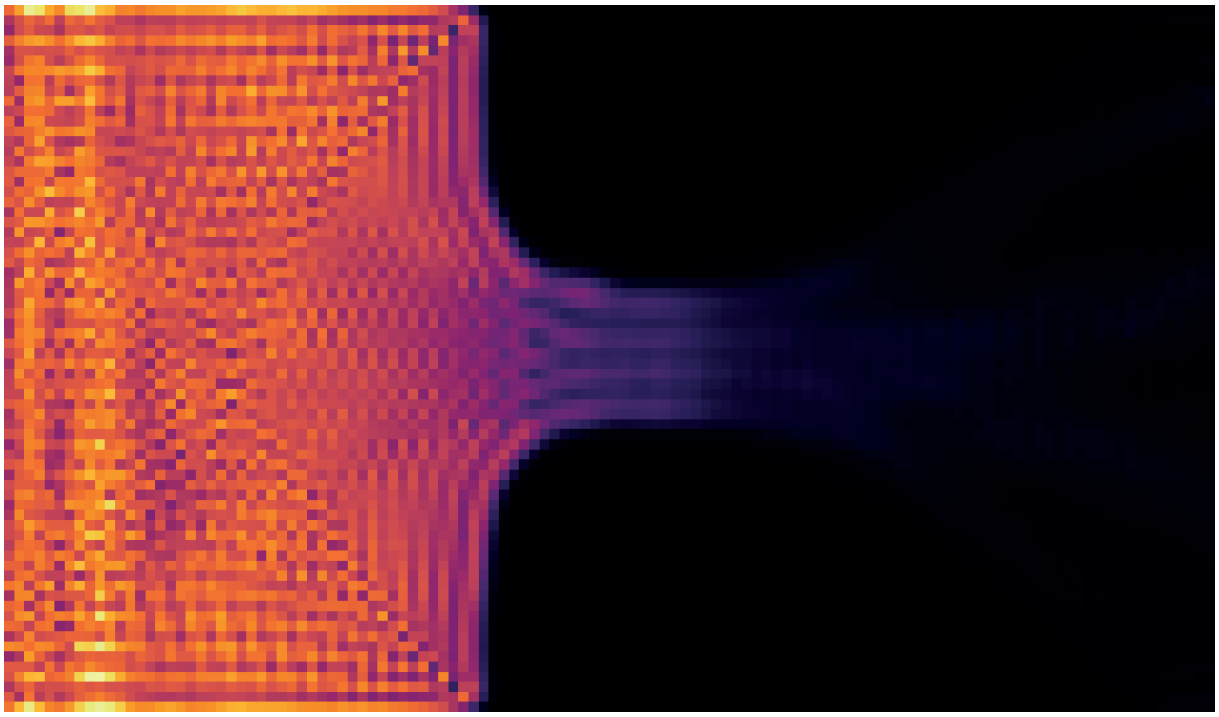
UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE



Master thesis

Autonomous tuning of gate-controlled quantum dots and Hall bars

Torbjørn Raasø Rasmussen

Supervisor: Ferdinand Kuemmeth

Co-Supervisor: Anasua Chatterjee

Submitted: January 7, 2022

Abstract

Autonomous tuning of gate-controlled quantum dots and Hall bars

by Torbjørn Raasø Rasmussen

As the scientific community moves towards realising quantum computers in semiconductor devices, these gate-controlled nanoscale devices themselves are becoming progressively more complex. The need for automated solutions for tuning and characterisation has become apparent, as the parameter space of very complex devices are quickly becoming too large to manually investigate. In this project we have focused on two different platforms for the investigation of automated exploration of quantum devices. For spin-qubits in gate-defined quantum dots we have developed an algorithm that automatically estimates the charge state boundaries in two- and three- dimensional gate-voltage space. For optimisation of quantum Hall physics within the constriction of a quantum point contact, we have implemented an algorithm that modulates the gate voltages applied to an array of pixel gates located above the constriction of the quantum point contact. Both algorithms have been tested on real devices in dilution refrigerators at subkelvin temperatures.

Acknowledgements

First I want to thank Bertram Brovang, not just for fabricating devices for this project, but for being my partner in crime through the last five years, I would not have made it through if not for your unwavering support. Secondly I want to thank the entirety of the spin-qubit group, both old and new. Working within this group has been fantastic and every single member have had a pivotal role in my development as a student. To Fabio, Federico and Heorhii the team i joined as a young bachelor student, you all took me under your wings and have shown me a whole new world in condensed matter physics. To the new team Joost, Fabrizio, Lazar it has been a joy working alongside you and every time I have needed help I always know who to come to. The entirety of Qdev can feel like a magical place to work, everything and everyone is always busy doing something that just might end up as the next amazing discovery. Oswin and Evert have been my very competent teachers regarding everything machine learning and algorithms, you have given me a great perspective on how those methods work and can be applied to such a variety of things. Thank you to all of the machine-learning-in-quantum-devices team, Bernd, Anton, Xavier and Charlie, it has been great taking part in the discussions we have had, and a special thanks to Bernd and Xavier for lending their students out for me to speak to. Michael Manfra has also been a part of the team and have fabricated the wafers upon which our devices are fabricated, without your materials there would be no project. Lastly I want to thank Ferdinand Kuemmeth and Anasua Chatterjee for letting me be a part of the spin-qubit group. Thank you Anasua for keeping me focused on the task at hand, always and especially when writing this thesis. Thank you Ferdinand, you have been a magnificent leader and it has been an amazing experience seeing you work at the setups, your way of dealing with the various problems one can encounter at such a setup is nothing short of awe-inspiring. The times you have dropped by the setup to see how things a going and perfectly structured a plan for investigating and dealing with an issue we are facing is inspiring.

Contents

List of Figures

Chapter 1

Introduction

The race for quantum supremacy is on and while many different possible platforms for implementation are suggested, the platform we have chosen is semiconducting devices [11, 27, 15]. Even within semiconducting devices there are several possibilities and some of the pros associated with semiconducting devices are the well developed fabrication lines, the computer industry already relies heavily on manufacturing semiconducting devices, so there is immense infrastructure already established. Other pros are associated with specific implementations of qubits, such as scalability and robustness against noise or decoherence [25, 30]. However there are also challenges, specifically when talking about scalability, scaling the number of qubits will always require more axes of manipulation leading to very complex device structures. The complex devices require complex manipulation, a task that for humans become almost infeasible when the dimensionality becomes too large. However the increased interest in combining machine learning and quantum devices seek to alleviate some of these problems. In ref. [23] the authors seek to automatically tune a gate defined double quantum dot device to a regime where double dot features appear. Simultaneously the same authors have made another algorithm for efficiently measuring bias triangles of double quantum dots[26], one can already grasp how not long in the future we will be able to cool down a device in the evening and come back next morning to a device that has been tuned and characterized without supervision. These cases were specialized for specific physical features, but they show that there is much to be gained in combining machine learning with quantum devices. The integration of machine learning and quantum devices can prove useful in two different avenues, as suggested above both in terms of automatising work previously done by scientists regarding preparation of devices, but also exploratory work. Where allowing algorithms to explore large parameter spaces can lead to new physics that are not yet understood.

In this thesis I will present work related to investigation and manipulation of quantum devices, specifically related to quantum dot arrays, quantum Hall effect and quantum point contact physics. The thesis will contain three major chapters, the first related to estimation of charge state boundaries in gate-voltage space in an array of quantum dots. The work presented in chapter 2 resulted in two manuscripts, [5] and [21], and a presentation by me at the 11th International Conference on Quantum Dots. In chapter three I will present work related to the optimisation of the physics of a quantum point contact in a device that simultaneously is used for quantum Hall effect experiments. Finally in chapter 4 I will briefly discuss the impact and future of these experiments.

Chapter 2

Automatic discovery of charge state transitions in quantum dots

2.1 Theory

2.1.1 Quantum Dots

Quantum dots are zero dimensional systems, where electrons are confined in all directions, such that they can only be on or off the dot. This enables some very interesting behaviour used widely to make spin-qubits, encoding bits of quantum information that can be used as the basis of the quantum computers of the future. One method of making quantum dots such as those used in this experiment is to first make a quantum wire, a one dimensional system, and then apply gate voltages along this wire to confine electrons in the last direction. Another typical method for making quantum dots would be to define them completely in a two-dimensional electron gas (2DEG) by applied gate voltages. Whichever method is chosen, their phenomenology is the same; quantum dots show nonlinear conductance and exhibit Coulomb blockade in certain regions of gate space where transport through them is not allowed. First, I will define what constitutes Coulomb blockade by looking at the Coulomb energy associated with electrons in close proximity. For N electrons on a small two-dimensional island of radius r the electrostatic energy is [16]

$$E_{elstat}(N) = \frac{e^2N^2}{2C} = \frac{e^2N^2}{16\epsilon\epsilon_r r}. \tag{2.1}$$

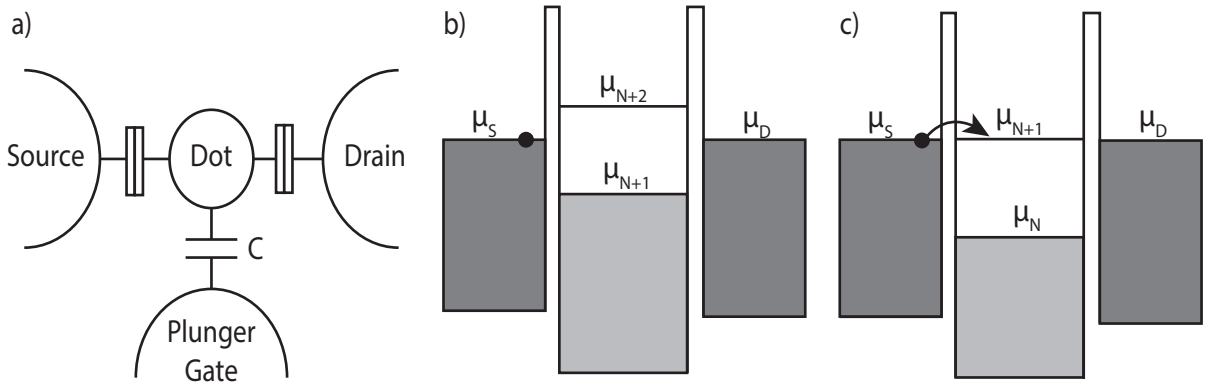


FIGURE 2.1: **Conductance of a Quantum dot.** Schematic of a quantum dot and two different cases of plunger gate voltage showing coulomb blockade and conductance resonance. **a)** Schematic setup. **b)** Coulomb blockaded transport because no energy level aligns with the source/drain electrochemical potential. **c)** Conductance resonance when the energy level of the dot is perfectly aligned with the energy level of source and drain.

If we assume that already N electrons are present on the island, the energy required to add another, known as the charging energy, is

$$E_c(N+1) = E_{elstat}(N+1) - E_{elstat}(N) = \frac{e^2}{C}(N+\frac{1}{2}) \approx \frac{e^2}{C}N. \tag{2.2}$$

Typically we want the general charging energy for adding an electron,

$$\Delta E_C = E_C(N+1) - E_C(N) = \frac{e^2}{C} = \frac{e^2}{8\epsilon\epsilon_0 r}. \tag{2.3}$$

This also gives a condition on the temperature of the environment. To be able to observe the physics of quantum dots the temperature must follow

$$k_B T \ll \Delta E_C = \frac{e^2}{C} \tag{2.4}$$

which is simultaneously a condition on the size of the island. Since the capacitance of a big island will be larger, making islands sufficiently small is crucial; typical sizes of quantum dots are in the order of tens of nm.

To understand what this means for the conductance of a quantum dot I will go through an example with a single dot tunnel coupled to two electron reservoirs we call drain and source, and a plunger gate that controls the electrochemical potential of the dot through capacitive coupling, schematically shown in Fig. 2.1a). μ_S and μ_D are typically related by an applied voltage bias $\mu_S - \mu_D = -|e|V_{SD}$ on one of the reservoirs. The electrochemical potential of the

quantum dot is dependent not just on the number of electrons on the dot, but also the plunger gate voltage applied. The energy levels of the dot as a function of small deviations in plunger gate voltages around a specific plunger gate voltage V_{pg}^0 are [16]

$$E_N(V_{pg}) = E_N(V_{pg}^0) - |e|N\alpha_{pg}\Delta V_{pg}. \tag{2.5}$$

Here $\Delta V_{pg} = V_{pg} - V_{pg}^0$ is the small deviation in plunger gate voltage, and α_{pg} is the lever-arm of the plunger gate when acting on the dot. With this we can now calculate the electrochemical potential for adding the N'th electron to the dot

$$\mu_N(V_{pg}) = E_N(V_{pg}) - E_{N-1}(V_{pg}). \tag{2.6}$$

Combining these two equations gives an expression for the electrochemical potential of adding the Nth electron, that is independent of N

$$\mu_N(V_{pg}) = \mu_N(V_{pg}^0) - |e|\alpha_{pg}\Delta V_{pg}. \tag{2.7}$$

Now in Fig. 2.1b) and c) the system is shown in the situations of Coulomb blockade and conductance resonance respectively. When the system is in Coulomb blockade the dot has $N+1$ electrons on it, and the energy required to add the $N+2$th electron is greater than the energy gained from removing an electron from the source, $\mu_{N+2} > \mu_S, \mu_D$, thus the transport through the dot is blocked and no current will flow. On conductance resonance, the electrochemical potentials of the source and drain contacts are aligned with the electrochemical potential of adding the $N+1$th electron to the dot, and there can flow a single electron through the level at a time. Additionally if there is a potential difference between the source and drain contacts, a window of $|e|V_{SD}$ is available for transport. If the quantum dot has an energy level, μ, within this range, current can flow between the two reservoirs.[16]

2.1.2 Constant Interaction Model

This can be extended for multiple quantum dots, given a system like shown in figure 2.2, where we for now ignore cross capacitances from one plunger gate to the dot it is not directly coupled

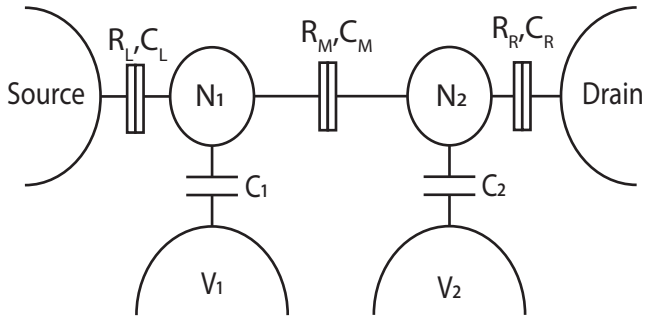


FIGURE 2.2: **Double Quantum Dot.** A system of two quantum dots with their own plunger gates, and connected through a tunneling barrier characterized by a resistor and a capacitor [29].

to. The total energy of the system is given by [29]

$$U(N_1, N_2) = \frac{1}{2}N_1^2 E_{C_1} + \frac{1}{2}N_2^2 E_{C_2} + N_1 N_2 E_{C_M} + f(V_1, V_2), \tag{2.8}$$

with

$$\begin{aligned} f(V_1, V_2) =& \frac{1}{-|e|}\left(C_1 V_1 (N_1 E_{C_1} + N_2 E_{C_M}) + C_2 V_2 (N_2 E_{C_2} + N_1 E_{C_M})\right) + \\ & \frac{1}{e2}\left(\frac{1}{2}C_1^2 V_1^2 E_{C_1} + \frac{1}{2}C_2^2 V_2^2 E_{C_2} + C_1 V_1 C_2 V_2 E_{C_M}\right) \end{aligned} \tag{2.9}$$

where $E_{C_{1(2)}}$ is the charging energy of a single dot, and E_{C_M} is the electrostatic coupling energy which is the energy change of one dot when an electron is added to the other. In terms of capacitances these energies are

$$E_{C_1} = \frac{e^2}{C_1}\left(\frac{1}{1 - \frac{C_M^2}{C_1 C_2}}\right); E_{C_2} = \frac{e^2}{C_2}\left(\frac{1}{1 - \frac{C_M^2}{C_1 C_2}}\right); E_{C_M} = \frac{e^2}{C_M}\left(\frac{1}{\frac{C_1 C_2}{C_M} - 1}\right) \tag{2.10}$$

This system of equations can be extended to any number of dots, and we have used it as the basis of a simulation with 4 dots that the algorithm shown in this chapter has been tested on. I will not discuss the simulation further here but instead reference the paper it was used for [21].

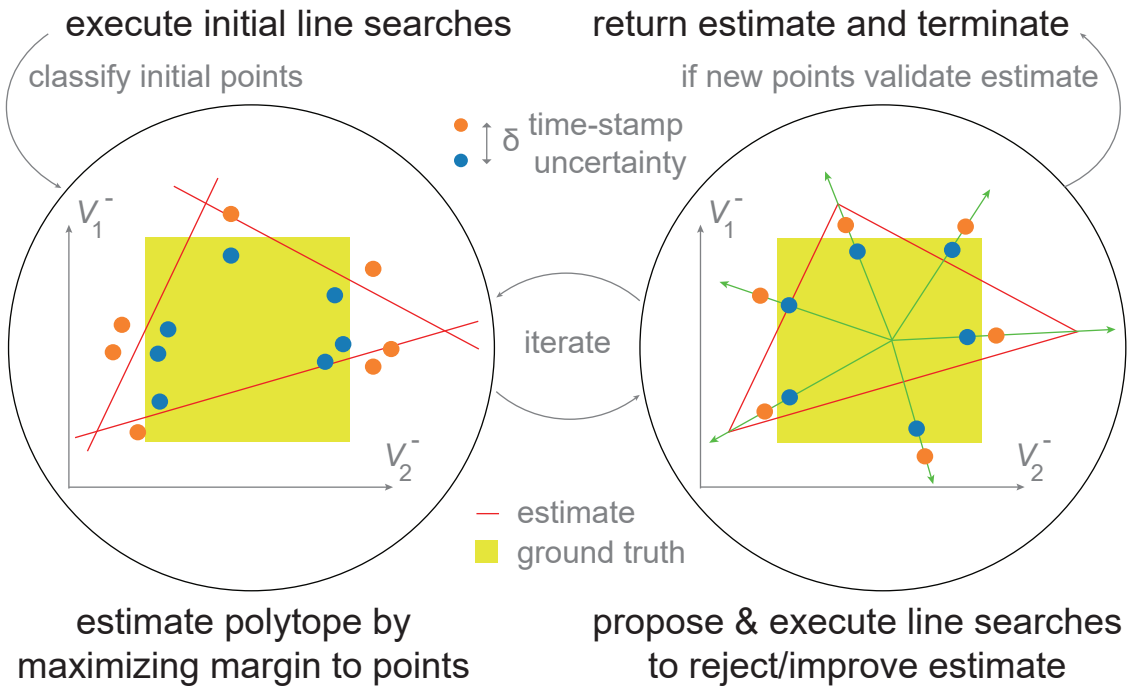


FIGURE 2.3: **Algorithm for Estimation of Convex Polytopes.** Algorithm consisting of two tasks. Estimate (red lines) is done on the set of inner and outer points generated by line searches (blue and orange points). New points are measured by performing line searches (green lines) through midpoints and vertices of estimates facets. Algorithm terminates if two succeeding estimates agree to sufficient degree. [5]

2.2 Algorithm

Our algorithm for the estimation of charge state transitions in an array of quantum dots is built on the assumption that the individual charge states are convex polytopes in gate voltage space. This assumption allows us to define points as being outside or inside the charge state based solely on line searches starting within the charge state. If you perform a line search from a point inside, at some point you will encounter the boundary to another charge state, and you will know with certainty that the initial charge state will not reappear anywhere beyond that boundary. The algorithm is based on active learning to iteratively propose new search directions for improving the estimate of the polytope [21]. For this we have built an algorithm that develops on the principles of large margin classifiers such as the support vector machine. Large margin classifiers are tasked with separating labeled points of two categories with a decision boundary that has the largest margin to the points. In other words, it seeks to find the hyperplane separating the two point clouds that is equidistant from the closest points [3]. In Fig. 2.3 an overview of the algorithm is given. I will now explain in a bit more detail every step

of the algorithm. Given a starting point known to be inside the unknown polytope, the first step is to perform line seaches in randomly chosen directions to generate a set of starting points. Next, one cycle of what I will refer to as estimation is performed, encapsulated in the left circle of Fig. 2.3. One cycle of estimation starts with a convex hull on the inner set of points, which generates a starting set of facets. These facets will be optimised by the large margin classifier to maximally separate the inner and outer points associated with each facet. Simultaneously, all the facets will be regularized in a way that favors setting multiple facets to zero, meaning that they will no longer be a part of the optimisation and not be assigned any new points, this is done to prune extra untrue facets. Now after estimation we are left with a set of facets that is the closest current estimate to the unknown polytope. The measurement part of the algorithm comes next and is encapsulated by the right circle in Fig. 2.3. This consists of choosing search directions and then performing the line searches themselves. The directions chosen will always be measuring through the vertices of two intersecting facets and the midpoints of those facets; this process will then return a new set of points. If the new points are consistent with the current best estimate, the algorithm will terminate. However if they do not agree with the current best estimate, the estimation process repeats.

2.3 Experimental Setups

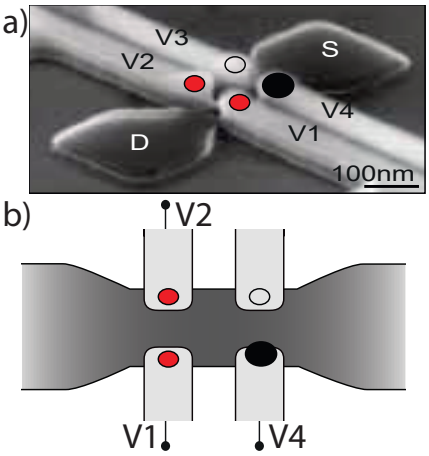


FIGURE 2.4: **2x2 Quantum Dot Device.** Foundry fabricated array of quantum dots in silicon [1]. **a)** SEM with annotated dots, gates and reservoirs. White dot is kept in Coulomb blockade (unused), red dots are active and black dot is used as a sensor dot. **b)** Schematic drawing of device.

We have carried out the experiment on the device shown in Fig. 2.4, where it is shown both in a scanning electron microscope (SEM) image in a) and schematically in b). The device has been entirely foundry fabricated and consists of an undoped silicon channel with four overlapping metallic polysilicon gates [1]. The quantum dots are denoted by circles and the black dot is used as a sensor dot. For the algorithm in two dimensions, using two quantum dots, the red dots are in use and the white dot under gate G_3 is kept in Coulomb blockade. Having a sensor dot in this experiment constitutes having a dot kept at a constant electrochemical potential equal to the electrochemical potential of the nearby reservoir of electrons, such that it constantly exchanges electrons with the lead. This process results in a non-zero signal as measured by RF reflectometry, which is our method of measurement (as detailed in [1] and [28]). It involves compensating the sensor dot for the cross capacitances it feels from the other metal gates, which allows us to measure raster scans such as the one shown in Fig. 2.5a). Here, the compensated region is (11). Whenever an electron moves within the array, the potential felt by the sensor dot is changed and it will therefore fall off the Coulomb peak, resulting in a large change in the demodulated voltage. This almost "digital" signal difference, is what is used to trigger the line searches to return a timestamp, such as shown in Fig. 2.5b). Here the green lines are the line search directions and red points are returned trigger points. This gives accurate information of the boundary of this (11) state to the other charge states while only measuring and saving a fraction of the data used in a raster scan.

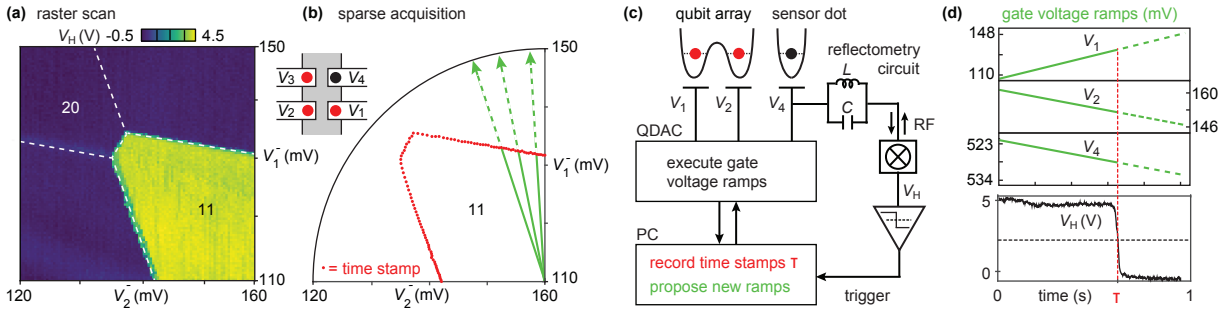


FIGURE 2.5: **Experimental Line Search Method.** The analog line search method is based reading the signal of a compensated sensor dot continually while ramping, triggering when a certain voltage drop in the signal is seen, and then only returning the timestamp for the trigger to the computer. Digitally the timestamp in combination with the ramp parameters are then used to reconstruct the gate values applied at the time of triggering. **a)** Conventional raster scan of the region containing the interdot transition (20)-(11), white lines added manually. **b)** Sparse acquisition with triggered line searches, essentially obtaining the same knowledge of boundaries. **c)** Schematic of the hardware setup with reflectometry circuit on 1 gate for the sensor dot. **d)** Digital reconstruction of voltages at time of trigger. [5]

2.4 Results

We have applied the algorithm to a 2x2 array of quantum dots in silicon [1], with either 2 or 3 dots and a sensor dot turned on at a time, where the unused dots are kept in Coulomb blockade by their associated gates. The results are shown in Fig. 2.6, in a) a conventional raster scan is shown in a blue/green/yellow color scale, overlaid on this is the result from the estimation algorithm. The red lines are the estimation of individual facets at termination. The overlay demonstrates the good fit with the dataset from a conventional raster scan. In total the algorithm run-time including measurement of 56 line searches, is less than a minute, in comparison to the conventional raster scan which has a measurement time upwards of 15 minutes. Such an upgrade of measurement time becomes even more significant when moving to higher dimensions where raster scans scale poorly. On top of the much better measurement time, the algorithm finds two interdot transitions that are hardly visible in the raster scan, but of great importance to qubit research, since these are where electron wavefunction overlaps within the array take place. In b) the result of the algorithm applied to the same device with all three dots filled by an electron, ie the (111) charge state, is shown. Validation of the triple dot result is more difficult since performing a raster scan with sufficient resolution would be very time consuming. In addition, the large amount of time taken means that the sensor would not be stable over this time period due to environmental drift. However overlaying cuts of the estimated 3 dimensional polytope on 2 dimensional raster scans of the same cuts shows good agreement. Interestingly, this algorithm estimation found 10 facets where one might expect

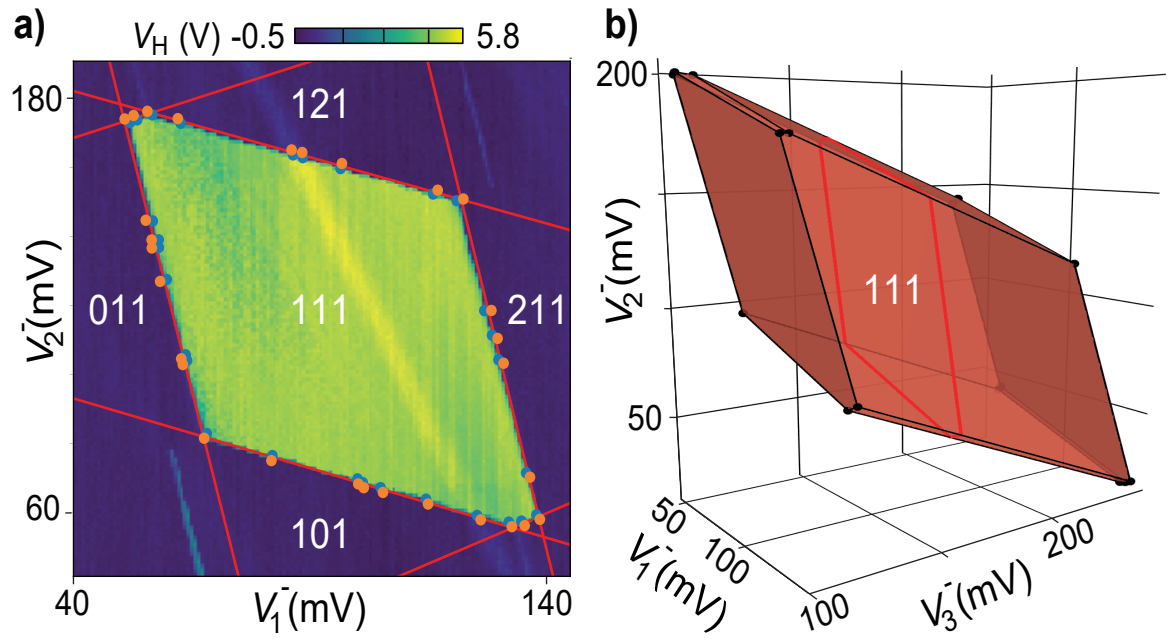


FIGURE 2.6: **Estimation of Charge State Transitions.** Results from running the algorithm on the array of quantum dots shown in Fig. 2.4. **a)** Two-dimensional results shown as red lines from measured blue and orange inner points, overlaid on top of conventional raster scan confirming the validity of the estimate generated by the algorithm. **b)** Three dimensional polytope estimated by running the algorithm with all three non-sensor dots in use. The marked cut of the polytope is consistent with a). [21]

there to be 12, involving 3 pairs of dot-to-lead transitions and 3 pairs of interdot transitions, however simulations of the system show that for specific arrangement of capacitances in the system, different numbers of facets can be achieved. The red lines cutting the polytope at $V_3^- = 150mV$ is a cut equivalent to what is shown in a).

Chapter 3

Optimization of Quantum Point Contacts

3.1 Motivation and Device

The main focus of this experiment is investigating how optimisation can be used to improve the physical features of quantum devices, in the particular experiment we are focused on quantum point physics, for its use in making interferometers for creating non-Abelian quasiparticles [24, 2, 17]. We have fabricated devices like shown in Fig. 3.1 with a 3x3 array of pixels deposited between sets of outer gates making a constriction in the 2DEG layer. Thus they can be used to fine-tune the potential in the constriction region and improve hopefully visibility of fractional quantum Hall states. The device has been fabricated on a GaAs wafer with a 190nm deep 2DEG that is highly doped for a mobility of $24.6 \cdot 10^6 \frac{\text{cm}^2}{\text{Vs}}$. The gates are fabricated with lithography for optimal in plane control, and made of gold. Further information on device fabrication can be found in the Master thesis by Bertram Brovang [4].

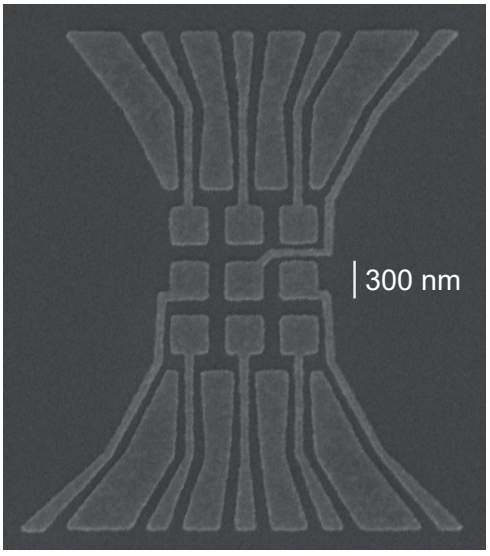


FIGURE 3.1: **SEM of Device.** Scanning electron micrograph of our gate pattern to form a tunable quantum point contact in the 2DEG located below the gate layer. In a completed device, ohmic contacts located on the left and right of this gate pattern allow the measurement of current flowing horizontally below the gate pattern (see also Fig. 3.3 and 3.11). Importantly, the 3x3 array of independent gate electrodes allow tuning of the potential landscape experienced by the 2DEG by application of independent gate voltages.

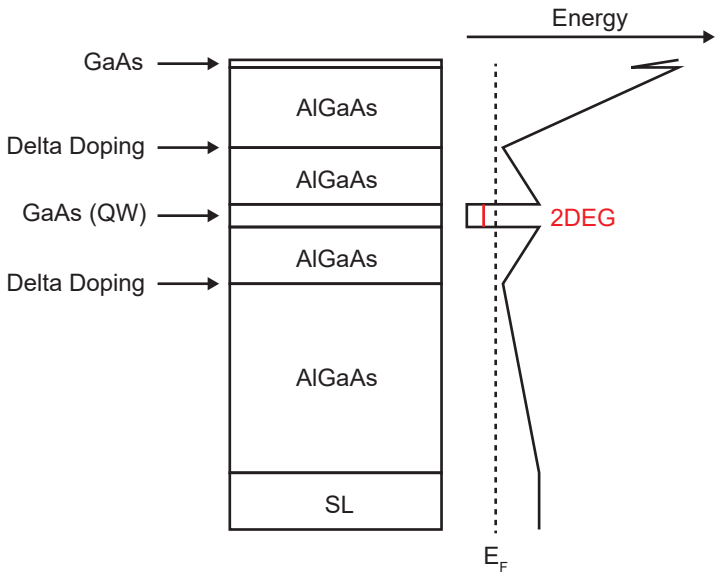


FIGURE 3.2: **Two Dimensional Electron Gas.** Formation of a two-dimensional electron gas (2DEG) in a GaAs quantum well (QW) in a AlGaAs/GaAs heterostructure. The vertical axis is the growth direction of the crystal, with a thin GaAs layer forming the surface of the crystal. Bandgap engineering and appropriate doping result in a Fermi level just above the first quantized subband of the quantum well (red line), thereby effectively restricting the spatial degrees of freedom of the electron gas to the two dimensions perpendicular to the growth direction.

3.2 Theory

3.2.1 Two Dimensional Electron Gas

The Two Dimensional Electron Gas, or 2DEG for short, is one of the most useful invented structures, being building blocks of many electronic devices in the form of Metal Oxide Semiconductor Field Effect Transistors (MOSFET). The MOSFET is as the name suggests made up of a semiconducting material, overgrown with an oxide and a metal gate, the metal gate can be applied a voltage and its electrical field will modulate the charge concentration of the semiconductor through the oxide. Thus one can realize a channel beneath the gate, where conduction can be turned off or on with the control of the metal gate. The 2DEG has been developed further for experimental use in the heterostructure, among other materials a hetero structure suitable for 2DEGs can be realised in $GaAs/AlGaAs$. This combination is especially suitable because their lattice constants are close to each other, and the band gap difference is suitable to make a quantum well in a layer of $GaAs$ between 2 layers of $AlGaAs$. In Fig. 3.2 such a hetero structure can be seen with associated conductance band energies, the lower conduction band energy in $GaAs$ allows trapping electrons in this spatial region, and the confinement potential

resembles a quantum well with allowed energy levels,

$$E_n = \frac{n^2\pi^2\hbar^2}{2m^*d^2}, \tag{3.1}$$

where m^* is the effective electron mass in $GaAs$ and d the depth of the $GaAs$ layer. For the electrons to be confined the difference in energy between $n = 1$ and $n = 2$ must be larger than the other energy scales involved with the system. [16]

The 2DEG in hetero-structures also allows for gating regions of the 2DEG by depositing metal gates on top of the structure, these can again modulate the carrier density in the region below them.

3.2.2 Quantum Point Contact

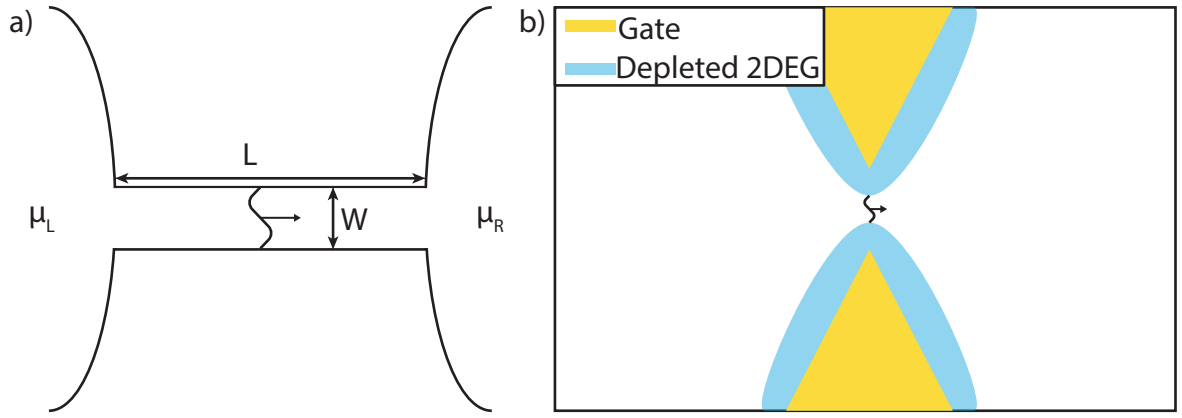


FIGURE 3.3: **Quantum Point Contacts.** Conductance quantization in quantum point contacts. (a) Theoretical model of a ballistic one-dimensional conductor connecting the left (L) and right (R) reservoir. Experimentally, the predicted conductance quantization in such a geometry is hard to observe, due to scattering within the wire. (b) Geometry of a gate-controlled quantum point contact, suitable for observing conductance quantization in high-mobility 2DEGs.

The quantum point contact is characterised by the experimental observation of conductance quantization, it was first observed in 1988 by van Wees and co-workers, and also independently by Wharam and co-workers. A quantum point contact is a narrow constriction allowing for ballistic transport between two connected electron reservoirs, this is typically realised in 2DEG systems with the constriction defined electrostatically by metallic gates, see Fig. 3.3. There is a set of criteria for observing the quantized conductance, as follows: the mean free path of electrons must be much bigger than the width and length of the constriction, $l_m \gg W, L$, the Fermi-wavelength must be comparable to the width, $\lambda_f \approx W$, and lastly the energy splitting of transverse modes in the constriction must be much greater than the thermal energy, $\Delta E_n \gg k_b T$.

Here I will go through the theoretical background for conductance quantization in quantum point contacts. The quantum problem in the wire is separable into the transverse modes and the modes along the wire. Combined, their wave functions are:

$$\psi_{n\mathbf{k}}(\mathbf{r}) = \chi_n(y,z) \cdot \frac{e^{ik_x x}}{\sqrt{L}} \tag{3.2}$$

Here, L is the normalisation length and χ_n the transverse modes. Assuming a parabolic energy dispersion along the wire,

$$E_n(k_x) = E_n + \frac{\hbar^2 k_x^2}{2m^*} \tag{3.3}$$

E_n is the contribution due to quantization of modes normal to the propagation direction. k_x accounts for both left and right movers depending on the sign, left movers will come from the right reservoir with electrochemical potential μ_R and right movers will have electrochemical potential μ_L. Now we want to figure out what the differential current of a single mode contributes. We start with the current density,

$$d\mathbf{j}_{nk_x}(\mathbf{r}) = -\frac{|e|\hbar}{2im^*}\left(\psi^*_{nk_x}(\mathbf{r})\nabla\psi_{nk_x}(\mathbf{r}) - \psi_{nk_x}(\mathbf{r})\nabla\psi^*_{nk_x}(\mathbf{r})\right). \tag{3.4}$$

Inserting the wave function, eq. 3.2, gives

$$d\mathbf{j}_{nk_x}(\mathbf{r}) = -\frac{|e|}{L}|\chi_n(y,z)|^2\frac{\hbar k_x}{m^*}\mathbf{e}_x, \tag{3.5}$$

where $\mathbf{e}_x$ is the unit vector in the wire direction. Inserting $dk_x = 2\pi/L$ leads to

$$d\mathbf{j}_{nk_x}(\mathbf{r}) = -\mathbf{e}_x\frac{|e|}{2\pi}|\chi_n(y,z)|^2\frac{\hbar k_x}{m^*}dk_x \tag{3.6}$$

Treating this as the equivalent of $\mathbf{j} = \rho\mathbf{v}$ in electrodynamics, we realize that the charge density is $\rho = -|e|dk_x|\chi_n(y,z)|^2/2\pi$, and the expectation value of the velocity in the x-direction is

$$\mathbf{v}_n(k_x) = \mathbf{e}_x\frac{\hbar k_x}{m^*} = \mathbf{e}_x\langle nk_x|\frac{\partial H}{\partial p_x}|nk_x\rangle = \mathbf{e}_x\frac{1}{\hbar}\frac{\partial E_n(k_x)}{\partial k_x}. \tag{3.7}$$

With this expression for the velocity and introducing the spin degeneracy g_s the current density becomes

$$d\mathbf{j}_{nk_x}(\mathbf{r}) = -\mathbf{e}_x g_s\frac{|e|}{h}|\chi_n(y,z)|^2\frac{\partial E_n(k_x)}{\partial k_x}dk_x. \tag{3.8}$$

Then substitution of $dk_k = dE\frac{\partial k_x}{\partial E_n(k_x)}$ and letting the energy terms associated with the density of states and the velocity cancel out, we are left with

$$d\mathbf{j}_{nk_x}(\mathbf{r}) = \mp \mathbf{e}_x g_s \frac{|e|}{h} |\chi_n(y,z)|^2 dE \tag{3.9}$$

with $-$ for right movers, and $+$ for left movers. Now getting the differential current requires integration over the cross-section of the wire, but since the transverse modes are normalized, we get

$$d\mathbf{I}_n(E) = \mp \mathbf{e}_x g_s \frac{|e|}{h} dE. \tag{3.10}$$

Now the total current through the wire when the reservoirs are not in thermodynamic equilibrium is

$$I_{tot} = g_s \frac{|e|}{h} \sum_n \int_{E_n}^{\infty} dE[f_l(E) - f_R(E)] \tag{3.11}$$

with f being the Fermi-Dirac distribution function $f_i = \frac{1}{\exp\left(\frac{E-\mu_i}{k_B T}\right)+1}$. Assuming a small voltage difference applied to the reservoirs is much smaller energetically than the thermal energy, we can expand

$$f_L(E) - f_R(E) = \frac{\partial f_L(E)}{\partial \mu_L}(\mu_L - \mu_R) = -\frac{\partial f_L(E)}{\partial E}|e|V_{SD}. \tag{3.12}$$

Inserting this into eq. 3.11 and integrating over the energy gives

$$I_{tot} = g_s \frac{e^2}{h} \sum_n f_L(E_n) V_S D. \tag{3.13}$$

Thus the conductance is

$$G = \frac{I_{tot}}{V_{SD}} = g_s \frac{e^2}{h} \sum_n f_L(E_n). \tag{3.14}$$

Now in the limit of zero temperature where f_L is a perfect step function, we arrive at the perfectly quantized conductance,

$$G = g_s \frac{e^2}{h} N, \tag{3.15}$$

where N is the number of occupied modes.

3.2.2.1 Saddle Point Potential

A potential model for the quantum point contact has been made in the saddle point potential, it assumes a potential of the form

$$V(x,y,z) = -\frac{1}{2}m^*\omega_x^2 x^2 + \frac{1}{2}m^*\omega_y^2 y^2 + V(z) \tag{3.16}$$

The $V(z)$ potential is governed by the 2DEG and its quantized states with energy E_z is assumed to have much greater separation of levels than any other energy scale present. Given the harmonic oscillator solutions in the y-direction and the separability of the electron motion, the equation of motion in the x-direction becomes

$$\left(-\frac{\hbar^2}{2m^*}\partial_x^2 - \frac{1}{2}m^*\omega_x^2 x^2\right)\xi(x) = E_x\xi(x) \tag{3.17}$$

By introducing the normalized energy scale $\epsilon = 2E_x/\hbar\omega_x$ and the length scale $l_x^2 = \hbar/m^*\omega_x$, we can rewrite as

$$\left(l_x^2\partial_x^2 + \frac{x^2}{l_x^2} + \epsilon\right)\xi(x) = 0 \tag{3.18}$$

The solutions to the equation of motion is given as a linear combination of parabolic cylinder functions $D_\nu(x)$

$$\xi(x) = c_1 D_{-\frac{1}{2}i(\epsilon-i)}[(1+i)x/l_x] + c_2 D_{\frac{1}{2}i(\epsilon+i)}[(-1+i)x/l_x] \tag{3.19}$$

Choosing the coefficients, c_1 and c_2 such that for $x \gg 0$ there is only transmitting modes, and for $x \ll 0$ there is incoming and reflected modes, it can be shown that the transmission of mode m is given by [16]

$$T_m(E) = \frac{1}{1+e^{-2\pi\epsilon_m}} \tag{3.20}$$

with

$$\epsilon_m = \frac{E - \hbar\omega_y(m+\frac{1}{2}) - E_z}{\hbar\omega_x} \tag{3.21}$$

The shape and transmission is shown in Fig. 3.4 a and b respectively.

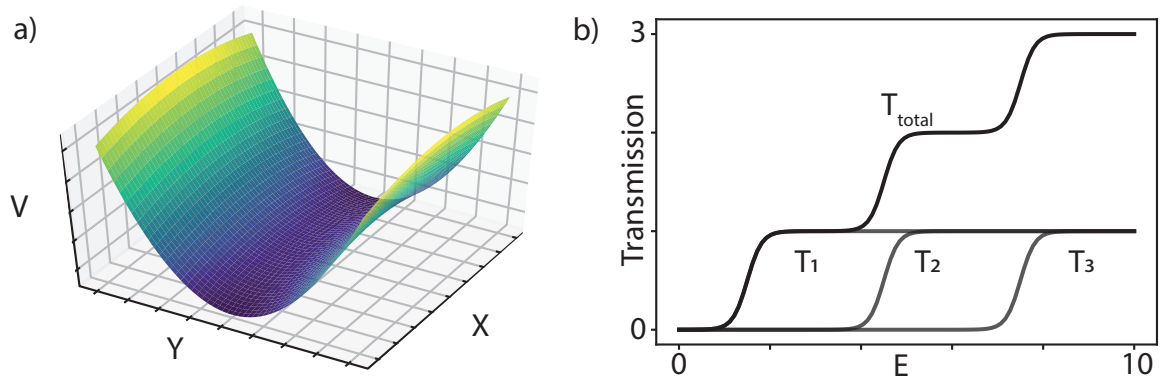


FIGURE 3.4: **Saddle Point Potential.** Saddle point potential to model a simple quantum point contact. **a)** The curvatures of the potential shape plotted here correspond to $\omega_y/\omega_x = 3.$, with current flow along x. **b)** Predicted transmission through the saddle point potential (see text).

3.2.3 Quantum Hall Effect

The Hall effect is a well known phenomenon occurring when a magnet field is applied perpendicular to the flow of current in a conducting material. The extension of Hall effect into the quantum regime for 2 dimensional materials, such as 2DEGs, is called quantum Hall effect and occurs for higher magnetic fields, the characteristics of the effect are easily recognized. The Hall resistance of a sample in the direction transverse to the current flow will increase and at higher magnet field values show plateaus, while the resistance along the current flow will oscillate and at higher magnet fields show peaks of high resistance and valleys of zero resistance. In Fig. 3.5 a plot from the Nobel Lecture by Klaus Von Klitzing is shown with these effects clearly visible, the discovery was made in 1980 and he was given the Nobel Prize in 1985. Arriving at an explanation for the high field behaviour requires first to solve Scrhödingers

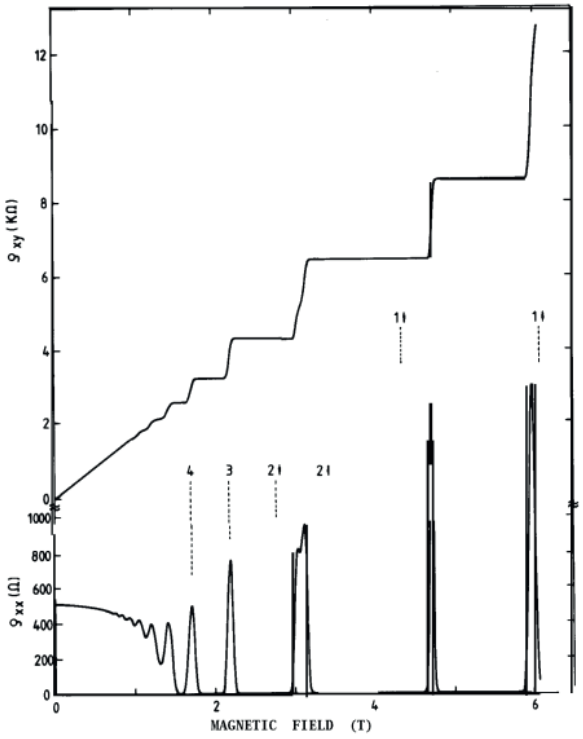


FIGURE 3.5: **Integer Quantum Hall Effect.** ρ_{xy} and ρ_{xx} as a function of magnetic field, as presented in the Nobel lecture by K. Klitzing who was awarded the Nobel price for its discovery in 1980 [20]. ρ_{xy} shows plateaus and ρ_{xx} is at zero resistivity for the same values of magnetic field.

equation for an electron in a magnetic field. The hamiltonian is

$$H = \frac{(\mathbf{p} + |e|\mathbf{A})^2}{2m^*} + V(z) \tag{3.22}$$

Here $V(z)$ is given by the confinement in the direction perpendicular to the 2D material. We are free to choose a vector potential giving the magnetic field $\mathbf{B} = (0, 0, B)$, and for simplicity we choose $\mathbf{A} = (-B_y, 0, 0)$. The hamiltonian can then be separated into

$$H_z = -\frac{\hbar^2}{2m^*}\frac{\partial^2}{\partial z^2} + V(z) \tag{3.23}$$

and

$$H_{xy} = \frac{(p_x - |e|B_z y)^2 + p_y^2}{2m^*} \tag{3.24}$$

The latter of which is independent of the confinement potential $V(z)$. Given that the energy level spacing of the confinement potential is large, only the lowest level will be occupied, with an educated guess on the wave-function we can solve the 2D problem

$$\psi(x, y) = e^{ik_x x}\eta(y) \tag{3.25}$$

giving the eigenvalue problem

$$\left(\frac{p_y^2}{2m^*} + \frac{1}{2}m^*\omega_c^2\left(y - \frac{\hbar k_x}{|e|B_z}\right)^2\right)\eta_{k_x}(y) = E\eta_{k_x}(y) \tag{3.26}$$

Where $\omega_c = \frac{|e|B}{m^*}$ is the cyclotron frequency. This equation is equivalent to a quantum mechanical harmonic oscillator with center coordinate $y_0 = \frac{\hbar k_x}{|e|B}$, and as a result we know the energy levels given by

$$E_n = \hbar\omega_c(n + \frac{1}{2}) \tag{3.27}$$

This leaves us with quantum states labeled with k_x and n, given that eq. 3.27 is independent of k_x, the states labeled by different k_x but the same n will be degenerate, this is called Landau-levels. The requirement arising from the center coordinate $y_0 = \hbar k_x/eB$ having to be within the width of the sample, and the density of k_x states in a sample of length, L being $L/2\pi$ leads to a requirement on the allowed values of k_X. Given a sample of length L, width W and area $A = WL$, the allowed values for k_x obey $0 \leq k_x L/2\pi \leq (eB/h)A$. Giving a total number of

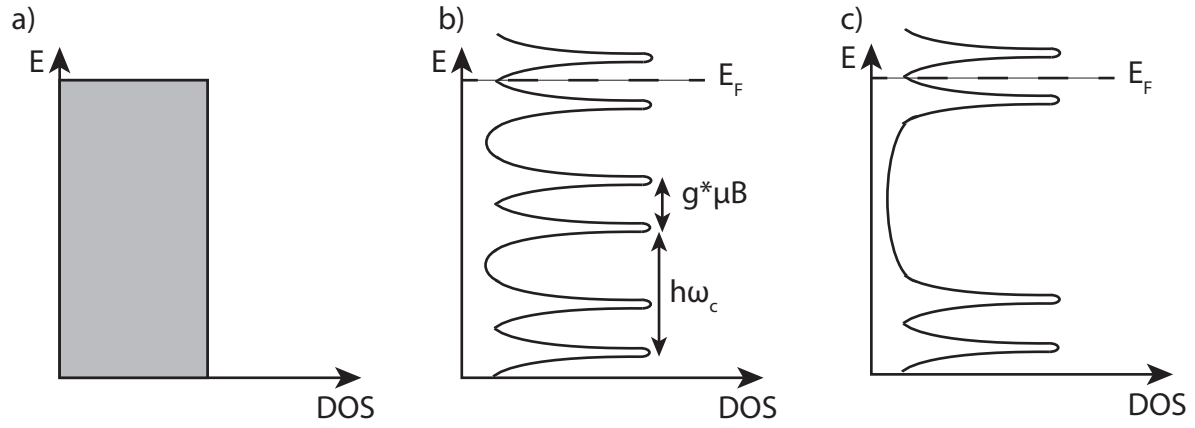


FIGURE 3.6: **DOS in a Magnetic Field** Density of States in the bulk of a 2DEG for increasing magnetic field strength. **a)** No magnetic field applied and just the 2DEG DOS. **b)** magnetic field is applied and landau levels appear, here also shown with Zeeman splitting. **c)** At even stronger field values the energy levels are pushed higher in energy and as a result higher levels will be depopulated.

allowed k_x states per unit area of

$$n_L = \frac{|e|B}{h} \tag{3.28}$$

Combined with the electron density of the sample, n_s we get the filling factor of the sample, $\nu = \frac{n_s}{n_L} = \frac{hn_s}{|e|B}$ at magnetic field B. Because the degeneracy of each Landau level is dependent on the magnetic field, as the magnetic field is increased, so is the degeneracy of the Landau-levels and higher energy states will be depopulated. The density of states in a sample will go through a transition from being continuous at 0 field to being discrete at higher field. In addition to this, there is Zeeman splitting of the electronic levels to take into account, so far we have neglected the spin of the electrons. The full energy levels are

$$E_n^{\pm} = \hbar\omega_c(n + \frac{1}{2}) \pm \frac{1}{2}g^*\mu_B B_z \tag{3.29}$$

This effect is shown in Fig. 3.6 for no magnetic field in a), a stronger magnetic field in b) and even stronger yet in c). The broadening of levels is caused by, among other things, scattering at both long and short range.

Fully understanding why the longitudinal resistance drops to zero while the transverse resistance plateaus require treating the edges of the sample [13]. I will now go through the theoretical understanding of edge states to fully account for integer quantum Hall effect. Starting

from eq. 3.26 we introduce a confinement potential in the y-direction giving

$$\left(\frac{p_y^2}{2m^*} + \frac{1}{2}m^*\omega_c^2\left(y - \frac{\hbar k_x}{|e|B_z}\right)^2 + V(y)\right)\eta_{k_x}(y) = E_n\eta_{nk_x}(y) \tag{3.30}$$

Treating $V(y)$ as a small perturbation gives $\langle V(y)\rangle = V(y_0(k_x)) = V(\hbar k_x/|e|B_z)$. We obtain the eigenenergies

$$E_n(k_x) = \hbar\omega_c(n + \frac{1}{2}) + V(\hbar k_x/|e|B_z). \tag{3.31}$$

The Landau level degeneracy is lifted at the edges by the potential, and when they intersect with the Fermi level they create 1 dimensional conductance channels, running in the x-directions. These channels carry current with group velocity

$$v_x = \frac{1}{\hbar}\frac{\partial V}{\partial k_x} = \left.\frac{\partial V(y)}{\partial y}\right|_{y=\hbar k_x/(eB)}\frac{1}{|e|B}. \tag{3.32}$$

Around integer filling factors, in the interior region of the sample, the electrons are localized and therefore do not couple the edge states at opposite edges running in opposite directions. This results in something analogous to the the modes of the quantum point contact, with no back scattering. In the quantum Hall effect the back scattering is almost completely suppressed because the modes moving in opposite directions are spatially separated [16]. At integer filling factors the longitudinal voltage drops to zero because contacts on the same side of the sample are connected by these edge channels that are perfect conductors.

3.2.4 Fractional Quantum Hall Effect

Furthermore the quantized conductance has been observed for filling factors that are not integer, this is what is known as fractional quantum Hall effect, owing its name to the fractional filling factors at which it appears. This effect is not yet fully understood, however the most commonly accepted theory is that of composite fermions, where electrons combine to particles with fractional charge [18, 19].

3.3 Optimization Algorithm

3.3.1 Loss Functions

Over the course of this experiment we have performed optimisations with a variety of loss functions and iterated these loss functions whenever we encountered problems that could be solved by accounting for them in the loss function. I will go through two of the loss functions we have used, those are also responsible for the two presented algorithm runs later. All the loss functions have in common some overall features, they share input and output and are all designed with staircases in mind. For the input we are using the measurement trace, a series of conductance measurements, this is converted to some output value, the loss which we are trying to minimize.

The first and simpler loss function has two main factors, one evaluating the "staircasiness" of the trace and another accounting for points with conductance equal to zero, in our case the staircasiness is to be minimised, so a lower staircasiness is given to a better staircase. The second factor is an example of a minor change implemented based on feedback from running the optimisation. For a measurement, M, with N total points:

$$S_1 = \frac{\sum_{i=1}^{N-1} \sqrt[3]{|M_{i+1} - M_i| + \epsilon_{noise}}}{1 + \frac{N(M>1e-5)}{N}} \tag{3.33}$$

In eq. 3.33 the numerator calculates the staircasiness and the denominator scales the staircasiness by the number of non-zero points such that measurements with more points above $g = 1e-5$ are favoured. Combined, this loss function seeks to minimise the staircasiness, while also maximising the number of points above 0 conductance to avoid closing off the conducting channel completely.

The second loss function I will go over is a modified version of the first where instead of looking at the entire measurement it looks only at points within a given interval. For this to work properly it has to normalise with respect to the number of points found within a given interval and the exact points measured at the borders of the interval. For a given upper and lower limit, μ_u and μ_l, if a measurement is completely outside we return 1. For measurements with points inside and outside the interval, we then find the outermost points within the interval, M_{μ_l} and M_{μ_u}, and their respective indices, I_{μ_l} and I_{μ_u}. From this we calculate the difference, $D = M_{\mu_u} - M_{\mu_l}$ and length of the interval $L = I_{\mu_u} - I_{\mu_l}$. Now I introduce the power variable,

p which serves a similar function as the cube root in the simpler loss function, experimentally we have chosen 0.2 for the value of p. All combined the loss function looks like this:

$$S_2 = \frac{\sum_{i=\mu_l}^{\mu_u - 1}(|M_{i+1} - M_i|/D)^p}{(L-1)^{1-p}} \tag{3.34}$$

Additionally we have introduced L_1 and L_2 regularisation, L_1 for feature selection and L_2 to combat overfitting, both of these have only been tested in combination with the Fourier modes that I will go over next. With X_i being the ith optimisation parameter the regularisation terms are [3]:

$$L_1 = \lambda_1 \sum_{i=0}^{N} |X_i| \ , \ L_2 = \lambda_2 \sum_{i=0}^{N} X_i^2 \tag{3.35}$$

All loss functions written for this project can be found within appendix C.

3.3.2 Fourier Modes

We have found the best results when including a Fourier transform, such that the optimisation is instead done on the Fourier parameters and then converted to gate voltages before being applied. This method involves a slight increase in computational time for measuring each trace, but the increase in performance by the algorithm makes up for it. The explanation for the performance increase from optimising Fourier modes, is likely that some of the parameters already resemble a saddle point potential. Thus the optimisation algorithm will have an easier time finding the overall shape of a saddle point potential, but it keeps the full degrees of freedom that would be lost if we instead chose to manually group sets of gates. The Fourier modes are obtained from the following method: The full real-valued 2D Fourier series is [9]

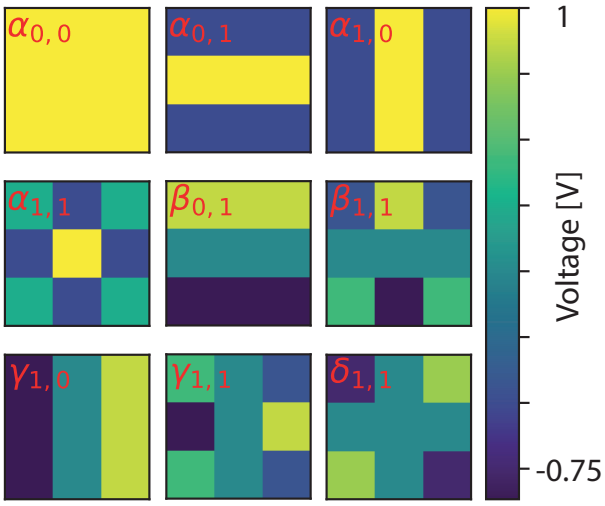


FIGURE 3.7: **Fourier Modes.** Individual Fourier modes for optimisation, each mode parameter is named in red, and its resulting voltages for parameter equal to 1 is shown in the 3x3 square corresponding to the pixels.

$$
\begin{aligned}
f(x,y) = &\sum_{n=0}^{\inf}\sum_{m=0}^{\inf} \alpha_{n,m} cos\left(\frac{2\pi n x}{\lambda_x}\right) cos\left(\frac{2\pi m y}{\lambda_y}\right) + \\
&\sum_{n=0}^{\inf}\sum_{m=0}^{\inf} \beta_{n,m} cos\left(\frac{2\pi n x}{\lambda_x}\right) sin\left(\frac{2\pi m y}{\lambda_y}\right) + \\
&\sum_{n=0}^{\inf}\sum_{m=0}^{\inf} \gamma_{n,m} sin\left(\frac{2\pi n x}{\lambda_x}\right) cos\left(\frac{2\pi m y}{\lambda_y}\right) + \\
&\sum_{n=0}^{\inf}\sum_{m=0}^{\inf} \delta_{n,m} sin\left(\frac{2\pi n x}{\lambda_x}\right) sin\left(\frac{2\pi m y}{\lambda_y}\right).
\end{aligned}
\tag{3.36}
$$

Since we are not working on an infinite lattice, n and m will only run up to 1, this means that a significant number of terms will drop out, and we will be left with only 9 terms, resulting in:

$$
\begin{aligned}
f(x,y) = &\alpha_{0,0} + \alpha_{0,1} cos\left(\frac{2\pi y}{\lambda}\right) + \alpha_{1,0} cos\left(\frac{2\pi x}{\lambda}\right) + \alpha_{1,1} cos\left(\frac{2\pi x}{\lambda}\right) cos\left(\frac{2\pi y}{\lambda}\right) + \\
&\beta_{0,1} sin\left(\frac{2\pi y}{\lambda}\right) + \beta_{1,1} cos\left(\frac{2\pi x}{\lambda}\right) sin\left(\frac{2\pi y}{\lambda}\right) + \\
&\gamma_{1,0} sin\left(\frac{2\pi x}{\lambda}\right) + \gamma_{1,1} cos\left(\frac{2\pi x}{\lambda}\right) cos\left(\frac{2\pi y}{\lambda}\right) + \\
&\delta_{1,1} sin\left(\frac{2\pi x}{\lambda}\right) sin\left(\frac{2\pi y}{\lambda}\right).
\end{aligned}
\tag{3.37}
$$

In Fig. 3.7 these Fourier modes are shown with all coefficients set to 1. Some modes immediately stick out as obvious choices for making a saddle point potential, modes like $\alpha_{0,1}$ and $\alpha_{1,0}$ combined can almost make it entirely on their own as they represent the transverse and longitudinal curvature. These Fourier modes were intended for use with the algorithm, but due to unforeseen difficulties with the device, we only used Fourier modes for optimisation on the simulated device.

3.3.3 Algorithm

We have implemented two optimisation algorithms for use on the quantum point contact devices. We started out by using a conjugate gradient descent algorithm, but quickly moved on to a gradient free approach from covariant matrix adaptation evolutionary strategies (CMA-ES). The gradient based approach is well developed and has been used on a wide range of different problems, both in physics and countless other topics. However for this optimisation

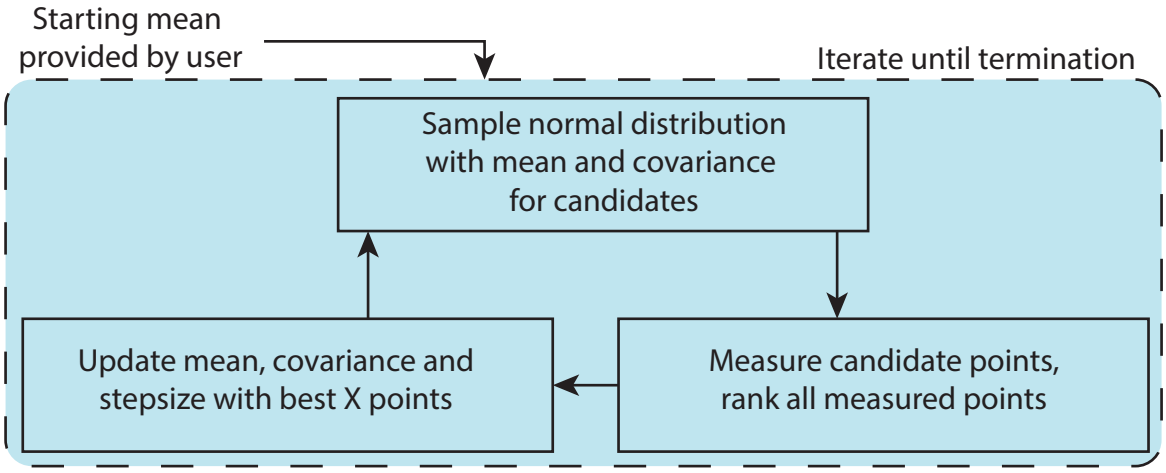


FIGURE 3.8: **CMA-ES Algorithm.** Flowchart of the CMA-ES algorithm, user provides the starting point for optimisation, then one iterations is as follows: sample the normal distribution with covariance matrix C. Next measure and evaluate the fitness of the sampled points and arrange them after best fitness together with all other measured points. Lastly update the mean and covariance matrix using the best encountered points. There are multiple available stopping criterions that will be checked after each iterations, these include time, stepsize, fitness goal, fitness stagnation and many more.

problem a different algorithm had more promising features, specifically the gradient calculation is in our case very inefficient and the gradient itself does not necessarily provide much information. Given our big search space in 9 dimensions, a numerical evaluation of the gradient is very costly. The loss landscape of staircasiness from a quantum point contact is often not monotonically increasing or decreasing along certain directions, but has both hills and valleys scattered across the landscape, a rugged landscape like this can be solved using gradient based approaches, but the combination of a rugged landscape and costly gradient calculation ultimately steered us away from this approach.

I will focus on describing the optimisation algorithm we have primarily used, CMA-ES. In Fig. 3.8 a flowchart of the CMA-ES algorithm is shown, the algorithm works by sampling points from a normal distribution with a given mean and covariance matrix and iteratively updating this mean and covariance matrix to move the search space in the direction of a global optimum for the parameters. Thus the optimisation is gradient free and is well suited for more rugged landscapes, two characteristics evaluated highly when choosing a loss function for this problem. [14] The code for interacting with the CMA-ES package can be found in appendix B.

3.3.4 Algorithm Test with Kwant Package

While fabricating devices we have run several test runs using the Kwant python package as a device simulation [12]. The Kwant package uses the tight binding model to simulate quantum

transport, and has been used for quantum Hall effect, Majorana states and more. We have built a basic simulation of the device utilising a simple calculation of gate potentials and add them directly to the energy of each tight binding lattice site. The simplistic simulation allows in conjunction with Qcodes to test out the algorithm and ensure that everything runs and is ready for the device. Kwant also includes features that allow for direct inspection of the scattering wave function which give insight into the performance of the algorithm.

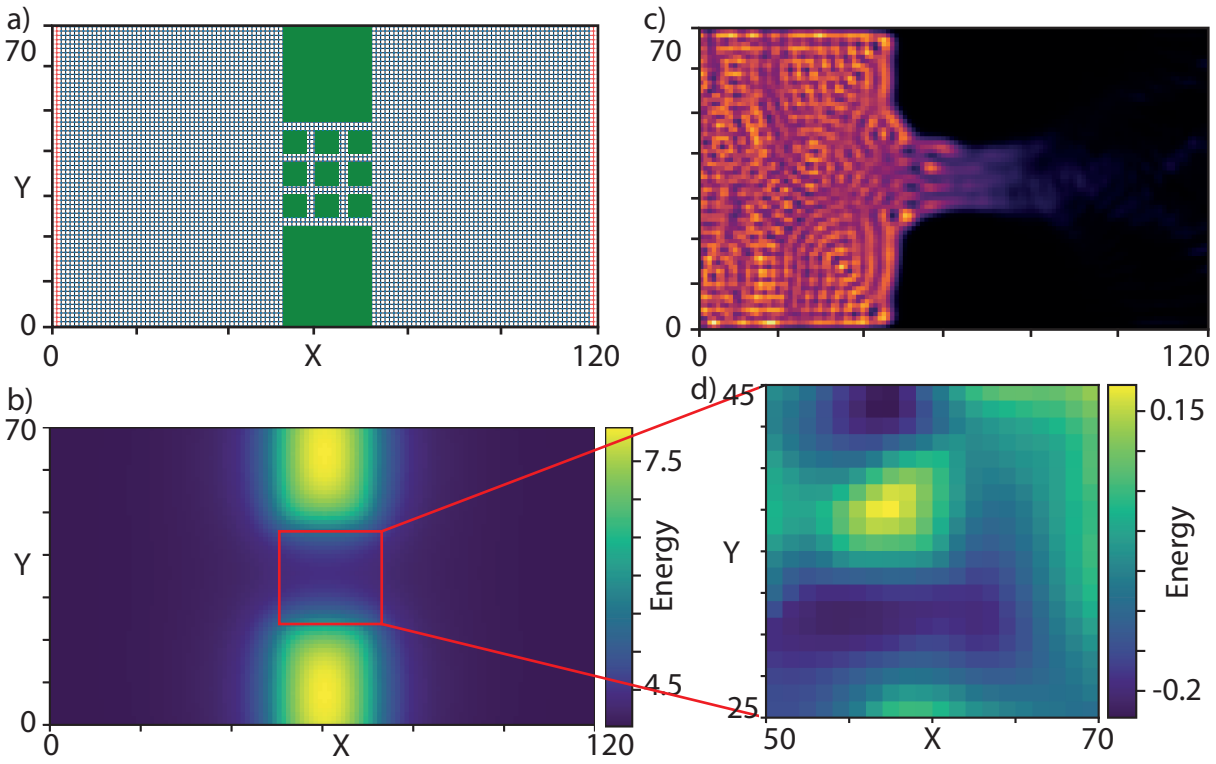


FIGURE 3.9: **Kwant Simulation.** The simulation is shown along with some of the features implemented with it. **a)** Simulation lattice, gates in green, sites in blue and leads in red. **b)** Simulated potential in energy units with strong outer gates. **c)** Scattering wave function from the left lead, with the potential from b) applied. **d)** Disorder implemented in simulation zoomed in to pixel region.

In Fig. 3.9a) the base of the simulation is shown, a lattice of size 70x120 and a set of rectangular gates with their potentials added directly to lattice site energies. The potentials are calculated from the gate dimensions following these formula:

$$g(u,v) = arctan2\left(u \cdot v, \frac{d \cdot \sqrt{u^2+v^2+D^2}}{2\pi}\right), \tag{3.38}$$

$$E(x,y,V) = -V \cdot \left(g(x-L,y-B) + g(x-L,T-y) + g(R-x,y-B) + g(R-x,T-y)\right). \tag{3.39}$$

Where D is the vertical distance to the gates from the lattice layer, L is the leftmost point, R is the rightmost point, B is the bottom and T is the top, x and y are the coordinates of the lattice sites, the energy contribution is calculated for and V is the voltage applied to that specific gate. [8]

All gate potentials are simply summed up to get the final energy for a lattice site. This method is not physically accurate, but it acts well enough for us to test out the algorithm without having to build an accurate physical simulation of electrostatics of gate potentials. In Fig. 3.9b) an example of such a calculated potential is shown, where the outer gates are at a strong negative value and the pixel gates are zero. There is a small disorder potential added as well, this is done by randomly generating values and add them to specific lattice sites evenly distributed in the lattice with a length scale equivalent to the size of the pixels, and then interpolating from these sites to the rest of the lattice sites. An example of just the disorder potential is shown in Fig. 3.9d) zoomed in to the region around the pixels.
The general approach to optimisation taken on the simulator was to use the outer gates for defining a quantum point contact, and then sweep the average voltage of the pixel gates to close off the channel. Optimisation is then done on the offset of the pixels, constrained by keeping the pixels to a set average. Lastly I will show an example of this algorithm in action on the simulated device. The method will be as described above, setting the outer gates to a set value, sweeping the pixels and optimising on their individual offset through Fourier modes.

In Fig. 3.10 the results from such a run are shown, a) shows the improvement from the non optimised case to the best encountered measurement, there is clearly better definition of steps at higher V_{QPC}. In b) and c) the resulting voltages and the optimised Fourier modes are shown, as seen in c) for this run a single mode is dominating. In d) and e) the scattering wave-function from the left lead is shown at two different values of V_{QPC}, looking closely at the pixel region one can actually see that in d) there are 4 visible modes, while in e) there are 5 in agreement with the conductance value. The code for simulating the device can be found in appendix A.

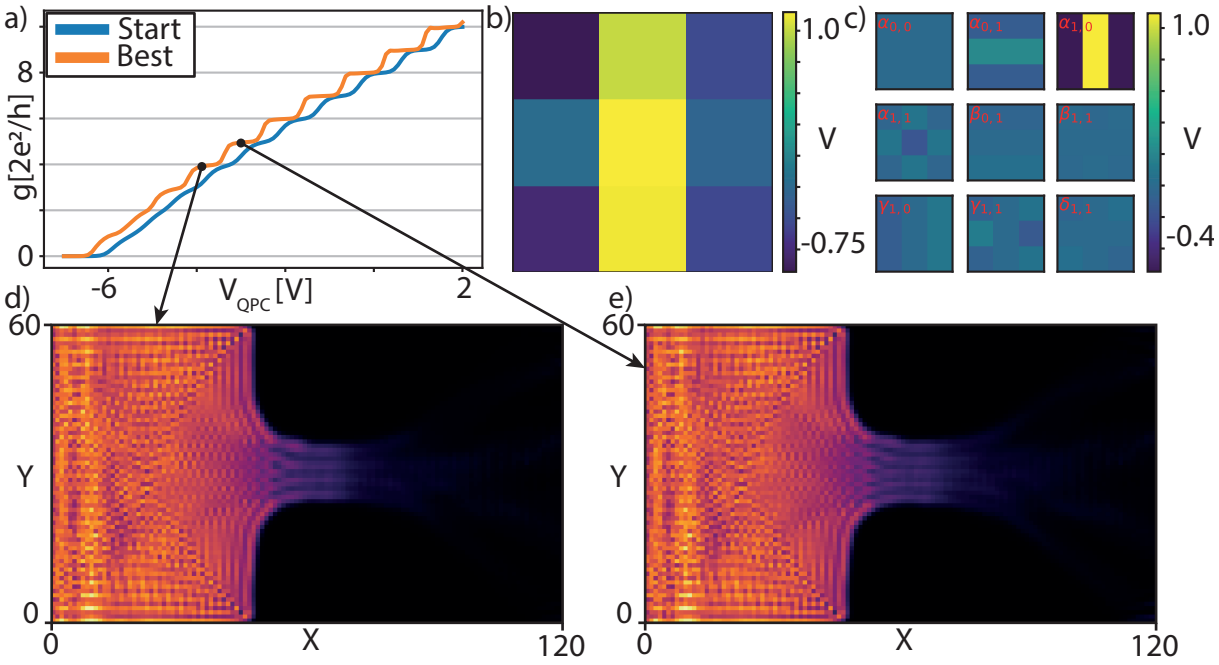


FIGURE 3.10: **Simulated Algorithm Run.** Running the algorithm on the simulated system in Kwant, we see obvious improvement, the resulting voltages are a bit puzzling, but this could be a result of punishing too heavily with L_1 regularisation. **a)** Unoptimised and best optimised measurement. **b)** Applied voltages to pixels. **c)** Fourier modes found for best optimised run. **d)** and **e)** Scattering wave function at different values of V_{QPC}, d) at -4 and e) at -3. Looking closely at the constriction region, one can see a different number of modes being let through, corresponding very to the conductance.

3.4 Experimental Setups

3.4.1 Device Schematic

The device is designed with two primary purposes in mind, it should be suited for quantum Hall measurements and QPC measurements, with the end goal of stabilizing fractional quantum Hall levels in the QPC constriction. Realizing these purposes requires ohmics for measuring on both sides of the gate fan outs, see Fig. 3.11 a), where the fan out of gates is shown in red. In Fig. 3.11 b) a zoomed in schematic of the gates are shown, the thought process behind this design is that the 8 outer most gates V_{O1}-V_{O8} should entirely deplete the 2DEG beneath them and create a narrow constriction between them. The pixel gates V_{P1}-V_{P9} are placed on top of this constriction and should help fine tune the potential in the constriction. There are two main factors the pixels should take care of, first the overall landscape of the potential, having this array of 3x3 pixels allows for great control of the potential, making the constriction wider, narrower, longer, deeper or more shallow. The second factor is disorder in the material, not all devices are created equally and if there is a very disordered potential in this region, optimal

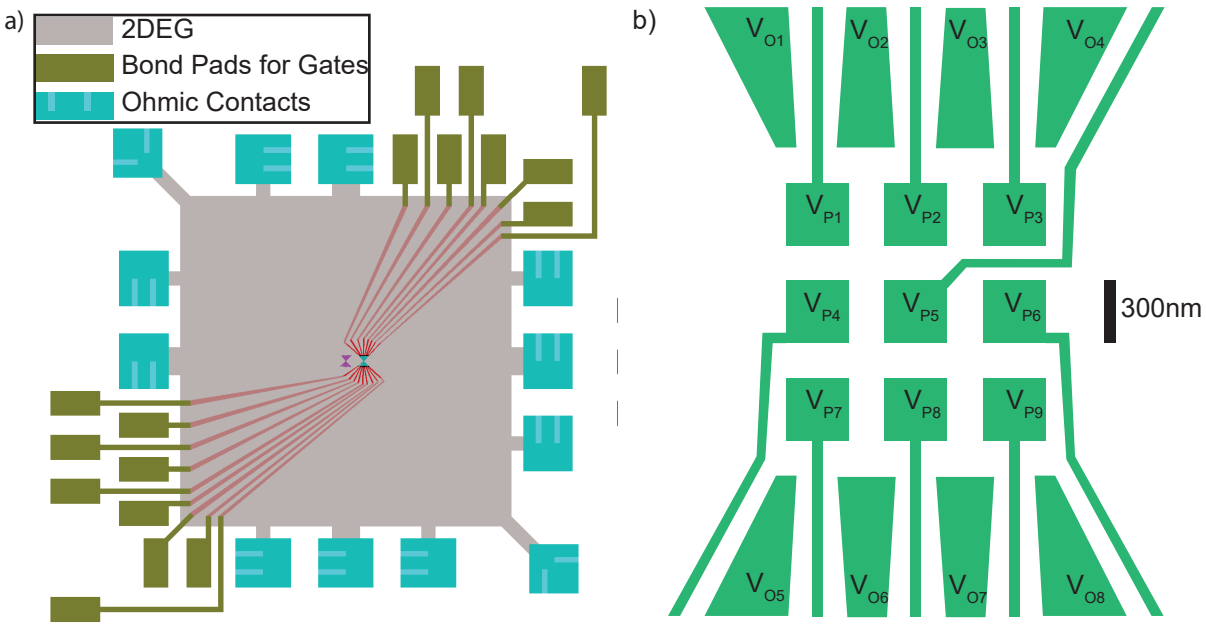


FIGURE 3.11: **Device Overview. a)** Mesa overview, showing the 2DEG in grey, ohmic contacts in blue and gate bond pads in brown. The fan-out of gates are shown in red. **b)** Layout of the pixel array, note especially how the middle row of pixels is connected to the bond pads by thin lines out from the pixel array.

QPC operation might not be possible, the pixels should allow tuning away some of this disorder. Of course in practice we will never really know which factor is the main contributor to improving the QPC features, since the potential cannot be exactly known.

3.4.2 Voltage Bias vs Current Bias

When performing measurements on the device we have been using two measurement techniques, voltage bias and current bias. They have been used in two different scenarios, we have been using voltage bias for measuring the quantum point contact conductance and current bias when measuring the quantum Hall effect.

When voltage biasing the device, we set up a 4-point measurement applying a voltage directly to an ohmic to create a potential difference across the QPC, measuring the current flowing out of a different ohmic and measuring the potential difference between two different ohmics situated on different sides of the QPC constriction. An example of this is shown in Fig. 3.12 a), where we are using the qdac voltage source to supply a DC offset to the ohmic, and in addition we use a lockin amplifier to supply an AC excitation to measure both the voltage drop and current using lockin techniques. The measurements are performed on 2 different lockins, one that gets its signal from an LI-75 voltage amplifier with amplification factor 100, this is

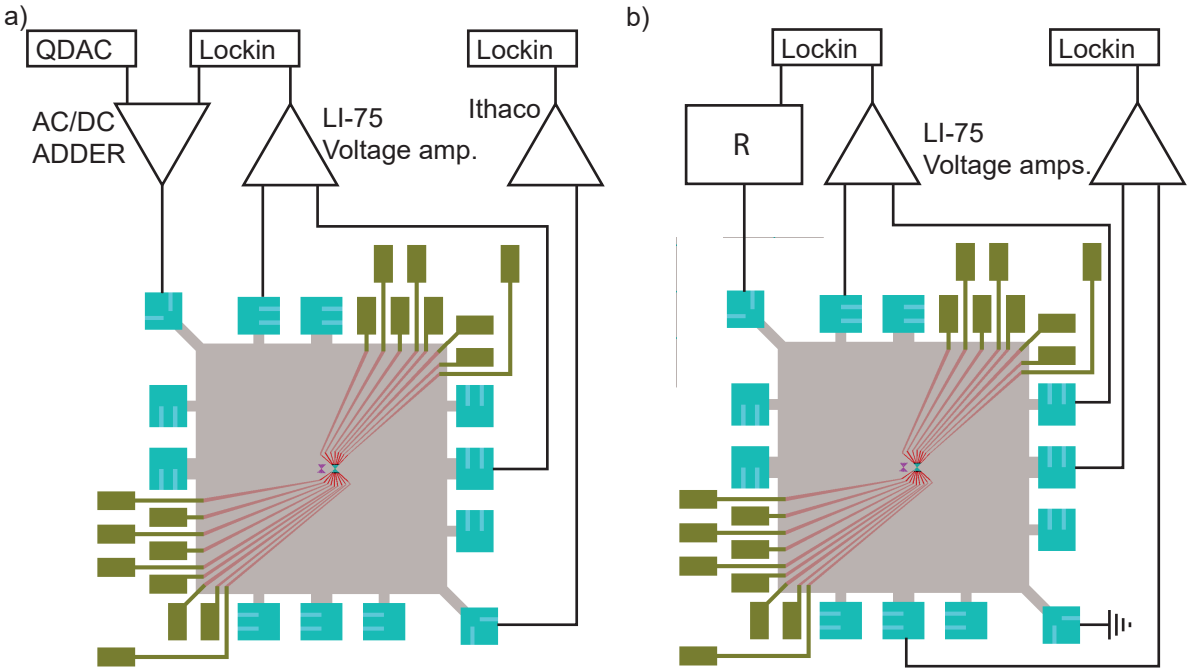


FIGURE 3.12: **Voltage and Current Bias.** Different setups used for measuring the quantum point contact features and quantum Hall effect features, voltage bias is used for the quantum point contact and current bias for the quantum Hall effect. **a)** Voltage bias. **b)** Current bias.

measuring the voltage difference between the two ohmics. The other lockin is fed a signal from the Ithaco current to voltage converter, thus measuring the current running out of an ohmic.

In Fig. 3.12 b) an example of a current bias measurement is shown, instead of applying a voltage directly, we apply a voltage first to a large resistor, typically 10MΩ. This is to ensure that the total resistance of the system (lines, device and the large resistor) is dominated by the large resistance [6]. Thus the current throughout the system should be independent of device and fridge resistance, this also means we do not measure the current flowing out. Since we have been using this setup for measuring quantum Hall effect we have still used two lockin amplifiers fed signals through LI-75 voltage amplifiers in A-B mode, exactly the same as in voltage bias setup. The two measurements are done such that one measures the voltage difference along the flow of current, while the other measures the voltage difference across the flow of current.

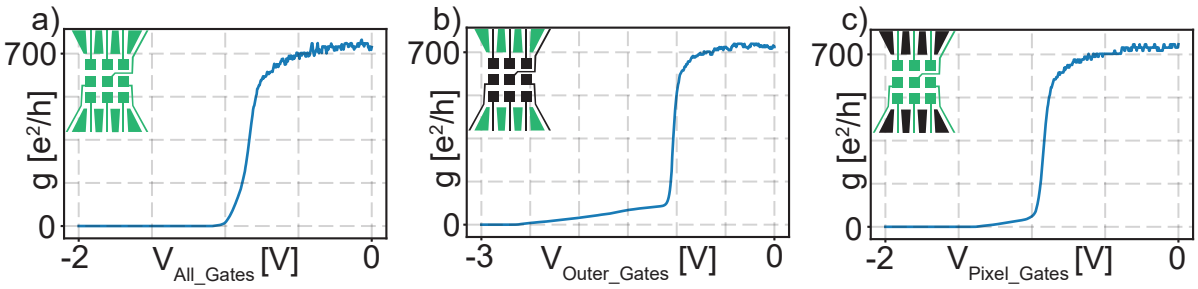


FIGURE 3.13: **Pinch Off Curves. a)** Pinch off with all gates. **b)** Pinch off with only outer gates, clear depletion at around -1V, the outer gates here manage to actually pinch off the channel completely, but this was not always the case. **c)** Pinch off with only pixel gates, clear depletion at -1V and then slow pinch off before -1.5V.

3.5 Preliminary Results

3.5.1 Pinch off and hand tuned QPCs

Some of the first measurements we did on this device are pinch off curves, these measurements are crucial in determining the gating action and give insight into when depletion occurs. Depletion is especially important for our purposes, since forming a QPC requires a very large portion of the 2DEG to be completely depleted only allowing a narrow channel to exist between to electron reservoirs. In Fig. 3.13 we show the pinch-of-curves taken for all gates combined, outer gates combined and pixel gates combined. All gates pinching off is completely as expected and is mainly done as a first test to see some form of gating action. The test with outer gates in Fig. 3.13b) has some more interesting features, it shows clearly that the outer gates deplete the 2DEG region beneath them at around 1V, this is the steep drop in conductance. From the depletion of the outer gate regions, as the gates become stronger, they are also able to pinch off the channel between them entirely, this however has not always been the case. We have seen varying degrees of pinch off from the outer gates, sometimes they can not pinch off at -3V, sometimes they pinch off completely at -2V, we have not arrived at a satisfying explanation for this behaviour and have not been able to discern any specific pattern for when they act in which way. As a last note on the outer gate pinch off, this inconsistent behaviour was a large factor in the choice to move the measurement sweeping style away from sweeping the outer gates to instead sweeping on a set of the pixel gates. In Fig. 3.13c) the pixel gates are the only gates being touched, meaning that the outer gates are kept to 0V, the fact that the pixel gates pinch off on their own is quite surprising and worrisome for the use of the device. Ideally the pixel gates should only affect the small region located beneath them inside the constriction

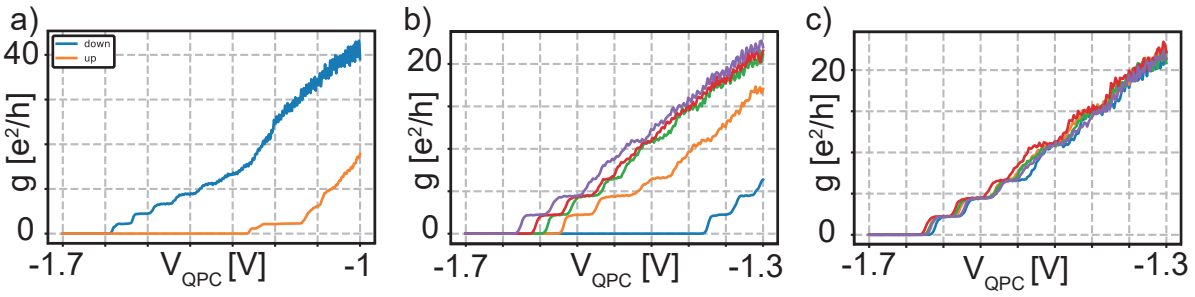


FIGURE 3.14: **Waiting Times.** Strange behaviour of the device forced us to include a jump to 0V and some waiting time there, before taking a new measurement. All measurements shown are done solely with the right QPC. **a)** Sweeping down and up in sequence **b)** Only sweeping down waiting for 10s at 0V and 3s at -1.3V for the QPC gates. **c)** Only sweeping down waiting for 20s at 0V and 10s at -1.3V for the QPC gates.

of the QPC made by the outer gates. The fact that they can pinch off alone suggests that the lines connecting the gates to their bond pads have a much stronger effect on the 2DEG than intended. This also suggests that the lines very close to the pixels have the same strong effect, this might strongly influence how the pixels affect the 2DEG in the constriction region and will have unintended effects. This issue does not mean that the device is of no use at all, but it complicates the interpretation of optimisation results.

3.5.2 Magnet Field Sweeps

Here I will present the results of preliminary magnetic field measurements, the measurements are conducted with current bias using an excitation of 4nA. Both V_{xx} and V_{xy} are measured through voltage amplifiers with lockin amplifiers, see ch. 3.4.2. For both measurements the gates are grounded and all the ohmics not actively being used are floating.

In Fig. 3.15 the first magnetic field sweep is shown, in a) the setup used is shown, this setup was chosen because it makes a regular Hall bar like setup and the Hall voltage measurement, V_{xy}, could be used directly for diagonal voltages V_D, with the QPC turned on. In b) the result is shown, some features are easily recognizable in R_{xx}, there are clear peaks forming at higher voltages and R_{xy} as well seems to even out more in the regions where R_{xx} is not on a peak. This indicates that Landau levels are getting depleted as we move higher in field, but with this measurement it is not obvious to discern what is actual features and what isn't. We suspected that even though the gates were grounded their presence might cause strain at the interface or in some other way interfere with the 2DEG beneath them.

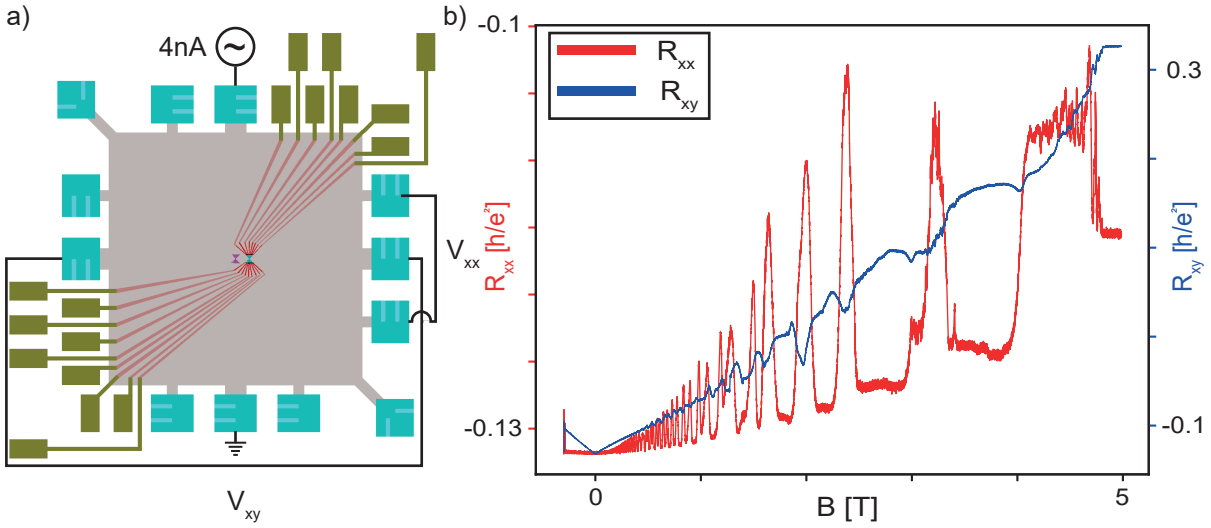


FIGURE 3.15: **First Magnet Field Sweep. a)** Measurement setup for measuring V_{xx} and V_{xy} in current bias. **b)** Result with R_{xx} in red and R_{xy} in blue. R_{xx} shows oscillations as expected, but does not drop to 0Ω in between peaks. R_{xy} also shows plateaus, where plateaus are consistent in location with R_{xx} but not flat.

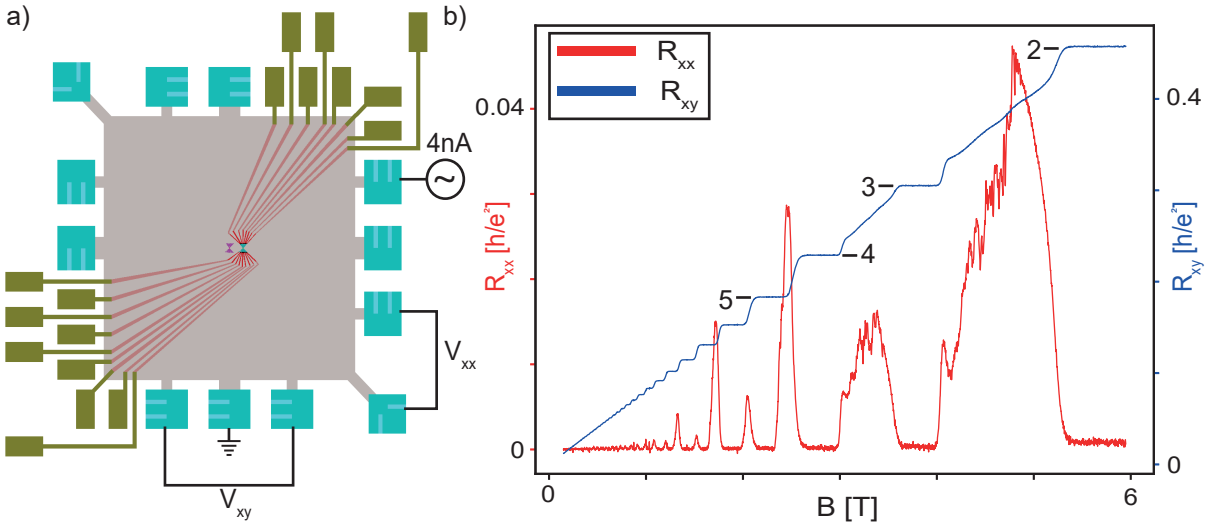


FIGURE 3.16: **Second Magnet Field Sweep. a)** Measurement setup for measuring V_{xx} and V_{xy} in current bias. Here all instruments are applied to one side of the fan-out of gates. **b)** A much better result than previously. Filling factors are assigned by first calculating the density from SdH oscillations, then calculating the filling factor locations in B from that density, method described in text.

Our next step was to perform the measurement again, but with all ohmics used located on 1 side of the gate-fanouts. This would allow the current to flow between contacts without going through the regions of the 2DEG below the gates. In Fig. 3.16a) we show the setup used, all ohmics used for measurements are located in the lower right corner of the mesa. In Fig. 3.16b) the resulting measurement is shown, where the improvement is dramatic, both R_{xx} and R_{xy} now show the features known from integer quantum Hall effect. But we do not identify any plateaus we suspect to arise from fractional quantum Hall effect. The plateaus we do see have been assigned filling factors according to the following method [7]:

1. Identify peak locations of R_{xx} in magnetic field, the values used are:
   $1.71T$, $2.05T$, $2.46T$, $3.34T$, $4.77T$

2. Calculate the density of electrons from the following formula:

$$n_s = \frac{2e/h}{\frac{1}{B_i} - \frac{1}{B_{i+1}}} \tag{3.40}$$

   Where B_i and B_{i+1} are sequential peak locations from the list above.

3. Calculate the the location of specific filling factors ν using the above calculated density.

$$B_j = \frac{n_s}{2e\nu_j/h} \tag{3.41}$$

When calculated from this method the locations of filling factors line up with locations of plateaus in R_{xy}, however this method comes with several error modes attached. The locations are not exact extractions but instead chosen by eye, the calculated density, $n_{s_{cal}} = 5.21\text{e+}11\text{cm}^{-2}$ is higher than the reported density of the wafer from the manufacturer, $n_{s_{rep}} = 3.06\text{e+}11\text{cm}^{-2}$. The density calculated is an average of the four sequential pairs and even within these four pairs there is quite a large variance, with them ranging from $n_s = 4.50\text{e+}11\text{cm}^{-2}$ to $n_s = 5.93\text{e+}11\text{cm}^{-2}$. To conclude the method is not strict enough for us to trust it completely and the data gathered is inconsistent with data reported by the manufacturer, at these values of magnetic field we expected to see clear signs of fractional quantum Hall effect, but these were not present. This suggests that there might be something wrong with this chips material. It has been suggested that the material was harmed during fabrication due to too high temperature when annealing ohmics.

3.6 Optimization Results

This section will go over 2 seperate runs of the algorithm, both runs are performed a bit differently that we had initially intended. The original plan was to make the constriction solely with the outer gates, then sweep the pixel gates combined to close off the channel, and optimise on the offset. This unfortunately did not produce any meaningful results, we suspect that dimensions are too big, with a spacing of the outer gates of 1.4μm the constriction was not sufficiently narrow to support a quantum point contact. Instead for both runs we have chosen to make a both narrower and shorter QPC by instead sweeping a set of the pixel gates, and optimising on the rest. The two presented runs are one with the right QPC and one with the middle QPC where an additional parameter is introduced to the algorithm, allowing it to move the measurement window up or down in gate voltage. For the latter middle QPC run, the loss function used has been changed to accommodate the effect of moving the window of measurement around. The code for the right QPC run can be found in appendix D.

3.6.1 Right QPC

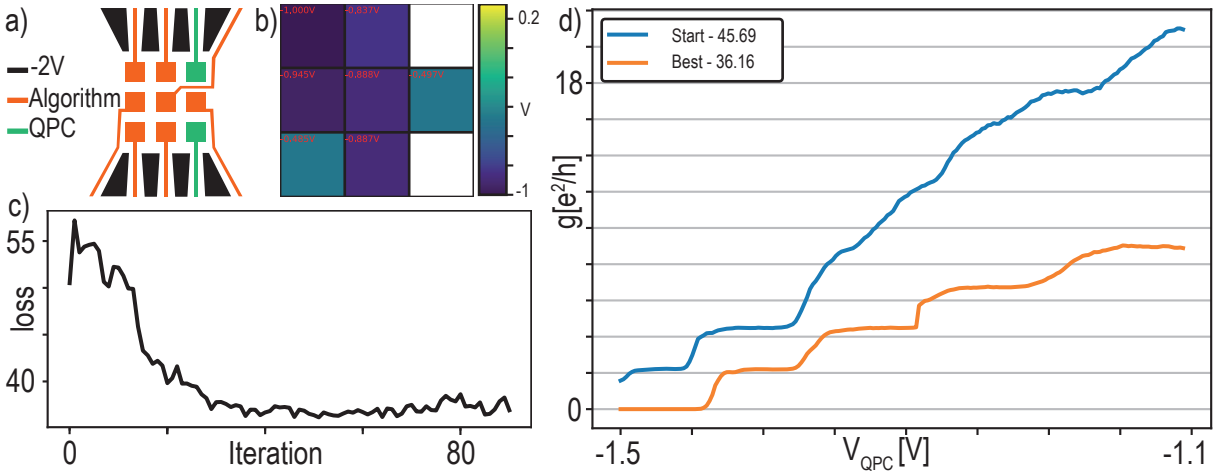


FIGURE 3.17: **Algorithm Results Right QPC.** Overview of the results running the algorithm of the rightmost QPC. **a)** Gate assignments. **b)** Voltages applied to pixels for best result, exact values shown in red. **c)** Best loss measured for each iteration. **d)** Traces from start (0V) on pixels and the best measured trace, showing a clear improvement where the QPC behaviour is present for 2 additional steps.

First I will show an algorithm run coming from the right QPC, this has also previously shown the best handmade QPCs. The run is performed with a DC bias and AC excitation in voltage bias, see Fig. 3.12a). We are using 100μV DC, and 40μV AC with a frequency of 17Hz, furthermore the run was performed at 100mT field. The fields presence is due to the dilution

refrigerator being shared with another team, but the small magnet field should theoretically make quantum point contacts easier to make, due to splitting left and right moving electrons spatially. In Fig. 3.17 an overview of this run is shown. The algorithm in total has 7 parameters, orange in 3.17a), these seven parameters are optimised freely in the range -1V to 0.3V. The two green pixels are used to narrow the QPC and swept within the same -1.1V to -1.5V range each time. Getting the most consistent results for the right QPC required including a jump to 0V for all of the pixel gates, waiting for 20s, before jumping them to the starting values of each trace, waiting another 10s and then starting the measurement. In b) the resulting voltages obtained from this run are shown. One expectation we had when starting the experiment was that the middle row of pixels would be best left at positive voltages to deepen the trench along the QPC. At this point it is difficult to say how much of this discrepancy is due to the effect of the thin lines connecting to the pixels, since they are routed on the outside of the other pixels, their effect could be screening the QPC gates. In c) the best loss encountered is shown for each iteration, already at 40 iterations the loss is starting to saturate at values around 36-38. At this point the average change of voltages within an iteration is less than 100mV and the difference between each measurement can just as well be dominated by hysteresis. The starting measurement with all pixels at 0V and the best measurement is shown in d), there is clear improvement throughout the run. The best evaluated measurement shows clear step-like features and also pinches off completely.

An investigation into the algorithm performance is shown in Fig. 3.18, where a select number of iterations are shown. Going through the iterations a few obvious things occur that are good indicators that the algorithm is working. First, the measurements by eye begin to look better as we go through the iterations. At 10 iterations we begin to see some step like features, at 25 iterations a few more steps are achieved starting from 0 conductance, and at 50 iterations we see even better staircases. Second, both the voltages and the measurements start becoming closer to each other. As we move toward a minimum in the loss landscape the voltages applied to the pixels converge to some value, and those values give reproducible results.

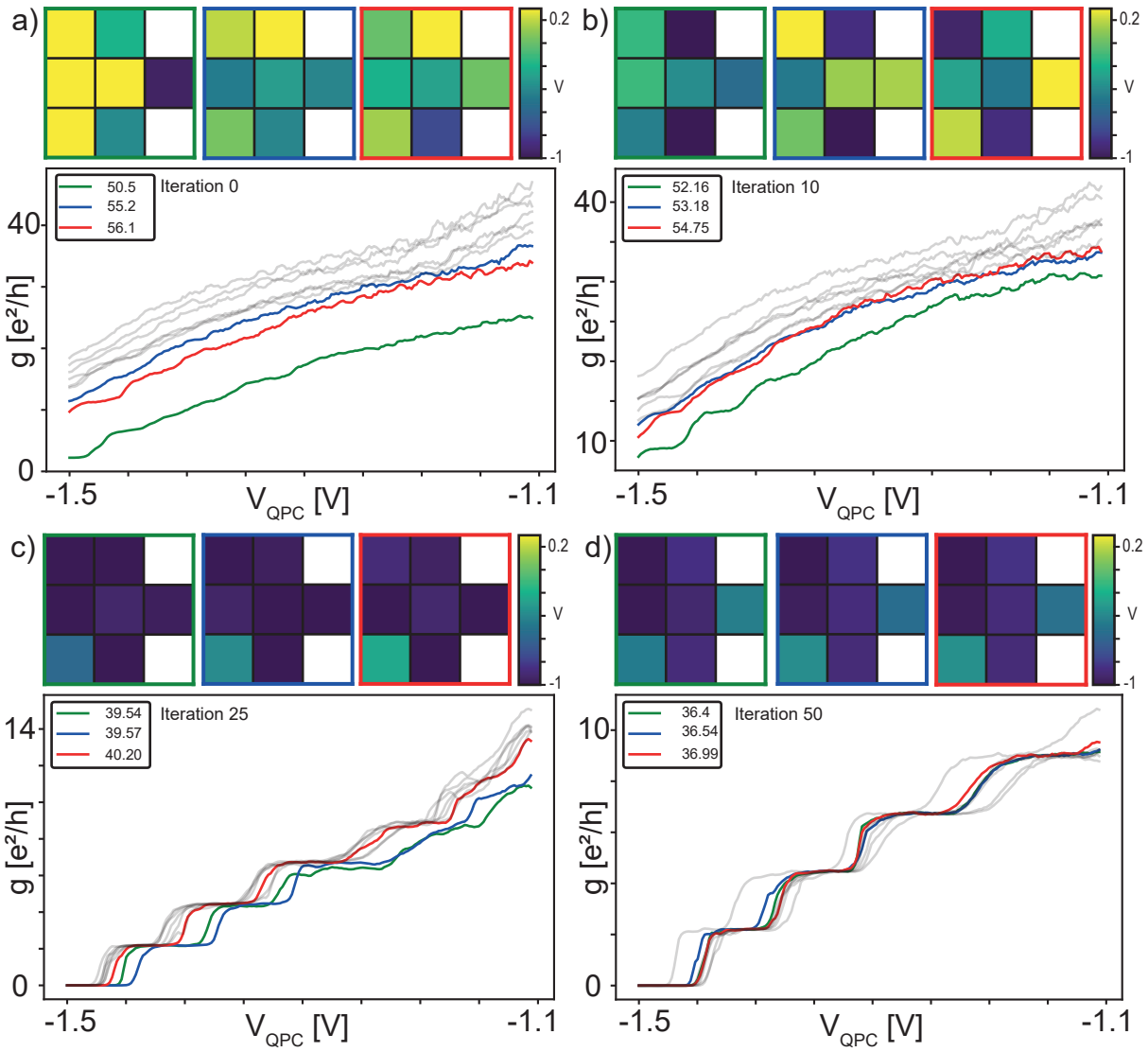


FIGURE 3.18: **Iteration Overview Right QPC.** Overview of the algorithm evolving through iterations. Each panel contains all the traces obtained in an iteration, the best three highlighted in green, blue and red. Their corresponding voltages are framed by the same colour. This really shows a good improvement by the algorithm, and it shows that both the algorithm and loss function has good potential for improving QPCs. **a)** iteration 0 **b)** iteration 10 **c)** iteration 25 **d)** iteration 50.

3.6.2 Middle QPC

The middle QPC, has generally shown worse behaviour than the right QPC, but we have tried running the optimisation on this regardless. The run is again performed in voltage bias, this time with 125μV DC bias, 60μV AC excitation and 150mT field. As mentioned previously, we have included an additional parameter for the algorithm to control, this parameter controls the window of operation. With a fixed window size of 800mV we allow the algorithm to freely move this window such that it can take on values between -2V and 0V, and optimise this parameter exactly as the rest. To account for moving the window around, we have introduced a

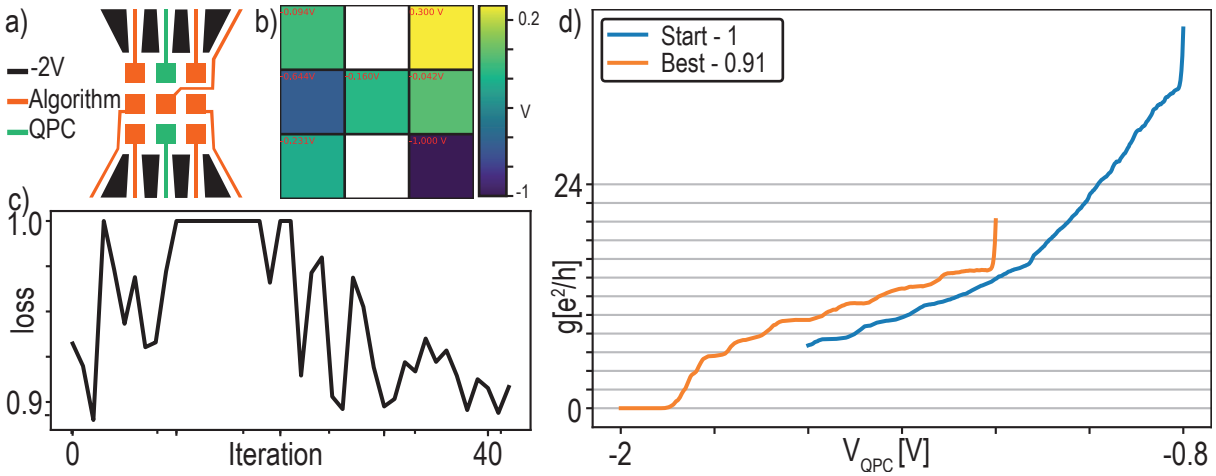


FIGURE 3.19: **Algorithm Results Middle QPC.** Overview of the results running the algorithm of the middle QPC. **a)** Gate assignments. **b)** Voltages applied to pixels for best result, exact values shown in red. **c)** Best loss measured for each iteration. **d)** Traces from start (0V) on pixels and the best measured trace.

different loss function that only evaluates on conductance values between $0.01\frac{e^2}{h}$ and $11\frac{e^2}{h}$, see ch. 3.3.1, previous runs without this condition saw the window being pushed far negative such that no conductance was allowed. Measurements without values spanning this entire range are given a score of 1.

In Fig. 3.19 we show an overview of the results from the optimisation run, while the result is not as good as the previously shown run for the right QPC, it still shows some promising results produced by the algorithm. The layout is the same as for Fig. 3.17, and the things to take note of are that we see a minima in the loss throughout the run at a very early iteration, and it even ends up being the global minimum shown in b) and d) as well. The loss overall does not decrease by a lot, but it is trending downwards. The run was unfortunately cut short by an instrumentation error, so it did not get to progress as far as we had hoped.

Fig. 3.20 again shows a set of iterations, where it is more obvious that the algorithm moves the window to the lowest part of the allowed region, and it also has a clear trend in the applied voltages in d). While not as homogeneous as the run with the right QPC, the voltages are still at this point fairly close and show a distinctly different pattern than for the right QPC. Whether it would have ended in a similar configuration as the right QPC is unfortunately too early to tell, and it is likely more difficult to find an optimum as the middle QPC seems more hysteretic. Interestingly we see some step-like features in b) at iteration 15, they just happen to occur outside the window we are trying to optimise in, which is $0.01\frac{e^2}{h}$ to $11\frac{e^2}{h}$, this means they will have no effect on the run going forward with this loss function.

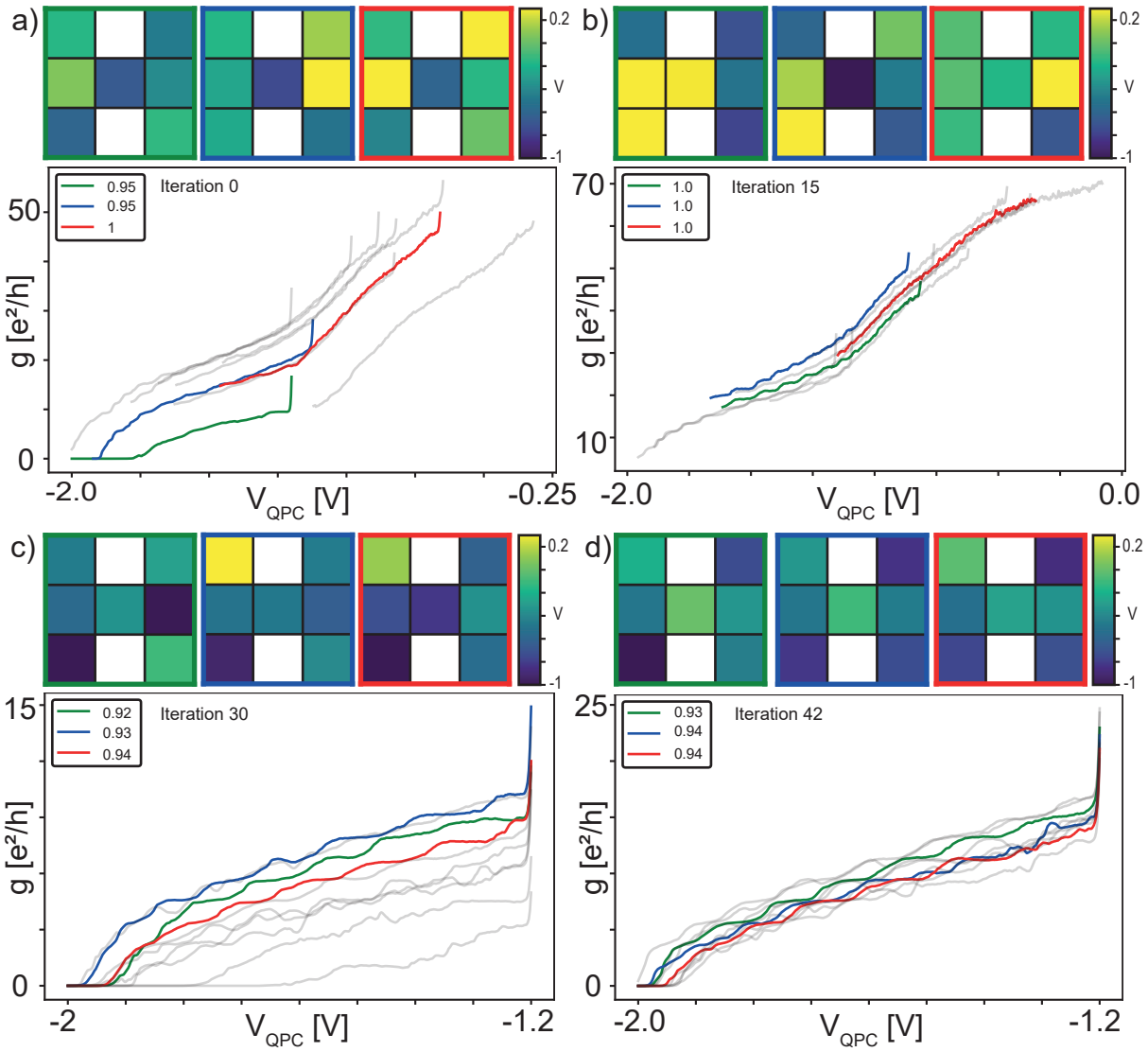


FIGURE 3.20: **Iteration Overview Middle QPC.** Overview of the algorithm evolving through iterations. Each panel contains all the traces obtained in an iteration, the best three highlighted in green, blue and red. Their corresponding voltages are framed by the same colour. **a)** iteration 0 **b)** iteration 15 **c)** iteration 30 **d)** iteration 42.

3.7 Summary

For the purpose of autonomously controlling the spatial flow of electrons within the constriction of a quantum point contact, we have developed an algorithm and a set of loss functions that shows promising results. The algorithm has been tested on a basic simulation using the Kwant python package for quantum transport, and later tested on a real device fabricated by Bertram Brovang [4]. The algorithm can easily be adapted to specific applications, which has been useful since the simulated device and the actual device behaved quite differently. Despite challenges associated with hysteretic gating behaviour in the actual device, conductance staircases typical for quantum point contacts could be tuned up algorithmically. Overall, this shows

promise for optimising the core problem of modulating the potential at the 2DEG. Further development of both the algorithm and devices may lead to interesting experiments within the realm of quantum point contacts and quantum Hall effects.

Chapter 4

Outlook

Using two very different material and device geometries, namely capacitively coupled quantum dots in silicon and gate-controlled quantum point contacts on GaAs Hall bars, we have demonstrated promising features of combining machine learning and quantum devices. As research into quantum devices progress, the devices themselves become more and more complex, the devices of the future might have many more axes of interaction than we even dream off now. Evidently developing programs that can interact with quantum devices, without human oversight will be needed, and it is very possible that optimisation based on artificial intelligence can lead to results not otherwise obtained by human scientists. The two experiments carried out served two different specific purposes, one was to estimate the multi dimensional polytope of a charge state in an array of quantum dots, the other two optimise the shape of a potential in a 2DEG to improve electrical capabilities. While the first was very successful, there is still room for further interesting experiments, a first step would be to test out the algorithm in even higher dimensions. A feasible platform for testing the algorithm in higher dimensions could be a 2xN array made in a long quantum wire [22, 10]. For the second algorithm, namely optimising spatially the potential of quantum point contact, our preliminary results from non-ideal devices show promise and would benefit greatly from more well-behaved devices. We are already in the process of fabricating new devices hoping that they will not show the same issues. On another front, the optimisation algorithm is not specific to the quantum point contact physics, only the loss function is specialised. The algorithm could be readily applied to a wide variety of different optimisation problems in quantum devices, the only changes needed to the loss function.

Appendix A

Kwant Simulation

```
import kwant
from kwant.digest import uniform
import numpy as np
from math import atan2, pi, sqrt
from cmath import exp
import matplotlib.pyplot as plt
from matplotlib.collections import PatchCollection
from matplotlib.patches import Rectangle
from .new_disorder import make_disorder, make_pixel_disorder
from types import SimpleNamespace
import scipy.sparse.linalg as sla

def rectangular_gate_pot(dims):
    """Compute the potential of a rectangular gate.

    The gate hovers at the given distance over the plane where the
    potential is evaluated.

    Based on J. Appl. Phys. 77, 4504 (1995)
    http://dx.doi.org/10.1063/1.359446
    """
    distance, left, right, bottom, top = dims[0], dims[1], dims[2],
        dims[3], dims[4]
```

```
    d, l, r, b, t = distance, left, right, bottom, top

    def g(u, v):
        return atan2(u * v, d * sqrt(u**2 + v**2 + d**2)) / (2 * pi)

    def func(x, y, voltage):
        return voltage * (g(x-l, y-b) + g(x-l, t-y) +
                          g(r-x, y-b) + g(r-x, t-y))

    return func


def make_gates(distance_to_gate=5,left=30,right=50,spacing=2.5,W=80,
    L=80,gates_outside=0):
    pixel_size=(right-left-2*spacing)/3
    array_size=3*pixel_size+2*spacing

    center=(W/2,L/2)

    gate1dims=[distance_to_gate,left,right,-gates_outside,center[0]-
        array_size/2-spacing]
    gate11dims=[distance_to_gate,left,right,center[0]+array_size/2+
        spacing,W+gates_outside]

    bottom_of_array=center[0]-array_size/2
    gates=[gate1dims]
    for i in range(3):
        for j in range(3):
            gates.append([distance_to_gate,
                          left+j*(pixel_size+spacing),
                          left+j*(pixel_size+spacing)+pixel_size,
```

```
                                bottom_of_array+i*(pixel_size+spacing),
                                bottom_of_array+i*(pixel_size+spacing)+
                                    pixel_size])
        gates.append(gate11dims)
        return gates

#standard size
if False:
    W=70
    L=60
    allgatedims=make_gates(left=20,right=40,W=W,L=L,spacing=2,
        gates_outside=10)
else:
    #longer
    W=70
    L=120
    allgatedims=make_gates(left=int(L/2-10),right=int(L/2+10),W=W,L=
        L,spacing=2,gates_outside=10)

# allgatedims=[gate1dims,gate2dims,gate3dims,gate4dims,gate5dims,
    gate6dims,gate7dims,gate8dims,gate9dims,gate10dims,gate11dims]

#gate 1 and 11 are the outer gates, 2-10 are the pixel array
_gate1 = rectangular_gate_pot(allgatedims[0])

_gate2 = rectangular_gate_pot(allgatedims[1])
_gate3 = rectangular_gate_pot(allgatedims[2])
_gate4 = rectangular_gate_pot(allgatedims[3])
_gate5 = rectangular_gate_pot(allgatedims[4])
_gate6 = rectangular_gate_pot(allgatedims[5])
_gate7 = rectangular_gate_pot(allgatedims[6])
```

```
_gate8 = rectangular_gate_pot(allgatedims[7])
_gate9 = rectangular_gate_pot(allgatedims[8])
_gate10 = rectangular_gate_pot(allgatedims[9])

_gate11 = rectangular_gate_pot(allgatedims[10])




def qpc_potential(site, V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11
    ):
    x, y = site.pos
    return _gate1(x, y, V1) + _gate2(x, y, V2) + _gate3(x, y, V3) +
        _gate4(x,y,V4) + \
            _gate5(x, y, V5) + _gate6(x, y, V6) + _gate7(x, y, V7) +
                _gate8(x,y,V8) + \
            _gate9(x, y, V9) + _gate10(x, y, V10) + _gate11(x, y, V11
                )

disorder_values=make_disorder(L, W, length_scale=5,random_seed=2).T
# print(disorder_values)
pixel_disorder_values=make_pixel_disorder(L,W,allgatedims[1:10])

def disorder(site,U0):
    x,y=site.tag
    # print(x,y)
    return disorder_values[x,y]*U0

def pixel_disorder(site,U0):
    x,y=site.tag
    # print(x,y)
```

```
    return pixel_disorder_values[x,y]*U0

def disorder_old(site, U0, salt=13):
    return  U0 * (uniform(repr(site), repr(salt)) - 0.5)

def hopping(site_i, site_j, phi):
    xi, yi = site_i.pos
    xj, yj = site_j.pos
    return -exp(-0.5j * phi * (xi - xj) * (yi + yj))


class pixelarrayQPC():
    def __init__(self,W=W,L=L,plot=False,disorder_type='regular'):
        #
           ------------------------------------------------------------------

        # Set up KWANT basics
        # Parameters are:
        # phi, flux through unit cell of the latice phi=Ba^2
        # V1-V3 voltage on the three gates,
        # salt is a parameter controlling random nr generation in
           kwant.digest.uniform, used in disorder
        # U0 parameter controlling the amount of disorder in the
           system
        # energy is the fermi level
        # t is a hopping parameter
        #
        #
           ------------------------------------------------------------------

        lat = kwant.lattice.square(norbs=1)
```

```
        self.phi=0
        self.salt=13
        self.U0=0
        self.energy=1
        self.t=1


        self.V1=-2

        self.V2=0
        self.V3=0
        self.V4=0
        self.V5=0
        self.V6=0
        self.V7=0
        self.V8=0
        self.V9=0
        self.V10=0

        self.V11=-2



def make_lead_x(start,stop, t=1):
    syst = kwant.Builder(kwant.TranslationalSymmetry([-1,
        0]))
    syst[(lat(0, y) for y in np.arange(start,stop))] = 4 * t
        #no disorder in lead
    syst[lat.neighbors()] = hopping
    return syst
```

```
def make_barrier(pot,dis, W=W, L=L, t=1):
    def onsite(s, V1, V2, V3, V4, V5, V6, V7, V8, V9, V10,
       V11, U0, salt, t):
        return 4 * t - pot(s,V1, V2, V3, V4, V5, V6, V7, V8,
            V9, V10, V11) + dis(s, U0)
    # Construct the scattering region.
    sr = kwant.Builder()
    sr[(lat(x, y) for x in range(L) for y in range(W))] =
       onsite
    sr[lat.neighbors()] = hopping



    lead = make_lead_x(start=0,stop=W, t=self.t)
    sr.attach_lead(lead)
    sr.attach_lead(lead.reversed())



    return sr

if disorder_type=='pixel':
    self.disorder_func=pixel_disorder
else:
    self.disorder_func=disorder
self.qpc = make_barrier(qpc_potential,self.disorder_func,t=
   self.t)
# Plotting the gates and sites/leads and potential
if plot:
    fig,ax=plt.subplots()
    kwant.plot(self.qpc,ax=ax)
    rects=[]
```

```
for gate,dims in enumerate(allgatedims):
    rect=Rectangle((dims[1],dims[3]),dims[2]-dims[1],
        dims[4]-dims[3],zorder=999)
    rects.append(rect)

    # ax.text(x=dims[1],y=dims[3],s=str(gate))


pc=PatchCollection(rects, facecolor='green',alpha=10)
ax.add_collection(pc)


xlims=ax.get_xlim()
ylims=ax.get_ylim()
fig.savefig(r'C:\Users\Torbj rn\Google␣Drev\UNI\
    MastersProject\Thesis\Figures\QPC␣chapter\algorithm\
    lattice.pdf',format='pdf')
#copy paste of plot potential section
bounds=((0,L),(0,W))
fig,ax=plt.subplots()
vals=np.zeros([bounds[0][1]-bounds[0][0],bounds[1][1]-
    bounds[1][0]])
i=0
for x in np.arange(bounds[0][0],bounds[0][1]):
    j=0
    for y in np.arange(bounds[1][0],bounds[1][1]):
        site=SimpleNamespace(tag=(x,y),pos=(x,y))
        vals[i,j]=4*self.t-qpc_potential(site, self.V1,
            self.V2, self.V3, self.V4, self.V5, self.V6,
            self.V7, self.V8, self.V9, self.V10, self.V11
            )+self.disorder_func(site,self.U0)
```

```
                j+=1
            i+=1
        h=ax.imshow(vals.T,origin='lower',extent=(bounds[0][0],
            bounds[0][1],bounds[1][0],bounds[1][1]))
        plt.colorbar(h)
        # kwant.plotter.map(self.qpc, lambda s: 4*self.t-
            qpc_potential(s, self.V1, self.V2, self.V3, self.V4,
            self.V5, self.V6, self.V7, self.V8, self.V9, self.V10
            , self.V11) \
        #                 +self.disorder_func(s,self.U0))

    self.fqpc = self.qpc.finalized()

def transmission(self):
    Params=self.__dict__
    smatrix = kwant.smatrix(self.fqpc, self.energy, params=
        Params)
    return smatrix.transmission(1,0)

def plot_disorder(self,bounds=((0,L),(0,W)),ax=None):
    if ax is None:
        fig,ax=plt.subplots()
    vals=np.zeros([bounds[0][1]-bounds[0][0],bounds[1][1]-bounds
        [1][0]])
    i=0
    for x in np.arange(bounds[0][0],bounds[0][1]):
        j=0
        for y in np.arange(bounds[1][0],bounds[1][1]):
            site=SimpleNamespace(tag=(x,y),pos=(x,y))
            vals[i,j]=self.disorder_func(site,self.U0)
            j+=1
```

```
            i+=1
        h=ax.imshow(vals.T,origin='lower',extent=(bounds[0][0],
            bounds[0][1],bounds[1][0],bounds[1][1]))
        plt.colorbar(h)
        return fig,ax,vals

    def plot_potential(self,bounds=((0,L),(0,W)),ax=None):
        if ax is None:
            fig,ax=plt.subplots()
        vals=np.zeros([bounds[0][1]-bounds[0][0],bounds[1][1]-bounds
            [1][0]])
        i=0
        for x in np.arange(bounds[0][0],bounds[0][1]):
            j=0
            for y in np.arange(bounds[1][0],bounds[1][1]):
                site=SimpleNamespace(tag=(x,y),pos=(x,y))
                vals[i,j]=4*self.t-qpc_potential(site, self.V1, self
                    .V2, self.V3, self.V4, self.V5, self.V6, self.V7,
                     self.V8, self.V9, self.V10, self.V11)+self.
                    disorder_func(site,self.U0)
                j+=1
            i+=1
        h=ax.imshow(vals.T,origin='lower',extent=(bounds[0][0],
            bounds[0][1],bounds[1][0],bounds[1][1]))
        plt.colorbar(h)
        return fig,ax,vals.T

    def set_all_pixels(self,val):
        if isinstance(val,float):
            self.V2=val
            self.V3=val
```

```
            self.V4=val
            self.V5=val
            self.V6=val
            self.V7=val
            self.V8=val
            self.V9=val
            self.V10=val
        elif isinstance(val,np.ndarray) or isinstance(val,list):
            self.V2=val[0]
            self.V3=val[1]
            self.V4=val[2]
            self.V5=val[3]
            self.V6=val[4]
            self.V7=val[5]
            self.V8=val[6]
            self.V9=val[7]
            self.V10=val[8]

    def plot_current(self, eig_num):
        # Calculate the wave functions in the system.
        fig,ax=plt.subplots()
        Params=self.__dict__
        ham_mat = self.fqpc.hamiltonian_submatrix(sparse=True,
            params=Params)
        evals, evecs = sla.eigsh(ham_mat.tocsc(), k=eig_num, sigma
            =0)

        # Calculate and plot the local current of the 10th eigenmode
            .
        J = kwant.operator.Current(self.fqpc)
```

```
        current=0
        for i in range(eig_num):
            current += J(evecs[:, i], params=Params)

        kwant.plotter.current(self.fqpc, current, colorbar=True,ax=
            ax)
        return fig,ax

    def wave_func(self,lead=0,ax=None,self_plot=True,plot_both=False
        ):
        if ax==None:
            fig,ax=plt.subplots()

        Params=self.__dict__
        wfs = kwant.wave_function(self.fqpc, energy=self.energy,
            params=Params)
        if plot_both:
            scattering_wf1=wfs(0)
            scattering_wf2=wfs(1)

            total=np.sum(abs(scattering_wf1)**2, axis=0)+np.sum(abs(
                scattering_wf2)**2, axis=0)

            h=ax.imshow(total.reshape((L,W)).T,origin='lower',cmap='
                inferno')
            return fig,ax,h
        scattering_wf = wfs(lead)  # all scattering wave functions
            from lead "lead"
        if self_plot:
            h=ax.imshow(np.sum(abs(scattering_wf)**2, axis=0).
                reshape((L,W)).T,origin='lower',cmap='inferno')
```

```
        # kwant.plotter.map(self.fqpc, np.sum(abs(scattering_wf)**2,
            axis=0),ax=ax)
        return fig,ax,h
```

Appendix B

CMAES

```
import cma

# general
import numpy as np
import os
import json
import pickle

def folder_name(data_path):
    if not os.path.exists(data_path+"outcmaes"):
        os.mkdir(data_path+"outcmaes")
    folders=folders=list(os.walk(data_path+"outcmaes/"))[0][1]
    lis=[int(f) for f in folders]
    lis.append(0)
    newfolder=data_path+'outcmaes/'+'{}/'.format(max(lis)+1)
    return newfolder


def optimize_cma(func_to_minimize,datahandler,start_point,maxfevals
    =99999,sigma=0.5,stop_time=None,callbacks=[None],args=[],options
    ={}):
    #make a seperate folder for this run
    data_path=datahandler.data_path
```

```
newfolder=folder_name(data_path)
print("data saved to:")
print(newfolder)
os.mkdir(newfolder[:-1])

#start a datadict and measure the starting point, cma-es for
    some reason doesnt measure the starting point
datadict={'next_key':0,'measurements':{},'starting_point':{'
    next_key':0,'measurements':{}}}
func_to_minimize(start_point,datadict['starting_point'])

args_send=[datadict]
args_send.extend(args)



options_send={'maxfevals':maxfevals,'verb_filenameprefix':
    newfolder}
for key in options:
     options_send[key]=options[key]
if not stop_time==None:
     options_send['timeout']=stop_time

x,es=cma.fmin2(func_to_minimize,start_point,sigma0=sigma,args=
    args_send,options=options_send,callback=callbacks)

#save stopping criterion
with open(newfolder+"stopping_criterion.txt",mode='w') as
    file_object:
     print(es.stop(),file=file_object)

#save the es instance
```

```
    string=es.pickle_dumps()
    with open(newfolder+'saved_es.pkl','wb') as file:
        file.write(string)

    #save the datadict
    with open(newfolder+"datadict.txt",mode='w') as file_object:
        file_object.write(json.dumps(datadict))

    return x,es, int(newfolder[-3:-1])
```

Appendix C

Loss Functions

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit


def invdist_to_plateau(staircase,plateau):
    dists=np.abs(staircase-plateau)
    invdists=1-dists
    return np.fmax(0,invdists)

def make_gauss_fit(staircase,plateau,plot=False):
    y=invdist_to_plateau(staircase, plateau)
    x=np.arange(len(y))

    n = len(x)
    mean = sum(x*y)/n
    sigma = sum(y*(x-mean)**2)/n

    def gaus(x,a,x0,sigma):
        return a*np.exp(-(x-x0)**2/(2*sigma**2))

    popt,pcov = curve_fit(gaus,x,y,p0=[1,mean,sigma])
```

```
    if plot:
        plt.plot(x,y,'b+:',label='data')
        plt.plot(x,gaus(x,*popt),'ro:',label='fit')
        plt.legend()
        plt.title('Fit for plateau {}'.format(plateau))
        plt.xlabel('index')
        plt.ylabel('max(0,dist_to_plateau)')
        plt.text(1,0.8,"sigma={:.2f}".format(popt[2]))
        # plt.savefig('Optimization/Fit for plateau {}.png'.format(
            plateau))
        plt.show()
    return abs(popt[2])



def make_staircase_fit(staircase,plot=False):
    def staircase_func(h,w,a,x):
        return h*(1/2*np.cosh(a/2)/np.sinh(a/2)*np.tanh(a*((x/w-np.
            floor(x/w))-0.5)) + 1/2 + np.floor(x/w))

    def linear_func(a2,b,x):
        return a2*x+b

    def fit_func(x,height,width,a,a2,b,xs,xs2):
        if isinstance(x, np.ndarray):
            datalist=[]
            datalist.extend(np.zeros(len(x[x<xs])))
            datalist.extend(staircase_func(height,width,a,x[(xs<=x)
                & (x<xs2)]))
            datalist.extend(linear_func(a2,b,x[x>=xs2]))
            return datalist
```

```
        # elif isinstance(x,float):
        #     if x<xs:
        #         return staircase_func(height,width,a,x)
        #     else:
        #         return linear_func(a2,b,x)

    popt,pcov = curve_fit(fit_func,np.arange(len(staircase)),
       staircase,p0=[2.5,30,300,0.2,-15,25,175])
    if plot:
        fig,ax=plt.subplots()
        ax.plot(fit_func(np.arange(len(staircase)),*popt))
        ax.plot(staircase)
    return popt,pcov

class staircasiness():
    def __init__(self,delta=0.05,last_step=20,favorite=100):
        self.delta=delta
        self.bins=[]
        self.last_step=last_step
        if isinstance(favorite,int):
            favorite=[favorite]
        self.favorite=favorite
        for i in range(last_step):
            self.bins.extend([i+1-delta,1+i+delta])

        self.arange=np.arange(1,last_step)

    def histogram(self,staircase):

        test=np.histogram(staircase,self.bins)[0]
        multiplier=np.zeros(len(test))
```

```
        multiplier[range(0,len(multiplier),2)]=0.1
        score=sum(test*multiplier)+1

        return 1/score

    def gaussian_fit(self,staircase):
        score=1
        highest_plateau=int(np.floor(np.max(staircase)))
        for plateau in np.arange(1,highest_plateau):
            score+=make_gauss_fit(staircase,plateau,plot=True)
        return 1/score

    def deriv_metric_zeros1(self,staircase):
        res=0
        zero_count=0
        for i in range(len(staircase)-1):
            res += np.sqrt(np.abs(staircase[i+1]-staircase[i]))
            if staircase[i]<=1e-5: # added afterwards
                    zero_count+=1

        res /= np.sqrt(np.max(staircase)-np.min(staircase))
        res/=len(staircase)-zero_count
        return res

    def deriv_metric_cube_addsmall(self,staircase):
        res=0
        for i in range(len(staircase)-1):
            res += np.cbrt(np.abs(staircase[i+1]-staircase[i])+0.01)

        return res
```

```
    def stairLossFunk(self,staircase):
        Res=0
        for i in range(len(staircase)-1):
            Res+=np.cbrt(np.abs(staircase[i+1]-staircase[i])+0.01)

        P_zeros=len(np.where(staircase<1e-5)[0])/len(staircase)

        return Res*P_zeros

    def stairLossFunk2(self,staircase):
        res=0
        for i in range(len(staircase)-1):
            res+=np.cbrt(np.abs(staircase[i+1]-staircase[i])+0.01)

        P_not_zeros=len(np.where(staircase>1e-5)[0])/len(staircase)

        return res/(P_not_zeros+1)

    def deriv_metric_cube_addsmall_zeros(self,staircase):
        res=0
        # zero_count=0
        for i in range(len(staircase)-1):
            if staircase[i]<=1e-5: # added afterwards
                continue
            res += np.cbrt(np.abs(staircase[i+1]-staircase[i])+0.01)
        # res*=len(np.where(staircase<=1e-3)[0])/len(staircase)
        res/=np.cbrt(np.max(staircase))
        return res

    def deriv_metric_original(self,staircase):
        res=0
```

```
        for i in range(len(staircase)-1):
            res += np.sqrt(np.abs(staircase[i+1]-staircase[i]))

        res /= np.sqrt(np.max(staircase)-np.min(staircase))

        return res

    def deriv_metric_cube_zeros(self,staircase):
        res=0
        zero_count=0
        for i in range(len(staircase)-1):
            res += np.cbrt(np.abs(staircase[i+1]-staircase[i]))
            if staircase[i]<=1e-5: # added afterwards
                    zero_count+=1

        res /= np.sqrt(np.max(staircase)-np.min(staircase))
        res/=len(staircase)-zero_count
        return res

    def deriv_metric_cube_mask(self,staircase,maskvals=[0.1,20]):
        mask=np.where((maskvals[0]<=staircase) & (staircase<=
            maskvals[1]))[0]
        mstaircase=staircase[mask]
        res=0
        for i in range(len(mstaircase)-1):
            res += np.cbrt(np.abs(mstaircase[i+1]-mstaircase[i]))

        res /= np.sqrt(np.max(staircase)-np.min(staircase))

        return res
```

```
def window_loss(self,staircase,p=0.2, noise_eps=0):
    upper_lim=9
    lower_lim=1e-2
    if staircase[0]>upper_lim:
        return 2
    if (staircase<upper_lim).all() or (staircase>lower_lim).all
       ():
        return 2

    small = np.where(staircase < lower_lim)[0]
    large = np.where(staircase > upper_lim)[0]
    if small.shape[0] > 0:
        small = small[-1]
    else:
        small = 0

    if large.shape[0] > 0:
        large = large[0]
    else:
        large = staircase.shape[0]

    numel = large - small
    diff = (staircase[large - 1] - staircase[small])
    res = 0
    for i in range(small, large - 1):
        x = (staircase[i+1] - staircase[i])/diff
        res += np.abs(x+noise_eps/numel)**p #

    if numel==1:
        return 2
    else:
```

```
        return res * (1.0/(numel-1))**(1-p)


    def L_1_regularization(self,x,lamb):
        return lamb*np.sum(abs(np.array(x)))

    def L_2_regularization(self,x,lamb):
        return lamb*np.sum(np.array(x)**2)

    def step_loss(self,last_transmission,transmission):
        indexs=np.digitize(np.array([last_transmission,transmission
            ]),self.bins)
        # print(indexs)
        if indexs[0]%2==1:
            if indexs[1]==indexs[0]:
                # print("same")
                return abs(np.round(last_transmission)-transmission)


            elif indexs[1]>=indexs[0]:
                # print("above")
                return abs((np.round(last_transmission)+1-
                    transmission))


            else:
                # print("below")
                return 10
        else:
            ceil=np.ceil(last_transmission)
            if transmission>=(ceil+self.delta):
                return 10
            else:
```

```
            return abs(ceil-transmission)

    def window_histogram(self,staircase,linear_factor=0,p=3,plot=
        False,ax=None):
        if not ((staircase>1e-2) & (staircase<11)).any():
            return 1e4

        if not ((staircase<1e-2)).any():
            return 1e4

        if not ((staircase>11)).any():
            return 1e4

        mask=(staircase>1e-2) & (staircase<11)
        staircase=staircase[mask]

        num_bins=100

        hist,bins=np.histogram(staircase,num_bins,density=True)

        width=bins[1]-bins[0]

        loss=np.sum(abs(np.diff(hist*width)+linear_factor)**p)
        if plot:
            if ax!=None:
                bin_mids=[(bins[i]+bins[i+1])/2 for i in range(len(
                    bins)-1)]
                ax.plot(bin_mids,hist*width,label="%.3f"%loss)

        return 1/loss
```

```
    def multiple_windows_histogram(self,staircase,linear_factor=0,p
        =3,plot=False,ax=None):
        if not ((staircase>1e-2) & (staircase<11)).any():
            return 1e4

        if not ((staircase<1e-2)).any():
            return 1e4

        if not ((staircase>11)).any():
            return 1e4

        windows=np.arange(1,min(np.max(staircase),12),2)
        loss=0
        for i in range(len(windows)-1):

            mask=(staircase>=windows[i]) & (staircase<windows[i+1])
            if not mask.any():
                continue

            num_bins=20

            hist,bins=np.histogram(staircase[mask],num_bins,density=
                True)



            width=bins[1]-bins[0]

            loss+=np.sum(abs(np.diff(hist*width)+linear_factor)**p)

        # if plot:
        #     if ax!=None:
```

```
#             bin_mids=[(bins[i]+bins[i+1])/2 for i in range(len
    (bins)-1)]
#             ax.plot(bin_mids,hist*width,label="%.3f"%loss)

    return 1/loss
```

Appendix D

Example Run Device

```
import numpy as np

from lossfunctions.staircasiness import staircasiness
from datahandling.datahandling import datahandler,
    save_optimization_dict, load_optimization_dict
from optimization.fourier.fourier_modes_hardcoded import
    fourier_to_potential, plot_fourier_modes
from optimization.cma2 import optimize_cma, resume_cma
from optimization.newpoint import new_point

import matplotlib.pyplot as plt
import time



from triton7.pixel_sweep import sweep_gates
from optimization.newpoint import new_point, simple_new_point



def outer_gates_set(val):
    qdac.BNC13(val)
    qdac.BNC16(val)
    qdac.BNC17(val)
    qdac.BNC20(val)
```

```
    qdac.BNC6(val)
    qdac.BNC4(val)
    qdac.BNC1(val)
    qdac.BNC49(val)

outer_gates = qc.Parameter(name='outer_gates',label='outer gates
   pixel device',unit='v', set_cmd=outer_gates_set)

pixel_gates_list=[qdac.BNC12,
                  qdac.BNC15,
                  qdac.BNC5,
                  qdac.BNC18,
                  qdac.BNC48,
                  qdac.BNC3,
                  qdac.BNC2]

def parameter_pixels_set(val):
    qdac.BNC12(val)
    qdac.BNC15(val)
    qdac.BNC5(val)
    qdac.BNC18(val)
    qdac.BNC48(val)
    qdac.BNC3(val)
    qdac.BNC2(val)

parameter_pixels = qc.Parameter(name='BNC_12_15_5_18_48_3_2',label='
   BNC_12_15_5_18_48_3_2',unit='V', set_cmd=parameter_pixels_set)

def set_19_50(val):
    qdac.BNC19(val)
```

```
    qdac.BNC50(val)

gate_19_50 = qc.Parameter(name='BNC_19_and_50',label='BNC19,50',unit
    ='V', set_cmd=set_19_50)

def Conductance_get():
    voltage=lockin2.X()/100
    current=lockin3.X()*1e-7
    if current==0:
        return 0
    return 1/((voltage/current)/25.8125e3)

Conductance = qc.Parameter(name='g',label='Conductance',unit=r'$e^2/
    h$', get_cmd=Conductance_get)



#%%
bounds=(-1,0.3)
pfactor = 0.001

start = -1
stop = -1.8
points = 400
wait = 0.1


vals = np.linspace(start, stop, points)


stairs = staircasiness(delta=0.05, last_step=30)
```

```
dat = datahandler('BBQPC3')


def func_to_minimize(x,table): #x len 7

    voltages,penalty=simple_new_point(x,bounds) #new_point fixes the
        sum to 0, simple_new_point just sets values beyond bounds to
        the bounds


    voltages_send=vals

    #implement going to 0 and waiting
    gate_19_50(0)
    parameter_pixels(0)
    time.sleep(20)
    # set the pixels
    for i in range(len(pixel_gates_list)):
        pixel_gates_list[i](voltages[i])
    gate_19_50(start)
    time.sleep(10)



    result,dataid=sweep_gates([gate_19_50],voltages_send,wait,
       Conductance)

    #np.flip result here because we measure towards pinch off
```

```
    val=stairs.stairLossFunk2(np.flip(result))#+stairs.
        L_1_regularization(voltages, 0.001)+stairs.L_2_regularization
        (voltages, 0.001)

    key=table['next_key']
    table['next_key']+=1

    table['measurements'][key]={'loss':val+penalty*pfactor,'
        staircase':result,'x':x.tolist(),'voltages':voltages.tolist()
        ,'dataid':dataid,'deriv_metric':stairs.deriv_metric_zeros1(np
        .flip(result))}

    return val+penalty*pfactor


#%%

# set bias on 1 ohmic with qdac
qdac.BNC8(0.000125)
lockin2.amplitude(0.06) #40uV
outer_gates(-2)
time.sleep(10)

xbest,es,run_id=optimize_cma(func_to_minimize,dat,start_point=np.
    zeros(7),stop_time=18*3600,options={'tolx':1e-3})
```

Bibliography

[1] Fabio Ansaloni, Anasua Chatterjee, Heorhii Bohuslavskyi, Benoit Bertrand, Louis Hutin, Maud Vinet, and Ferdinand Kuemmeth. "Single-electron operations in a foundry-fabricated array of quantum dots". In: *Nature Communications* 11.1 (Dec. 2020). ISSN: 2041-1723. DOI: 10.1038/s41467-020-20280-3. URL: http://dx.doi.org/10.1038/s41467-020-20280-3.

[2] Waheb Bishara, Parsa Bonderson, Chetan Nayak, Kirill Shtengel, and J. K. Slingerland. "Interferometric signature of non-Abelian anyons". In: *Physical Review B* 80.15 (Oct. 2009). ISSN: 1550-235X. DOI: 10.1103/physrevb.80.155303. URL: http://dx.doi.org/10.1103/PhysRevB.80.155303.

[3] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York Inc., 2011. ISBN: 9780387310732.

[4] Bertram Brovang. "Multi-gate control of quantum-point-contact potentials in GaAs Hall bars". thesis. University of Copenhagen, 2021.

[5] Anasua Chatterjee, Fabio Ansaloni, Torbjørn Rasmussen, Bertram Brovang, Federico Fedele, Heorhii Bohuslavskyi, Oswin Krause, and Ferdinand Kuemmeth. *Autonomous estimation of high-dimensional Coulomb diamonds from sparse measurements*. 2021. arXiv: 2108.10656 [cond-mat.mes-hall].

[6] Sara M. Cronenwett. "Coherence, Charging, and Spin Effects in Quantum Dots and Quantum Point Contacts". dissertation. Stanford University, 2001. URL: https://qdev.nbi.ku.dk/student_theses/marcuslabthese/.

[7] Supriyo Datta. *Electronic Transport in Mesoscopic Systems*. Cambridge University Press, 1995. ISBN: 9780511805776.

[8] John H. Davies, Ivan A. Larkin, and E. V. Sukhorukov. "Modeling the patterned two-dimensional electron gas: Electrostatics". In: *Journal of Applied Physics* 77.9 (1995), pp. 4504–

4512. DOI: 10.1063/1.359446. eprint: https://doi.org/10.1063/1.359446. URL: https://doi.org/10.1063/1.359446.

[9] John C. Davis. *Statistics and Data Analysis in Geology*. Wiley, 2002. ISBN: 978-0-471-17275-8.

[10] Jingyu Duan, Michael A. Fogarty, James Williams, Louis Hutin, Maud Vinet, and John J. L. Morton. "Remote Capacitive Sensing in Two-Dimensional Quantum-Dot Arrays". In: *Nano Letters* 20.10 (2020). PMID: 32946244, pp. 7123–7128. DOI: 10.1021/acs.nanolett.0c02393. eprint: https://doi.org/10.1021/acs.nanolett.0c02393. URL: https://doi.org/10.1021/acs.nanolett.0c02393.

[11] Federico Fedele, Anasua Chatterjee, Saeed Fallahi, Geoffrey C. Gardner, Michael J. Manfra, and Ferdinand Kuemmeth. "Simultaneous Operations in a Two-Dimensional Array of Singlet-Triplet Qubits". In: *PRX Quantum* 2.4 (Oct. 2021). ISSN: 2691-3399. DOI: 10.1103/prxquantum.2.040306. URL: http://dx.doi.org/10.1103/PRXQuantum.2.040306.

[12] Christoph W Groth, Michael Wimmer, Anton R Akhmerov, and Xavier Waintal. "Kwant: a software package for quantum transport". In: 16.6 (June 2014), p. 063065. DOI: 10.1088/1367-2630/16/6/063065. URL: https://doi.org/10.1088/1367-2630/16/6/063065.

[13] B. I. Halperin. "Quantized Hall conductance, current-carrying edge states, and the existence of extended states in a two-dimensional disordered potential". In: *Phys. Rev. B* 25 (4 Feb. 1982), pp. 2185–2190. DOI: 10.1103/PhysRevB.25.2185. URL: https://link.aps.org/doi/10.1103/PhysRevB.25.2185.

[14] Nikolaus Hansen. *The CMA Evolution Strategy: A Tutorial*. 2016. arXiv: 1604.00772 [cs.LG].

[15] Yu He, Samuel Gorman, Daniel Keith, Ludwik Kranz, Joris Keizer, and Michelle Simmons. "A two-qubit gate between phosphorus donor electrons in silicon". In: *Nature* 571 (July 2019), p. 371. DOI: 10.1038/s41586-019-1381-2.

[16] Thomas Ihn. *Semiconductor Nanostructures*. Oxford University Press, 2009. ISBN: 9780199534432.

[17] Douglas Templeton Mcclure III. "Interferometer-Based Studies of Quantum Hall Phenomena". dissertation. Harvard University, 2012.

[18] J. K. Jain. "Composite-fermion approach for the fractional quantum Hall effect". In: *Phys. Rev. Lett.* 63 (2 July 1989), pp. 199–202. DOI: 10.1103/PhysRevLett.63.199. URL: https://link.aps.org/doi/10.1103/PhysRevLett.63.199.

[19] J. K. Jain. "Theory of the fractional quantum Hall effect". In: *Phys. Rev. B* 41 (11 Apr. 1990), pp. 7653–7665. DOI: 10.1103/PhysRevB.41.7653. URL: https://link.aps.org/doi/10.1103/PhysRevB.41.7653.

[20] Klaus Von Klitzing. *The Quantized Hall Effect*. 1985. URL: https://www.nobelprize.org/prizes/physics/1985/klitzing/lecture/.

[21] Oswin Krause, Torbjørn Rasmussen, Bertram Brovang, Anasua Chatterjee, and Ferdinand Kuemmeth. *Estimation of Convex Polytopes for Automatic Discovery of Charge State Transitions in Quantum Dot Arrays*. 2021. arXiv: 2108.09133 [cs.LG].

[22] A. R. Mills, D. M. Zajac, M. J. Gullans, F. J. Schupp, T. M. Hazard, and J. R. Petta. "Shuttling a single charge across a one-dimensional array of silicon quantum dots". In: *Nature Communications* 10.1 (Mar. 2019). ISSN: 2041-1723. DOI: 10.1038/s41467-019-08970-z. URL: http://dx.doi.org/10.1038/s41467-019-08970-z.

[23] H. Moon et al. "Machine learning enables completely automatic tuning of a quantum device faster than human experts". In: *Nature Communications* 11.1 (Aug. 2020). ISSN: 2041-1723. DOI: 10.1038/s41467-020-17835-9. URL: http://dx.doi.org/10.1038/s41467-020-17835-9.

[24] Gregory W. Moore and Nicholas Read. "Nonabelions in the fractional quantum Hall effect". In: *Nuclear Physics* 360 (1991), pp. 362–396.

[25] Juha T. Muhonen et al. "Storing quantum information for 30 seconds in a nanoelectronic device". In: *Nature Nanotechnology* 9.12 (Oct. 2014), pp. 986–991. ISSN: 1748-3395. DOI: 10.1038/nnano.2014.211. URL: http://dx.doi.org/10.1038/nnano.2014.211.

[26] V. Nguyen et al. "Deep reinforcement learning for efficient measurement of quantum devices". In: *npj Quantum Information* 7.1 (June 2021). ISSN: 2056-6387. DOI: 10.1038/s41534-021-00434-x. URL: http://dx.doi.org/10.1038/s41534-021-00434-x.

[27] J Petta, AC Johnson, Jacob Taylor, EA Laird, Amir Yacoby, M Lukin, Charles Marcus, Maxwell Hanson, and AC Gossard. "Applied physics: Coherent manipulation of coupled electron spins in semiconductor quantum dots". In: *Science (New York, N.Y.)* 309 (Oct. 2005), pp. 2180–4. DOI: 10.1126/science.1116955.

[28] Christian Volk, Anasua Chatterjee, Fabio Ansaloni, Charles M. Marcus, and Ferdinand Kuemmeth. "Fast Charge Sensing of Si/SiGe Quantum Dots via a High-Frequency Accumulation Gate". In: *Nano Letters* 19.8 (July 2019), pp. 5628–5633. ISSN: 1530-6992. DOI: 10.1021/acs.nanolett.9b02149. URL: http://dx.doi.org/10.1021/acs.nanolett.9b02149.

[29] W. G. van der Wiel, S. De Franceschi, J. M. Elzerman, T. Fujisawa, S. Tarucha, and L. P. Kouwenhoven. "Electron transport through double quantum dots". In: *Rev. Mod. Phys.* 75 (1 Dec. 2002), pp. 1–22. DOI: 10.1103/RevModPhys.75.1. URL: https://link.aps.org/doi/10.1103/RevModPhys.75.1.

[30] Jun Yoneda et al. "A quantum-dot spin qubit with coherence limited by charge noise and fidelity higher than 99.9%". In: *Nature Nanotechnology* 13.2 (Dec. 2017), pp. 102–106. ISSN: 1748-3395. DOI: 10.1038/s41565-017-0014-x. URL: http://dx.doi.org/10.1038/s41565-017-0014-x.