Predictive modeling of nanoelectronic devices

Antonio Lacerda Santos Neto

▶ **To cite this version:**

Antonio Lacerda Santos Neto. Predictive modeling of nanoelectronic devices. Physics [physics]. Université Grenoble Alpes [2020-..], 2024. English. NNT : 2024GRALY022 . tel-04718517

HAL Id: tel-04718517
https://theses.hal.science/tel-04718517v1

Submitted on 2 Oct 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : PHYS - Physique
Spécialité : Physique Théorique
Unité de recherche : PHotonique, ELectronique et Ingéniérie QuantiqueS

Modélisation prédictive des dispositifs nanoélectroniques

Predictive modeling of nanoelectronic devices

Présentée par :

Antonio LACERDA SANTOS NETO

Direction de thèse :

Xavier WAINTAL DIRECTEUR DE RECHERCHE, CEA DE GRENOBLE	Directeur de thèse
Christoph GROTH INGENIEUR CHERCHEUR, Université Grenoble Alpes	Co-encadrant de thèse

Rapporteurs :

ANTON AKHMEROV
ASSOCIATE PROFESSOR, DELFT UNIVERSITY OF TECHNOLOGY
FABIENNE MICHELINI
PROFESSEURE DES UNIVERSITES, AIX-MARSEILLE UNIVERSITE

Thèse soutenue publiquement le **28 juin 2024**, devant le jury composé de :

RODOLFO JALABERT, PROFESSEUR DES UNIVERSITES, UNIVERSITE DE STRASBOURG	Président
ANTON AKHMEROV, ASSOCIATE PROFESSOR, DELFT UNIVERSITY OF TECHNOLOGY	Rapporteur
FABIENNE MICHELINI, PROFESSEURE DES UNIVERSITES, AIX-MARSEILLE UNIVERSITE	Rapporteure
HERMANN SELLIER, MAITRE DE CONFERENCES HDR, UNIVERSITE GRENOBLE ALPES	Examinateur

GRENOBLE ALPES UNIVERSITY

DOCTORAL SCHOOL OF PHYSICS

P H D T H E S I S

to obtain the title of

PhD of Science

of the Grenoble Alpes University

Specialty : Theoretical Physics

Defended by

Antonio Lacerda Santos Neto

Predictive modeling of nanoelectronic devices

Thesis Advisor: Xavier Waintal and Christoph Groth

prepared at the Quantum Photonics, Electronics and Engineering Laboratory (PHELIQS) - Theory Group

defended on the 28 of June 2024

Jury :

President :	Rodolfo Jalabert	-	Université de Strasbourg
Reviewers :	Fabienne Michelini	-	Aix-Marseille Université
	Anton Akhmerov	-	Delft University of Technology
Examinator :	Hermann Sellier	-	Université de Grenoble

Acknowledgments

This work would not have been made possible without the support of the many amazing people that went through my life.

First, to my two good friends Loman and Flora, who have always been there for me whenever I needed the most. To my aunt Silvia, a beautiful person whose words have always comforted me. I would not have been able to finish this thesis without the support of the three of you. I am also thankful for the support my family has given me all these years since I left Brasil.

To the first person who mentored me at Pheliqs, François Lefloch, thank you for your support during all these years at CEA. To Christoph Groth whom I own much of what I know about programming. And, to Xavier, who patiently guided me all theses years and gave me the freedom I needed - not only to learn how to research, but also find out what I was searching for [1]. You have taught me much more than physics.

I would like to thank all of those who collaborated with me during this Thesis, among them : Gaëtan, Dietmar, Boris, Thomas Kloss, Eleni, Christopher. To the interns who worked with me, namely Matthias Flor, Thomas Hellemans, Ahmad Fouad Kalo, Paul Aubriot, Ravel Nantenaina – thank you all for the help and feedback on *PESCADO* .

I quite enjoyed my years at Pheliqs, and this is thanks to all the students, post-docs researchers and technicians that I've had the pleasure to encounter. First is Pacome Armagnat, whom I met at my first internship at the Theory Group. Thank you for all the discussions inside and outside the laboratory. I would like also to mention Romaine and the discussions about feminism, veganism and techno haha. It was nice to have met you. Then there is Anthony, an interesting and talented person - it was definitely a pleasure to have worked on my PhD along side you during these years. Of course, Yoan and Thomas for the many (many ...) passionate discussions - and Romy who came a bit later but it did not take much time before we started those half-hour discussions about, e.g. the definition of dictatorship. Corentin, one very bright person with whom I enjoyed snowboarding (and talking about physics, mais cela va de soi), see you soon at Île-de-France. Matthieu as well, it was a pleasure sharing MESO and Nano with you all these years. I cannot finish before mentioning Marjan (hopefully you are doing well, and cycling a lot), Yuriel, Alessandro (I still owe you a plant ...), Anas, Anna and Prasoon. I'm getting out of ideas to start new sentences, but I have to thank Marielle, a wonderful person and very reliable secretary ! Be well.

[1] Before I get lost again, but that is just normal

Contents

List of Acronyms

SCQE Self-Consistent Quantum Electrostatics . . . 4

ILDOS Integrated local density of states . . . 11

QPC Quantum Point Contact . . . 2

2DEG two-dimensional electron gas . . . 2

MZI Mach-Zehnder interferometers . . . 2

TF Thomas-Fermi . . . 12

NLH Non Linear Helmholtz . . . 12

LH Linear Helmholtz . . . 21

LDOS Local Density of States . . . 11

DOS Density of States . . . 11

QHE Quantum Hall Effect . . . 4

QH Quantum Hall . . . 2

CSG Chklovskii, Shklovskii and Glazman . . . 3

LB Landauer-Büttiker . . . 4

PESCA Pure Electrostatic Self consistent Approximation . . . 13

SGM Scanning Gate Microscopy . . . 23

Chapter 1

Introduction

"L'humain est la nature prenant conscience d'elle-même " [1]

[Élisée Reclus (1830-1905) Geographe 1905]

Philosophical preliminaries

Nature is one, and as such we are an intrinsic part of it. Through physics we can understand how we are connected to the world that surrounds us. Understating nature is a two step process, differentiating and unifying. First we differentiate nature into isolated objects, moving as if they were disconnected from the whole. Then we weave them together in an effort to best describe what our senses detect and our subjective mind deduces. In that sense, mathematics is a language, a grammar, made for us to communicate with one another. In that same way a theoretician is much like a poet, where each verse follows the rules dictated by mathematics. In that logic, numerics is nothing more than illustrating that poem through an image. Let it be to magnify a certain aspect of the poem or to unify it into something our senses are more familiarized with. To be a great poet we need to write about what we have lived. We need to interact with nature, fell it and then write about it. And then interact again, confront what we have written and correct ourselves as we iterate. As we iterate, we calibrate our brain into better correlating what our senses detect to what our subjective mind interprets. And that is what I think to be so beautiful about physics, it helps us calibrate ourselves such that we become, with each iteration, closer to the nature surrounding us. The purpose of this thesis is to investigate how to visualize the environment electrons live in using numerics.

General Introduction

In 1948, when John Bardeen and Walter Brattain realized the first (point-contact) transistor [Bardeen & Brattain 1948], they made one of the first tools for us to control with precision the electrostatic environment where electrons propagate. It consisted of a simple - and large - superposition of P-N doped germanium sandwiched between two metals. The precision at which we modified the electrostatic

[1] The original sentence is "L'homme est la nature prenant conscience d'elle-même". However I decided to actualize it.

environment however was still too rough for the electrons to sense it individually; hence only their macroscopic behavior was modified. The increase in technical proficiency of the academic community allowed us to downsize the initial devices from the cm scale (e.g. MOS transistor [Atalla *et al.* 1959]) to μm and nm scale. By reducing the noise from defects and impurities, by reducing the size of the metallic gates and by reducing their distances from the conducting region, we increased the precision with which we alter the electrostatic environment of the electrons - to the point it is of the order of their wavelength. An example of a typical device is the quantum point contact: two gates placed a distance of tens of nanometers from each other, and a few hundreds of nanometers above the propagating electrons. By decreasing the gate voltage we can create a electrostatic potential barrier that constricts the electron flow through a valley that is on the order of the electronic wavelength. Hence by measuring the electronic flow after the constriction, we see the quantum behavior of electrons. Moreover, as the electrons leave the constriction, they are characterized by a quantum phase. However, material defects and impurities altering the electrostatic environment of the electrons in addition to ourselves also scatter the electrons, thus changing the quantum phase and the electronic trajectory. To quantify this we can define a quantum coherence length, upon which the electronic wave looses its phase coherence. Systems whose coherence length is smaller than the active region of the device are considered ballistic, i.e. can be treated by single-particle quantum models. This length can go from μm to nm depending on the material the electrons propagate. Therefore, there are two key quantities characterizing a quantum device, the carrier wavelength (concerning manipulation) and coherence length (concerning measurement).

Historically, in 1988 two independent groups measured conductance quantization due to confinement in GaAs-AlGaAs hetero-junctions [van Wees *et al.* 1988, Wharam *et al.* 1988]. They were the first to be able to measure ballistic transport in nanoelectronic devices due to the high mobility of the electrons in the two-dimensional electron gas (2DEG), see e.g. [Pfeiffer *et al.* 1989, Shayegan *et al.* 1988a]. This meant we could use microscopic semiconducting devices to measure the quantum states we manipulate. Coupled to advances in cryogenics and metrology, nanoelectronics became a miniaturized laboratory for probing the quantum behavior of matter. Formally we call it mesoscopic physics. Fast forward to today, we are not only exploring the quantum helm, but we have also started to manipulate with great precision its different elements. As examples we can cite the control of quantum confinement (Quantum Point Contact (QPC)s [van Wees *et al.* 1988, Topinka *et al.* 2000, van Houten & Beenakker 1996, Bauer *et al.* 2013]), coherent transport of single electrons [Bäuerle *et al.* 2018, Duprez *et al.* 2019] and controlled interferometry of coherent quantum states (Mach-Zehnder interferometers (MZI) [Wei *et al.* 2017, Jo *et al.* 2021, Ji *et al.* 2003]). They have all become building blocks for technological applications (quantum bits [Bautze *et al.* 2014, Edlbauer *et al.* 2022, Yamamoto *et al.* 2012, Koch *et al.* 2007]) or for more complex experiments (probing fractional Quantum Hall (QH) quasiparticles [Saminadayar *et al.* 1997, Shayegan *et al.* 1988b, Nakamura *et al.* 2020,

Chang 1990, Carrega *et al.* 2021, Bolotin *et al.* 2009, Bhattacharyya *et al.* 2019]).

Although we have managed, through mesoscopic physics, to experimentally bridge the gap between classical electrostatics (energy scales of eV), e.g. gate field effect electrostatics, and that of quantum particles (meV to μeV), the theoretical tools used to model them are - for the most part - disconnected. This large difference in energy scales is what renders difficult unifying models capturing the electrostatic environment of the device to those aimed at studying the quantum effects we seek to understand. However, this large difference in energy scales is also what renders the quantum mechanical effects we study extremely sensible to small variations of their electrostatic environment [Martinez & Niquet 2022, Percebois & Weinmann 2021]. Usually what is done is to represent the electrostatic environment of a device with an effective potential added to the hamiltonian [Bautze *et al.* 2014]. In part, this effective potential absorbs all complicated features semiconducting devices have. However, they are unable to relate the experimental voltages applied at the gates to the measured quantum transport quantities. They also ignore details of the device geometry and interactions with its surrounding environment. Already this renders predictive simulations of nanoelectronic devices impossible. More importantly, using an effective potential can be a fatal limitation to the qualitative simulation of quantum transport. In 1993 Chklovskii, Shklovskii and Glazman (CSG) showed sometimes the electrostatic environment must be treated quantitatively for the models to capture quantum behavior qualitatively [Chklovskii *et al.* 1992a, Chklovskii *et al.* 1992b, Chklovskii *et al.* 1993]. In this thesis we have also shown that effective potential models can misguide researchers and lead to a wrong physical interpretation of experimental results [Flór *et al.* 2022]. Therefore, as the field of mesoscopic physics progresses, the need for a tool capable of predicting quantum features from quantitative models of the device electrostatic environment becomes pressing.

Within this context :

> The objective of this thesis is to develop a theoretical model, numerical algorithm and software capable of calculating the electrostatic control we exert over the charge carriers in mesoscopic devices so that we can predict their behavior.

This implies correlating how the eV energies we fix at the device gates affect the meV electrostatic landscape where the carriers propagate. It also means capturing the meV to μeV electrostatic screening the charge carriers exert on the electrostatic environment. This requires :

I) Modeling the mesoscopic device electrostatic environment;

II) Capturing the interaction between the electrostatic environment and the electron quantum mechanics.

These are two very distinct tasks. The first task requires precise understanding of the materials science behind mesoscopic devices and a pragmatic knowledge of

their measurable effects on quantum transport. It also requires working closely with experimental research groups to get a regular feedback on our model. The second task requires accounting for the self consistent effect the charge carriers have on their electrostatic environment and the electrostatic environment has on the charge carriers. It can be achieved by accounting for the electrostatic screening of the charge carriers in the mean field approximation. This implies solving a highly non-linear self-consistent problem formulated by coupling the Schrödinger equation to the Poisson equation.

The first part of this Introduction illustrates why it is fundamental to quantitatively model the electrostatic environment where the charge carriers propagate. We do so using the Quantum Hall Effect (QHE) as an example. The second part formulates the Self-Consistent Quantum Electrostatics (SCQE) problem, review the existing methods to solve it and explain where they fail. The third part is semiconductor materials science. We list the different microscopic effects found in semiconducting mesoscopic devices and explain how they contribute to quantum transport experiments. The fourth explain in detail the purpose of this thesis and summarizes the main results. Last is the outline.

1.1 The importance of electrostatics at a quantitative level

Most quantum transport simulations are limited to a qualitative view of the device's electrostatic environment. For example, suppose we want to calculate the conductance through a 2DEG in a heterostructure over which we depose metallic gates. Figure 1.1 (a) shows a simplified schematic for a device invariant on $\vec{y}$ and whose 2DEG is formed at a AlGaAs/GaAs heterostructure. To calculate the conductance defined in, e.g. the Landauer-Büttiker (LB) formalism, one solves the non-interacting Schrodinger equation under the effective mass approximation at the 2DEG:

$$\frac{1}{2m^*}(i\hbar\vec{\nabla} + e\vec{A})^2\Psi(x,y) - eU(x)\Psi(x,y) = E\Psi(x,y) \tag{1.1}$$

with $m^* = 0.067m_e$ the effective mass for GaAs [Lawaetz 1971] and e the electron charge. The effect of the gates and the device microscopic details, such as dopants and interface charges, are captured by the potential term $U(x)$. Here we consider the potential invariant on $\vec{z}$ due to the 2DEG small dimension ($\approx 10nm$) compared to the distance of the 2DEG from the gates, dopants and device interfaces.

Often $U(x)$ is an effective potential, a function chosen so that the solution of Eq.(1.1) best captures the experimental features one seeks to explain. For instance, one can take analytical solutions to $U(x)$, e.g. a saddle point potential of the form $U(x,y) = V_0 - 1/2\omega_x^2x^2 + 1/2\omega_y^2y^2$ whose ω_x and ω_y values are calibrated to best describe the experimental results. The problem with assuming the shape of the potential is that first it can not give a quantitative result and second it can trick us into thinking we have a correct understanding of the physics taking place.

To get an intuition to what is wrong with this approach think of the energy scales in place. The electrostatic landscape where the charges propagate are on the order of tens - hundreds of meV. The quantum oscillations we want to understand are of the order of tens - hundreds of μeV. To be sure we are looking at the right place in the Hilbert space it is important to first capture correctly the meV physics. This requires a quantitative knowledge of the nanoelectronic device electrostatics. To get a quantitative knowledge we need know where are the charges in the device. Changing the number of electrons in a system is extremely costly energetically, hence if we can calculate the charge dispersion at the meV precision, then calculating the charge dispersion at the μeV precision is only a matter of changing the result very slightly. In contrast the shape of the potential can change quite a lot. Hence if we simply assume a shape of the electrical potential, the corresponding charge dispersion can be quite far from the correct solution.

In the next section we shall illustrate our argument by comparing the difference in the physical interpretation of the QHE obtained under a qualitative treatment of the device electrostatics versus a quantitative treatment.

1.1.1 Case study : the quantum hall effect

In this section we shall compare the qualitative treatment of electrostatics of the LB approach [Büttiker 1988] to the quantitative treatment of CSG [Chklovskii *et al.* 1992a]. In doing so we shall elucidate how the results obtained with the two pictures differs as we go from a low magnetic field QHE regime to a high magnetic field regime, see [Armagnat & Waintal 2020].

As a toy model, we take the device on Fig 1.1. It is close enough to experimental devices to illustrate some interesting QH physics, and simple enough to be easily solved. The gates (in gray) apply a confining potential $U(x)$ at the 2DEG, depleting the charge beneath them. A magnetic field $\vec{B} = B\vec{u}_z$, perpendicular to the 2DEG, is also applied. We disregard the spatial extension of the 2DEG in $\vec{z}$.

Lets start by calculating the energy spectrum and electronic states at the two dimensional electron gas (2DEG) by solving Eq.(1.1). A general solution for Eq.(1.1) takes the form of plane waves along the y direction s.t.

$$\Psi(x, y) = e^{iky}\psi(x), \tag{1.2}$$

with momentum k. To find $\psi(x)$ we need to specify a gauge for the vector potential s.t. $\vec{B} = \vec{\nabla} \times \vec{A}$. Under the Landau gauge - $\vec{A} = Bx\vec{u_y}$, Eq.(1.1) writes:

$$\psi''(x) + \frac{2m^*}{\hbar^2}[(E - eU(x)) - \frac{1}{2}m^*\omega_c(x - x_k)^2]\psi(x) = 0 \tag{1.3}$$

with $\omega_c = (eB)/m^*$ the cyclotron frequency, $x_k = kl_b^2$ and $l_b = \sqrt{\hbar/(eB)}$ the magnetic length.

For $U(x) = 0$ this is the equation for a harmonic oscillator, oscillating at ω_c around the center coordinate x_k. The energy spectrum is called Landau levels [Landau & Lifshitz 1981]: degenerated energy levels equally spaced following $E_n =$

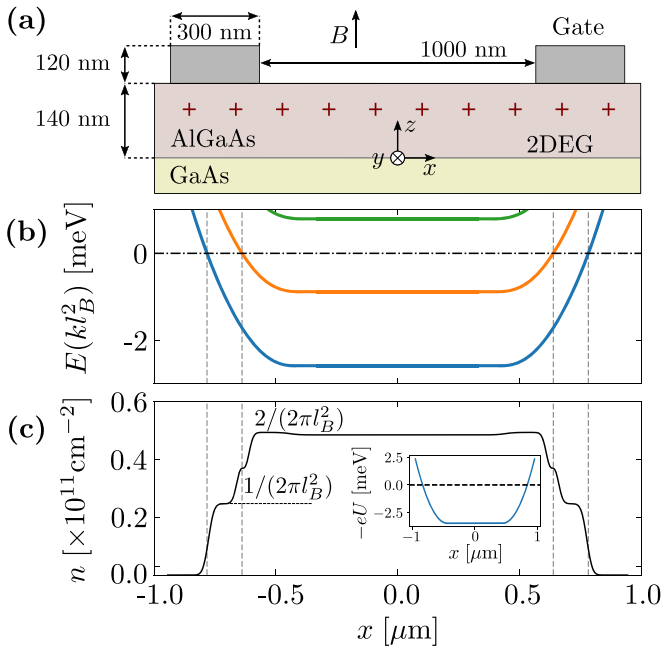


Figure 1.1: Taken from [Armagnat & Waintal 2020] - a) 2D cut along $x - z$ of a typical (simplified) nanoelectronic device invariant in $\vec{y}$. The 2DEG is located at the GaAs side of the heterostructure. The plus sign in red indicate the ionized donor dopants at the AlGaAs layer. b) Non interacting energy dispersion at the 2DEG for the effective potential on the inset of (c) and constant magnetic field $\vec{B} = B.\vec{u}_z$, with $B = 1T$. The fermi level is $E_F = 0$ and $x = kl_b^2$, with l_b the magnetic length and k the momentum (see text). c) Charge density at the 2DEG for the energy dispersion in (b). Both (b) and (c) have been calculated by [Armagnat & Waintal 2020] using Eq.(1.1).

$\hbar\omega_c(n+1/2)$, whose states are exponentially localized in x at around kl_b^2. This is the bulk QH physics.

In the next two sections we consider the case where $U(x)$ is not zero. In the first we use the LB approach. In the second we use the CSG approach. We follow the works of [Büttiker 1988, Chklovskii *et al.* 1992a, Chklovskii *et al.* 1992b, Chklovskii *et al.* 1993, Armagnat & Waintal 2020].

1.1.1.1 Quantum hall physics under Landaeur Buttiker and Halperin

In the LB picture $U(x)$ is an external input of the problem [Büttiker 1986, Halperin 1982]. For instance, it can be an analytical function, e.g. $U(x) = U_0 + ax^2$ for a constant. If $U(x)$ is slowly varying, we can approximate the spectrum following:

$$E_n(k) \approx \hbar\omega_c(n+1/2) - eU(x_k) \tag{1.4}$$

Figure 1.1 (b) shows a numerically calculated energy spectrum for the device on (a) with an external potential $U(x)$ (inset on (c)). In the calculations $E_F = 0$, hence $E_n(k) = 0$ gives the conducting channels. They are centered at around x_k, with propagating velocity $v_k = (1/\hbar)dE/dk$. Hence they are located at the edge, with positive v_k for the channels on the right and negative for those at the left.

Up to here the LB picture seems to capture some essential features of the QHE, notably the chiral edge states. However, the charge profile at Figure 1.1 (c) reveals a profound flaw of the LB approach. In the calculations shown in Figure 1.1 (b) - (c) there are two filled Landau levels. The total number of available states in a Landau level is $1/(2\pi l_b^2)$, hence the bulk electron density is $n = 2eB/h$. As we go to the edge of the sample, the number of filled Landau levels decreases to one, s.t. $n = eB/h$, and then to zero when it reaches the edge. This means the density at the sample is set by the magnetic field B, which is unreasonable. Indeed, in the LB picture the density profile under magnetic field field changes considerably compared to density profile for $B = 0$. Such large displacement of charges has a high potential energy cost, which can not be explained by the change in magnetic energy, $\hbar\omega_c$ - typically of the order of $10meV$. In practice the charge density profile changes very little as the magnetic field is added to the system, the LB picture is wrong. Furthermore, a large displacement of charges is associated with a large change of the electrostatic potential surrounding it. Therefore even if there was a large displacement of charges, there is no valid reason to consider the $U(x)$ invariant under variations of the magnetic field. This leads one to conclude the main flaw of the LB picture is to consider $U(x)$ a fixed input of the problem.

1.1.1.2 Quantum hall physics under Chklovskii, Shklovskii and Glazman

In 1992 CSG presented a series of papers revisiting the LB picture [Chklovskii *et al.* 1992a, Chklovskii *et al.* 1992b, Chklovskii *et al.* 1993]. From the energetic argument presented in Section 1.1.1.1, they argued it is the charge profile that should slowly change with the magnetic field. Therefore, they first calculate the

charge profile at $B = 0$, namely $n(x, B = 0)$. Under the assumption the magnetic field will only slightly alter $n(x, B = 0)$, they derive $n(x, B \neq 0)$ and the energy dispersion. In this section we shall follow their method to derive the charge profile and energy dispersion at $B = 3.7T$ for the device in Fig.1.1 (a).

First we calculate the charge density and the energy dispersion at $B = 0T$, c.f. top of Fig.1.2. To do so we solve the SCQE problem under the Thomas-Fermi approximation using the *PESCADO* software, c.f. Chapter 4. For the 2DEG density of states, we use the bulk value. We choose a dopant concentration such that the charge density at the 2DEG is $n = 2.02.10^{11} cm^2$ at zero gate voltage. We apply a gate voltage $V_g = -0.05V$ for the calculations shown on the top of Fig.1.2. Second, we calculate the Landau level charge density for $B = 3.7T$, $n_{LL} = eB/h$. With this we can find the charge density positions of the Landau level plateaus at the $n(x, B = 0)$ profile. This is the Step I shown in Fig.1.2. From the charge profile at the top of Fig.1.2 we deduce that under $B = 3.7T$ there can be two filled Landau levels, $\nu = 1$ and $\nu = 2$ with ν the filling factor of Landau levels. The gray dotted lines shown where $n(x, B = 0)$ crosses $n = \nu eB/h$. The position x where they cross marks the central position of the edge state, namely x_ν. This concludes the first Step. Here we can already deduce that the charge profile at $n(x, B \neq 0)$ will be divided into regions of varying charge density and constant charge density, called respectively compressible and incompressible stripes. At the compressible stripes the potential is constant and at the incompressible stripes the potential varies.

We need now to extract the size of incompressible stripes, noted a_ν. To do so we will use an energetic argument. First, we can relate x_ν to the electron momentum k using the relation $x = kl_b^2$. Therefore, between the two edges of an incompressible stripe there is a potential energy drop of $\hbar\omega_c$. This change in potential energy is accompanied by a charge displacement :

$$\delta n = a_\nu \frac{dn}{dx}|_{x=x_\nu} \tag{1.5}$$

where the derivative is taken with respect to the charge profile for $B = 0T$ and assumed constant within $x \in [x_\nu - a_\nu/2, x_\nu + a_\nu/2]$. To find the energy variation associated with displacing δn we can model the incompressible stripe as a 2D capacitor composed of two plates placed next to each other and separated by a distance of a_ν, s.t. :

$$a_\nu \frac{dn}{dx}|_{x=x_\nu} = c\frac{\varepsilon}{e a_\nu}\delta U \tag{1.6}$$

with the geometrical constant for the device in question $c = 5.1$, c.f. [Armagnat *et al.* 2019]. Replacing $e\delta U$ with $\hbar\omega_c$ we obtain :

$$a_\nu = \sqrt{\frac{\hbar\omega_c}{\frac{dn}{dx}|_{x=x_\nu}}\frac{c\varepsilon}{e^2}} \tag{1.7}$$

Using Eq.(1.7) we can place the charge plateaus of size a_ν in $n(x, B = 0)$, c.f. Step II of Fig.1.2. With this we can sketch $n(x, B \neq 0)$ and the energy dispersion,

c.f. black profile in bottom of Fig.1.2. In light grey is shown the charge profile for $B = 0T$.

The CSG approach is valid when the spatial spread of the Landau level is much smaller than the size of the incompressible stripe, i.e. $l_b \ll a_\nu$. This is only true for high magnetic fields. For intermediary and low values of magnetic field the CSG approach looses accuracy. Recently [Armagnat & Waintal 2020] performed full self consistent quantum electrostatics calculations for the toy model of Fig.1.1 (a). They showed that the qualitative difference between CSG (thomas fermi) and a full self consistent calculation is the absence of charge plateaus in the incompressible stripes. This is indeed due to the spread of the Landau levels. At the high magnetic field limit they recovered the CSG results. At low values of magnetic field the LB approach of spatially localized propagating channels is recovered.

1.2 The Self Consistent Quantum - Electrostatics problem

The SCQE problem, also known as Schrödinger-Poisson, describes moving charged quantum particles coupled to the classical electrostatic environment that they generate. It is that of many independent (non-interacting) particles propagating together in the same electrostatic landscape. While they propagate, they affect their landscape, as much as the landscape affects the particles themselves - and their density. Although we assume the particles do not interact directly with themselves, they fell each other through their impact in the electrostatic landscape surrounding them. In the SCQE problem, the two quantities we seek to calculate are the quantum charge density $n(\vec{r})$ and the classical electrostatic potential - $U(\vec{r})$. The SCQE problem is formulated by a set of three equations, relating $n(\vec{r})$ calculated with the Schrodinger equation to $U(\vec{r})$ solution of the Poisson equation. Formally, it accounts for the effect of coulomb interaction at the mean field level of approximation - the Hartree term of the Hartree-Fock method. It provides a sound foundation from which many-body models can be constructed. In what follows we shall formulate the SCQE problem and discuss the approach taken in this thesis to solve it.

1.2.1 Problem definition

First comes the Schrödinger equation :

$$[H_0 - eU]\Psi_{\alpha E} = E\Psi_{\alpha E} \tag{1.8}$$

where H_0 is the system Hamiltonian, $U(\vec{r})$ the electrostatic potential seen by the carriers (that we suppose to be electrons with negative charges without loss of generality), $e > 0$ the elementary charge, $\Psi_{\alpha E}$ the electronic wave function at energy E and sub-band α. In the simplest situation $H_0 = p^2/(2m^*)$ is the effective mass Hamiltonian but it can also account for orbital degrees of freedom and/or spins, superconductivity etc. We suppose that the system is in the thermodynamic limit

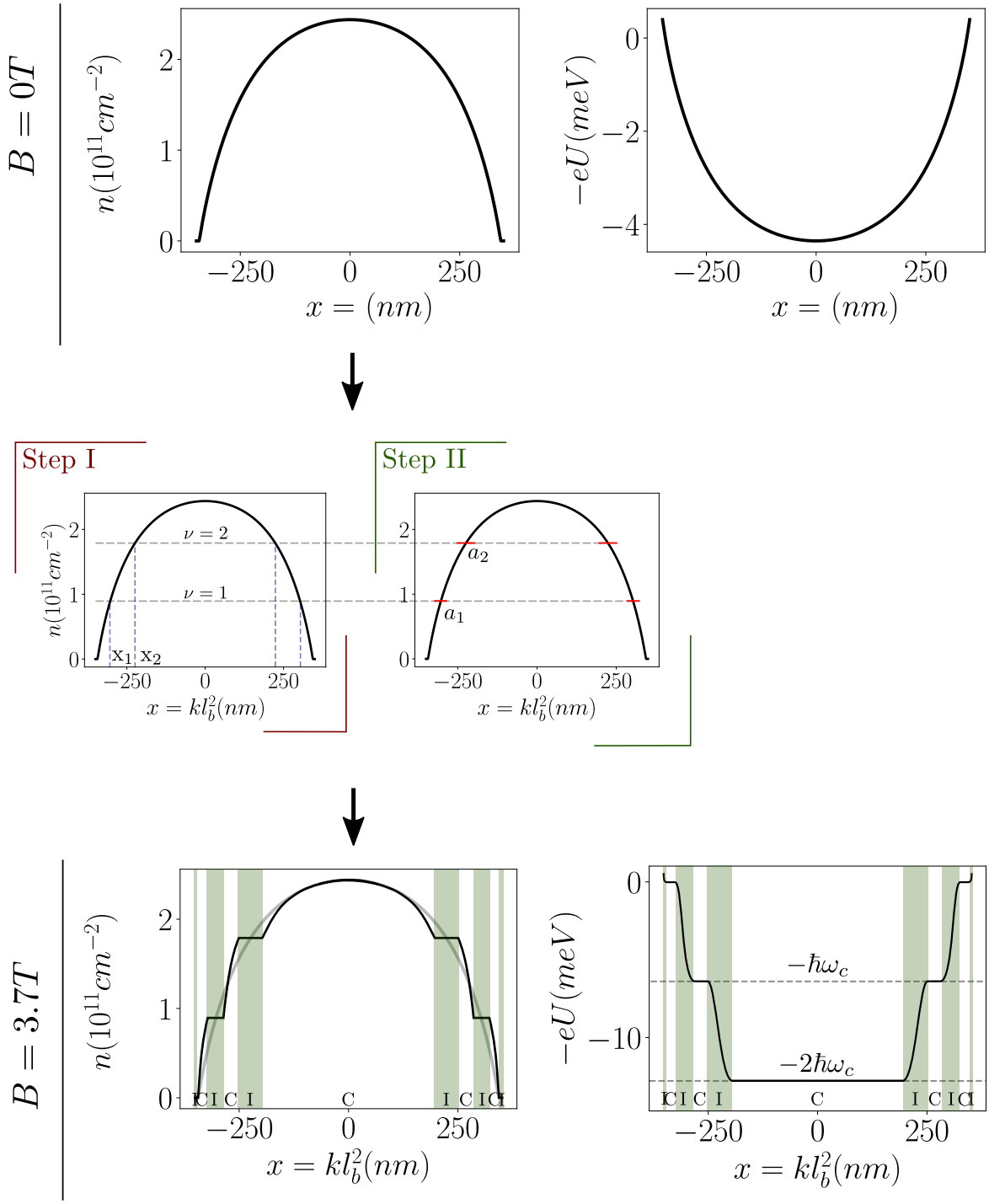


Figure 1.2: Illustration of the CSG method to derive the $B = 3.7T$ charge profile and energy dispersion from $B = 0T$ calculations. Here n and U are respectively the charge density and potential at the 2DEG of Fig.1.1 (a), with l_b the magnetic length and k the electron momentum. The profile at $B = 0T$ (top) is calculated using the *PESCADO* software, c.f. Chapter 4. The $B = 3.7T$ profile (bottom) is obtained using the CSG method. C and I correspond to respectively the compressible and incompressible stripes of the CSG model for the QHE. In the middle the two Steps to go from top to bottom, with ν the filling factor, x_ν the central position for the Landau level ν and a_ν the size of the Landau level ν.

so that the energy E varies continuously. The index α, however, is discrete and labels the different sub-bands (including those originating from the non-trivial band structure). A typical setup that corresponds to the above is the one of the Kwant package [Groth *et al.* 2014]. Solving the quantum problem provides the wavefunction $\Psi_{\alpha E}(\vec{r})$ or more generally the Local Density of States (LDOS) that counts the density of electrons per unit volume and per unit energy:

$$\rho(\vec{r}, E) = \frac{1}{2\pi} \sum_{\alpha} |\Psi_{\alpha E}(\vec{r})|^2 \tag{1.9}$$

Filling the states up to the Fermi energy (statistical physics) provides the electron density $n(\vec{r})$, also called Integrated local density of states (ILDOS):

$$n(\vec{r}) = \int_{-\infty}^{\infty} dE \rho(\vec{r}, E) f(E) \tag{1.10}$$

where $f(E) = 1/[e^{\beta(E-E_F)} + 1]$ is the Fermi function with E_F the Fermi energy and $\beta = 1/k_B T$ the inverse temperature. The last equation is simply the Poisson equation :

$$\nabla \cdot (\varepsilon(\vec{r}) \nabla U(\vec{r})) = e n(\vec{r}) - e n_d(\vec{r}). \tag{1.11}$$

where $\varepsilon(\vec{r})$ is the dielectric constant and the doping charge density $n_d(\vec{r})$ accounts for electric charges that are not treated at the quantum mechanical level (for instance dopants or charges trapped at a surface). The Poisson equation is complemented by boundary conditions such as Dirichlet $U(\vec{r}) = V_g$ at metallic gates or Neumann on other boundaries.

1.2.2 Review of common numerical approaches

The traditional approach to solve the SCQE problem involves calculating the potential $U(\vec{r})$ with the Poisson equation from a given input density $n(\vec{r})$. Then to calculate a new value for the density from the potential $U(\vec{r})$ using the ILDOS. Then to iterate over this until the charge $n(\vec{r})$ converges. Alternatively one can start with an initial guess for the potential $U(\vec{r})$ [Gummel 1964, Tan *et al.* 1990, Gudmundsson 1990]. To speed up convergence a simplified model can be used to find an initial guess for either $U(\vec{r})$ or $n(\vec{r})$, e.g. [Wang *et al.* 2006]. These simple approaches work when the filling of energy levels is independent of the variations the value $U(\vec{r})$ takes in between iterations. In fact, we can use such criteria to classify SCQE problems into two categories: confined systems (e.g. molecules) described by a discrete gapped set of orbitals and extended systems (e.g. semiconductors and 2D materials, the focus of this thesis) with a continuous spectrum. For confined systems calculating the ILDOS (from the Density of States (DOS)) is straightforward, since the presence of a large gap makes the integrated charge density invariant under the variations $U(\vec{r})$ might take in between iterations. For extended systems however, this is not the case. The ILDOS can be highly non-linear, specially at lower temperatures, making iterative algorithms unstable and even diverge.

To improve on the convergence one of the simplest method is the under-relaxation algorithm [Rana *et al.* 1996, Stern 1970]. It consists of adding a damping parameter on the previous iteration solution. Regardless, even when simple iterations with or without under-relaxation converge, they do it slowly [Trellakis *et al.* 1997]. More stable and faster approaches tend derive the next iteration input by mixing the solutions of several previous iterations [Eyert 1996]. For instance, the Anderson mixing uses a linear combination of several previous iteration as the next iteration input [Vuik *et al.* 2016, Anderson 1965, Antipov *et al.* 2018, Escribano *et al.* 2019]. Other mixing methods worth mentioning are the Direct Inversion in the Iterative Subspace, a.k.a. Pulay mixing [Pulay 1980] and the Broyden mixing [Broyden 1965]. We refer to [Eyert 1996] for a good formal reference of the most used mixing algorithms. We refer to [Vuik *et al.* 2016] for a good practical example on the implementation of the Anderson mixing. Not only they justify their choice of mixing by benchmarking with other methods, but they even make their code available ![2]

Other than direct iterative algorithms, root-finding methods have also been used. For instance, [Andrei & Mayergoyz 2004, Lake *et al.* 1997, Pacelli 1997, Kumar *et al.* 1990] have used Newton-Raphson based algorithms to solve the SCQE problem. The major improvement over the mixing approaches is that Newton-Raphson and variations also make use of the gradient of the function they seek to find the root. For a good source to delve into the helm of non-linear PDEs - of the likes found in fluid dynamics simulations; we refer to [Knoll & Keyes 2004].

However, either root-finding or mixing algorithms are still too unstable. More modern methods implement a Predictor-Corrector approach [Mikkelsen *et al.* 2018, Trellakis & Ravaioli 1999, Curatola & Iannaccone 2003, Ren *et al.* 2003]. It consists of finding an approximation to SCQE problem - called "predictor"; that can be solved using root-finding or mixing algorithms. Then we solve the predictor, take its solution and use it to update the predictor. The problem we use to correct the predictor is called "corrector". Therefore a simple Predictor-Corrector scheme is to solve the predictor, use its solution as input to the corrector, and then from the corrector output update the predictor.

For the SCQE problem the predictor takes the form of a Non Linear Helmholtz (NLH) equation [Trellakis & Ravaioli 1999, Curatola & Iannaccone 2003, Niquet *et al.* 2014]:

$$\nabla.(\varepsilon(\vec{r})\nabla U(\vec{r})) = \int_{-\infty}^{\mu(\vec{r})} \rho(\vec{r}, E)\mathrm{d}E \tag{1.12}$$

with

$$\mu(\vec{r}) \pm eU(\vec{r}) = E_F \tag{1.13}$$

and where $\rho(\vec{r}, E)$ is calculated by solving Eq.(1.9) and is now independent of $U(\vec{r})$. Often we use the Thomas-Fermi (TF) approximation to calculate the LDOS

[2] It is unfortunate that this is not standard practice

on the right hand side of the NLH. In the TF approximation we calculate the LDOS for a potential U that is not necessarily the solution of the NLH problem, but close enough to give a reasonable LDOS. For instance, one generally uses the bulk LDOS, i.e. $U = 0$. This transforms the SCQE problem into:

$$\nabla.(\varepsilon(\vec{r})\nabla U(\vec{r})) = \int_{-\infty}^{\mu(\vec{r})} \rho(E)\mathrm{d}E \tag{1.14}$$

with $\rho(E)$ independent of $U(\vec{r})$.

Depending on the physics we seek to capture, solving the SCQE at the TF approximation gives a solution not far from the full SCQE problem. See [Winkler *et al.* 2019, Escribano *et al.* 2019], notably [Mikkelsen *et al.* 2018] for a good comparison of the potential obtained within the TF approximation and a full SCQE calculation for a Semiconductor-Superconductor nanowire device. Another example is our study of the QH edge channel spacing in Graphene PN junctions [Flór *et al.* 2022]. The calculations we performed under TF were in good agreement with the experimental measurements. See Chapter 5 for more detail.

To finish formulating the Predictor-Corrector problem, we define the corrector as the Schrödinger equation. Therefore in a Predictor-Corrector scheme for the SCQE, we solve Eq.(4.3) for an initial $n(\vec{r})$, then from its solution $U(\vec{r})$ we use the Schrödinger equation to update $n(\vec{r})$ - and then iterate. For instance, this is the algorithm implemented by the NextNano commercial software [Birner *et al.* 2007, Trellakis *et al.* 1997].

The main limitation of the Predictor-Corrector approach is the stability of the NLH equation solver (the predictor). When the LDOS shows rapid variations in energy, the ILDOS becomes too non-linear and current approaches tend to fail. Generally the most advanced methods to solve the predictor are gradient based root-finding schemes. However, they do not handle well functions with cusps. Cusps are discontinuities at the derivative of the function. Those discontinuities can destabilize gradient based methods. In fact, in extreme cases it can even cause them to diverge. One good example is the ILDOS for the QHE. At zero temperature the ILDOS consists of a series of cusps stacked together. In [Armagnat *et al.* 2019] they use a simple 1D toy model to illustrate how fast a SCQE problem formulated using the QHE ILDOS diverges. It is possible to smooth out an ILDOS with cusps. For instance one can use an artificial temperature. However, this comes with the cost of degrading the energy resolution of the simulations. During my thesis we have developed a simple algorithm that isolates the cusps and treats them with care. The algorithm is called Pure Electrostatic Self consistent Approximation (PESCA), we explain it in detail in Chapter 2.

The PESCA algorithm was inspired by my predecessor's work, Pacome Armagnat [Armagnat 2019]. During his thesis he managed to pinpoint when the established algorithms, such as the one of NextNano, failed to converged. After much suffering he figured the origin of the instabilities in the self-consistent iterations were the regions in the simulation where the charge density reached zero. The algo-

rithm P.Armagnat developed during his thesis [Armagnat *et al.* 2019] is already an improvement over the traditional Predictor-Corrector approach.

1.3 Modeling the electrostatic environment of semiconductor based nanoelectronic devices

Predicting the field effect a gate has on a conducting region placed tens to hundreds of *nm* into a semiconductor heterostructure is not a simple task. The materials science behind semiconducting devices is incredibly rich. The workfunction difference at the gate - heterostructure interfaces and the band offset at the semiconductor - semiconductor interfaces modify the band diagram of the device. The dopants dynamics and chemistry is rich, e.g. dopants can interact with each other forming new energy levels in the band diagram. And on top of all that, defects permeate the device adding unintended energy states in the band gap. The objective of this thesis is not to perform a model capturing each detail of the device materials science, it is rather to understand how - as a whole, they affect the field effect of the gates. To this goal we formulate two questions. First is how the device materials science contribute to the potential we fix at the gates. Second is how this potential is going to be screened by the charges (intended and unintended) everywhere in the structure.

In searching for an answer to these two questions we learn how each individual microscopic effect contributes to the behavior of the device. Nonetheless, in itself each effect is complex enough [Das Sarma & Hwang 2014, Wang *et al.* 2013, Das Sarma & Stern 1985, MacLeod *et al.* 2009] such that a model capturing their behavior as a unique ensemble should not search to capture their individual behavior to perfection. In fact - a so to say - model with perfect fidelity to each microscopic effect individually, would have so many fitting parameters that it could be used to describe virtually anything. Therefore loosing any sort of predictive quality. To fix this we need an intermediary description to their behavior that satisfies the level of approximation we require and minimizes the fitting parameters. Preferably one that removes any fitting parameters and replaces them by experimentally tuned parameters. This raises a third question, how the screening of the gate potential can be measured experimentally. That is, what type of experiments can help us develop the appropriate level of modeling required for predicting quantum transport measurements ?

In this section I shall first go through the main microscopic effects taking place in a semiconducting device. The goal here is to have a correct phenomenological view of the physics taking place. What happens, where it happens, and how does it affect the gate field effect. Then I shall discuss how they can be measured experimentally.

1.3.1 Microscopic description of semiconducting nanoelectronic devices

A typical nanoelectronic device is a quantum point contact (QPC) fabricated over an AlGaAs/GaAs heterojunction. We shall use it as a case study. As we define the microscopic effects taking place in a semiconducting device, we shall locate it on the QPC. Figure 1.3 shows an schematic of the QPC. The small schematics on the top left shows the entire device. The enlarged schematics on the lower right gives a detailed view of the different effects taking place at the device. In grey are the undoped dielectrics, green the doped dielectric layer, red the 2DEG formed at the GaAs layer, wine and and gold the metallic gates. The simplest model for the device of Figure 1.3 is one where the charge density at the dopants is constant and the voltage set at the gates is the value fixed during experiments. At the 2DEG the charge density is calculated by solving the SCQE problem. Everywhere else in the structure the charge density is set to zero. In what follows we explain how this reference - minimalistic - model differs from a real device.

1.3.1.1 Workfunction difference

A materials workfunction is the smallest energy required to move an electron from the materials surface to infinity. When stacking materials with different workfunction electrons will move from the highest workfunction material to the lowest until equilibrium is reached. This induces an electrostatic potential at the interface that bends the valence and conduction bands.

For a metal-insulator-semiconductor junction this represents an additional potential at the gates. In nanoelectronics we call flat band voltage the gate voltage applied at the gate to compensate for this additional potential. In practice, we account for it as an offset to the gate voltage used in the simulations. For the QPC example of Figure 1.3 this implies a voltage offset must be added to the side and QPC gates in gold and wine respectively. The actual values of the voltage offset depend strongly on the interface physics taking place. It is something that should be calibrated experimentally (unless you wanna do a PhD in interface physics).

For a semiconductor-semiconductor interface the band bending can cause the accumulation of charges in e.g. a potential well. Generally the accumulation region is dynamically filled with the gate voltage, hence a proper treatment its behavior requires solving the SCQE problem. At the QPC heterostructure - beyond the accumulation region where the 2DEG is formed, there is also one located at the GaAs/AlGaAs interface between the gates and the dopants. Recent measurements made by Boris Brun within the Lateqs group [Brun & Fouad Kalo 2023] seem to indicate that the charges accumulated at the GaAs/AlGaAs interface might measurably affect their device behavior.

1.3.1.2 Dopant chemistry and dynamics

To a perfect semiconducting material, we can add impurity atoms. Intentional impurities are called dopants. These dopants will add energy levels in the band gap. They are called donors if close to the conduction band and acceptors if close to the valence band. The energy levels added to the gap are characterized by a density n_0 and activation energy E_A. The activation energy is the energy required for a dopant to be ionized, i.e. for an electron to go from a donor level to the conduction band or from the valence band to an acceptor level.

With regards to the dopants dynamics, lets at first assume n_0 and E_A to be known and constant under temperature and potential energy variations. Under this considerations, the dopants dynamics are controlled by the E_A parameter. The ratio between E_A and the thermal energy dictate how many dopants will be ionized. If E_A is small compared to the thermal energy, then all dopants are ionized. If E_A is on the order of the thermal energy then the dopants are only partially ionized. If E_A is large compared to the thermal energy, then the dopants are frozen. When the dopants are fully ionized the dopant layer behaves as a charged layer. However, if the dopants are partially ionized, the number of ionized dopants change with, e.g. gate voltage [Buks *et al.* 1994b, Buks *et al.* 1994a]. A correct treatment of this behavior requires solving the SCQE problem at the dopant layer. Furthermore, if the charge density at the dopant layer can change, then it can screen the field effect of the gate. At high and intermediary temperatures, the dopants are generally fully or partially ionized, i.e. they are mobile. For the low temperatures of quantum transport measurements the dopants are generally frozen. The variation of dopant dynamics as a function of temperature can lead to a phenomena known as bias cooling. There the gate voltage required to deplete the charge carriers at low temperatures depend on the voltage applied at the gates during cooldown [Buks *et al.* 1994a, Buks *et al.* 1994b]. This is an important effect to account for when studying quantum nanoelectronic devices, we shall come back to it in Chapter 6 when discussing the predictive modeling of quantum point contacts.

With regards to the dopants chemistry, this will play a role on the actual n_0 and E_A values of the added energy levels. First, during sample fabrication the doping impurities can move, causing n_0 to vary in space and deviate from the intended value, e.g. due to thermal diffusion. Second, E_A is controlled by the energy position in the band gap of the doping energy state. This is related to the configuration a doping impurity occupies in the semiconductor crystalline structure. First, a doping impurity will not necessarily have a single stable configuration. Hence for a single type of doping impurity one can have different E_A values. And even if there was a single configuration, "the" stable configuration is affected by modifications of the crystalline structure, e.g. strain, alloy composition. It also follows that other dopants, which shuffle the original atomic atoms around, will interact to form additional dopant configurations. Lets take as example the doped AlGaAs layer of Figure 1.3. When doped with Si atoms [Chand *et al.* 1984], the AlGaAs crystalline structure will generate what we call a d_0 donor level. It is energetically interesting

for two d_0 levels to react and form a negatively charged bound state called DX^- center. We refer to [Chadi & Chang 1989] for a detailed description of DX centers. The consequence of this that the actual behavior of the donor layer will be a result of not only the gate voltage and temperature, but also correlation between the DX^- centers and d_0 donor levels [Buks *et al.* 1994a]. In conclusion, detailed modeling of the dopant chemistry is complex. Fortunately, we do not need to perform such modeling to capture the dopants chemistry effect on quantum transport, see Chapter 6.

1.3.1.3 Charged defects

Lets move on to the defects found in bulk semiconductors and at the interfaces between two different materials. To the level of detail of this introduction we classify them into two types: unintended impurities and bulk crystalline defects. The unintended impurities are atoms (accidentally) added during growth, processing (e.g. doping) and due to reaction with external environment (e.g. air). They behave in the same way as a doping impurity. Only their chemistry is less understood. The crystalline defects can be located at the bulk (e.g. dislocations due to growth) but are generally located at the interfaces. For instance, at the surface of a semiconductor (interface with e.g. air) one can find dangling bonds [Atalla *et al.* 1959] that capture charge and cause scattering [Wang *et al.* 2013]. At the interface between two materials two relevant defects are interface roughness and charged traps, e.g. [Martinez & Niquet 2022]. Indeed, at the interface the different lattice spacings will induce a mechanical strain. This mechanical strain will at best distort the atomic lattice and at worst break the atomic bonds. Lattice defects change locally the potential background on which the electrons live. By themselves this change can create energy states in the band gap that are both localized in energy and space. The change in the potential landscape can also attract impurities, which themselves carry additional energy states. This leads, for instance, to a concentration of impurities at the interfaces or at the grain boundaries in polycrystalline materials. Anyhow, to the level of modeling regarding this thesis, we only consider the defects to add available energy levels in the band gap. That is, we do not introduce in the model neutral defects such as interface roughness. The charged defects (either in the bulk or at the interface) are characterized by a density and activation energy, e.g. [Rassekh *et al.* 2019]. Their dynamics follow the same discussion as for the dopant impurities. Applied to the QPC device of Figure 1.3, the main source of charged defects is the Air/GaAs interface, i.e. surface states [Wang *et al.* 2013]. A second meaningful source is the bulk GaAs wafer, e.g. background impurities added during growth [MacLeod *et al.* 2009, Shayegan *et al.* 1988a, Pfeiffer *et al.* 1989].

One special consideration must be added to the interface defects at the gate/insulator interface, e.g. Metal/GaAs interfaces in Figure 1.3. If the defect distribution is homogeneous at the gate/insulator interface, then their effect is to induce a potential offset. However this is rarely true and their usually inhomogeneous distribution breaks the equipotential lines formed at the gate side of the interface. When the

gate is metallic the high DOS implies a fast stabilization of the electric field lines. Hence when simulating a metallic gate, e.g. gold and wine regions in Figure 1.3, the true equipotential assumption is good.

1.3.1.4 Summary

The difference in workfunction between the gate and semiconductor causes a voltage offset. The defects at the interface will both cause a voltage offset and break the equipotential lines at the gate interface. The accumulation region at the interfaces, charged defects (bulk and interface) and the complicated dopant dynamics / chemistry will alter the field effect of the gate over the active region of the device. The enlarged schematics of Figure 1.3 contains all relevant microscopic effects for the AlGaAs/GaAs QPC.

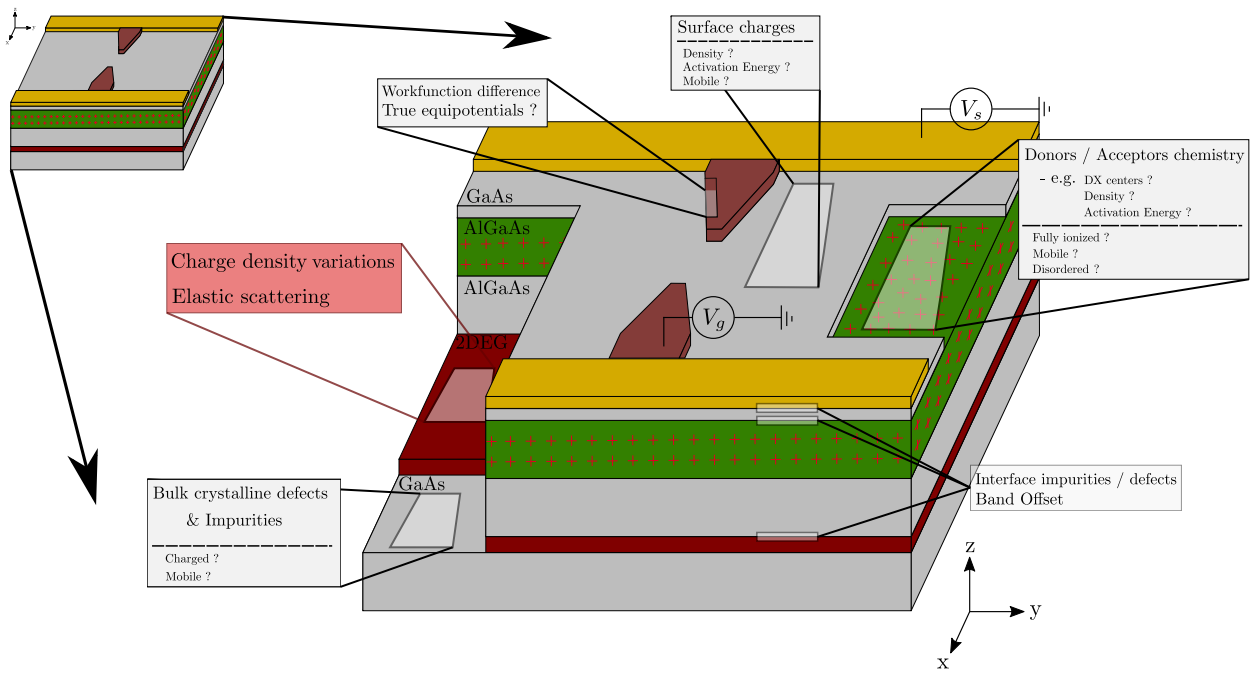


Figure 1.3: Relevant defects in a AlGaAs/GaAs heterostructure. Inspired from Fig.1 of [Armagnat *et al.* 2019]

1.3.2 Experimental signatures

The workfunction difference and charges located near the gate cause - for the most part - a shift of the entire quantum transport curve. For the devices studied in this thesis the gates are a few hundreds of *nm* away from the active region. Hence any inhomogeneity in the electric field lines at the gate interface are smoothed out. Adding a voltage offset parameter - calibrated from experimental data - seems enough to capture the physics at the gate interface.

Regarding the dopant behavior and other unintended and intended charges in the structure, it is a bit more complicated. Their effect will be to screen the field effect of the gate and to add disorder to the electric potential at the device active region. The screening effect can be observed by studying the conductance versus gate voltage characteristics. The voltages for which the conductance vanishes or presents a cusp are called pinch-off voltages. The pinch off values are directly correlated to the amount of screening the gate suffers from the charge everywhere in the device. Hence, up to a quite good approximation, understanding the charge distribution in a device amounts to predicting its pinch-off voltages. This is essentially what we do in Chapter 6.

The effect of dopant disorder and unintended charges can be summed up as a disordered potential $U_d(\vec{r})$ defined at the device active region. Lets take as example the the 2DEG in Figure 1.3. Assume we have a model that calculates $U_d(\vec{r})$, by solving e.g. the SCQE problem at the 2DEG with a disordered dopant charge distribution. Then we model the 2DEG with a Hamiltonian for a free electron under the effective mass approximation, noted H_0, to which we add the disordered potential $U_d(\vec{r})$ s.t. $H(\vec{r}) = H_0 + U_d(\vec{r})$.

First, spatial variations of $U_d(\vec{r})$ are followed by spatial variations of the charge density in the active region, e.g. the 2DEG. Here we can introduce two length scales, the distance between two identical nanoelectronic devices in the same wafer (or die) and the length of the device active region. If the spatial variations of the charge density are on the order of the distance between two devices, then the quantum transport measurements for a device with the same design will be sample-dependant. This effect has been observed experimentally by the QPC fabricated by C.Bäuerle we study in Chapter 6. If the spatial variations of the charge density is on the order of the active region, then lower density regions can be depleted before higher density ones. In Chapter 6 (section 6.5.3) we use a percolation model to describe how transport can be affected by such variations. Other recent experimental studies on the spatial variation of the charge density are [Chung *et al.* 2019, Qian *et al.* 2017, Zhou *et al.* 2015].

Second, a disordered potential will cause the electrons to scatter. We focus on systems whose propagating states coherence length is larger then the active region of the device. Hence we can neglect inelastic scattering. It is well known that time independent fluctuations of $U_d(\vec{r})$ generate measurable sample dependant fluctuations on coherent quantum transport. For instance, spatial fluctuations of the potential have been shown to generate sample dependant magnetoresistance and conductance fluctuations in mesoscopic wires [Stone 1985, Al'tshuler & Spivak 1985, Lee & Stone 1985]. Another interesting work is that of [Ralls *et al.* 1984]. They show that changing the position of a single impurity can induce conductance oscillations of the order of $e^2/\hbar$. Lastly, [Topinka *et al.* 2001] observed branching of the electronic flow after it goes through a constriction, in their experiment a QPC. This branching is due to disorder in the dopant layer of their device. The elastic scattering due to disorder in the potential can focus the electronic flow in a 2DEG to certain areas of the sample [Fratus *et al.* 2019]. In Chapter 7 we use neu-

ral networks to reconstruct the disorder potential in a 2DEG from scanning gate microscopy measurements [Percebois & Weinmann 2021, Jura *et al.* 2009].

1.4 Purpose of this thesis and summary of the main results

The electrostatic energy is the largest in the system, the energy associated with the quantum effect we actually seek to understand is only a fluctuation in comparison. Nature always seems to minimize energy, and in the same way as we would not approach any optimization problem, it doesn't make sense to attempt to understand Nature by only minimizing its fluctuations. In Section 1.1 we have illustrated this argument through the QHE, and have shown the importance of correctly understanding electrostatics (the sea) to correctly understand the quantum effects (the waves). After, in Section 1.2, we have formulated the SCQE problem, which upon solving couples the waves to the sea they lie upon. As we have seen, this is not an easy problem to resolve.

The difficulty in solving the SCQE problem stems from: (i) potentially strong non-linearities due to quantum mechanics and (ii) non-locality of the electrostatic problem (long range electrostatic interaction). From my predecessor work, P.Armagnat, we took mainly two conclusions [Armagnat 2019, Armagnat *et al.* 2019]. The first regards difficulty (i), the SCQE problem becomes a lot more manageable when approximated as the NLH equation. Although he did not develop a method to actually solve the NLH, he did figure out an approximation allowing us to simplify the SCQE into a NLH, called the quantum adiabatic approximation - a generalization of the TF approximation. The second regards difficulty (ii), he attempted to approximate the non-local electrostatic problem into a more locally defined one. Although he was successful, the algorithm was not stable. The main message is that for one to develop a robust algorithm to solve the SCQE problem, one must deal with the non-local aspect of the electrostatic interaction without cutting corners.

The main purpose of this thesis is to solve the SCQE problem by first approximating it into a NLH equation and then solving the latter equation exactly. That is, without smoothing out the non-linearities in the ILDOS - as it is often done (e.g. increase temperature); and by accounting for the spatial fluctuations of the potential when calculating the *local* DOS - which are often ignored. This brings us to the first main result of this thesis: solving the NLH problem. To that avail we have :

- Introduced the PESCA. It removes the quantum aspect of the problem altogether by simplifying the semiconductor as a metal - the ILDOS has two branches, one where its slope (the DOS) is zero, and the other is infinity. What PESCA taught us is how to treat the cusp of the ILDOS - the discontinuity in the DOS. In developing PESCA we learned to isolate the cusp of

the ILDOS, and hence how to approach the non-linearities of any ILDOS in a simple and efficient way. One simply needs to keep track of the DOS and not (only) the ILDOS (the charge). It is the DOS that contains the information a NLH solver requires for it to converge, not the charge;

- We have implemented an iterative algorithm to solve the NLH problem that, at each iteration, takes as input the DOS. By having as information the dependence of the charge on the chemical potential, our algorithm knows where the cusps are located in the ILDOS. We can then isolate them and treat them accordingly;

- To be able to take as input the DOS, we have developed a Linear Helmholtz (LH) equation solver - the *PESCADO* software (see Eq.(3.15) for a definition of the LH). *PESCADO* is an open source python software. It solves the LH equation for 1D, 2D and 3D systems. To define the nanoelectronic device we simulate we have developed a geometrical engine and to discretize it a memory efficient and adaptive finite volume mesher.

Since the subject of this thesis is the predictive modeling of quantum devices, we want to predict the field effect of the gate on the active quantum region of the device. Naively we would think that in order to solve such problem we are required to model in detail the charges in the device. In the third section of this Introduction, section 1.3.1, I have gone through where the charges are in the device and how they behave. The lesson is that the complexity of the phenomena taking place means we should not seek to model precisely their behavior. To discover with which precision we should model the charges, we have worked closely with Christopher Bäuerle's experimental group at Institut Néeel. They have fabricated 110 quantum point contacts (QPC) of 48 different gate designs [Chatzikyriakou *et al.* 2022]. For each device they have measured the conductance versus gate voltage characteristics. Their QPCs are built over the 2DEG located at a AlGaAs/GaAs heterostructure. By using their data we have concluded that :

- A minimal model should not attempt to predict the charge distribution at the 2DEG along the plane of propagation (we do not care about the charge profile perpendicular to the 2DEG plane). It should instead use the experimental value for the charge distribution as a input parameter. We have developed such model for Chris Bäuerle's QPCs. Using a single global value for the charge distribution, we have been able to predict the 2DEG depletion voltages of the QPCs with a precision of $5-10\%$. The limiting factor of our accuracy seems to be spatial variations of the charge density.

1.5 Outline of this thesis

This thesis is split into two parts. In the first we discuss the technical developments (Chapter 2-4) to solve the SCQE problem. In the second we talk about the modeling of nanoelectronic devices through solving the SCQE problem (Chapter 5-7).

In Chapters 2 and 3 we explain three new algorithms we have developed:

- In Chapter 2 we introduce the PESCA. It is the simplest level of modeling required to account for electrostatic screening and partial carrier depletion in a semiconductor device. We show that the validity of PESCA depends on the smallness of the $\kappa = C_g/C_q$ parameter, with C_g and C_q the geometrical and quantum capacitance respectively. We apply PESCA to a few common 2DEG devices, where κ is on the order of few percent. We show that it can quantitatively estimate the position in space where the 2DEG charge depletes as a function of gate voltage. We also extend the PESCA to the integer QHE, and obtain results in good agreement to those of [Armagnat *et al.* 2019]. Section 2.2 gives a detailed explanation of the PESCA algorithm;

- In Chapter 3 we explain two algorithms we have developed to solve the NLH equation: the piecewise Newton-Raphson algorithm and the piecewise linear helmholtz algorithm. We demonstrate the convergence of the second method within a finite number of iterations. Section 3.4.2 explains the two algorithms.

Then,

- In Chapter 4 we give a detailed description the *PESCADO* software. We illustrate the capabilities of the software by solving the SCQE problem for a constriction in a 2DEG (3D model) - see Figures 4.3 and 4.4. Then, we explain in detail (implementation details and code examples) the geometrical engine, mesher and linear helmholtz equation solver implemented in *PESCADO* . Lastly we explain how to solve the NLH with *PESCADO* using either of the two algorithms described in Chapter 3.

Regarding the second part of this Thesis, it is the fruit of collaborations with different groups, made possible through the technical developments of the first part. In what follows I will summarize the main results and indicate which sections contains new results :

- In Chapter 5 we calculate the QH edge channel separation in Graphene PN junctions. We study the experiments of [Wei *et al.* 2017, Jo *et al.* 2021] with a 2D model were we solve the SCQE problem under a Generalized TF approximation. The model has no fitting parameters. The main lesson learned from our simulations is that the edge separation is controlled by the electrostatics of the device. This is in contrast to the explanation given by [Wei *et al.* 2017], where they considered the Exchange energy to control the separation. This conclusion can already be reached from the qualitative energetic argument of CSG we have described earlier in this Introduction, see Section 1.1. In Chapter 5 we went a step further and have shown quantitatively for it to be the case. In fact, we show the separation value to be independent of the exchange energy. For [Jo *et al.* 2021] our results match the experimental data with an

error smaller than 10%. For [Wei *et al.* 2017] the difference is on the order of 20%. Our results have been published as [Flór *et al.* 2022]. Sections 5.1 and 5.2 introduce the experiments. Sections 5.3, 5.4 and 5.5 explain the main results of the chapter;

- In Chapter 6 we have developed a minimal model free of fitting parameters capturing the electrostatics of quantum point contacts. We have compared our model to 110 experimental devices of 48 different designs fabricated and measured by Christopher Bäuerle's group at Institute Néel. We calculated the pinch-off voltages with an accuracy of 5-10%. The factor limiting the precision of our model seems to be slow spatial variations of the electronic density at the 2DEG. We give several arguments in our favor, notably experimental data that points out towards charge fluctuations on the order of ±5-10%. We propose a percolation model to account for the effect of charge density fluctuations on the pinch-off values. Our results have been published as [Chatzikyriakou *et al.* 2022]. A summary of the main results is given in Section 6.1 and for a detailed description see Section 6.4. Sections 6.2 and 6.3 give the technical details and Section 6.5 gives a critical discussion of the main results.;

- In Chapter 7 we use a neural network and *PESCADO* to reconstruct the potential disorder (due to dopant disorder) from Scanning Gate Microscopy (SGM) measurements. We have used the model developed in Chapter 6, *PESCADO* and Kwant [Groth *et al.* 2014] to generate a large numerical dataset of SGM data. Then G.Percebois and D.Weinmann at Strasbourg developed and trained a neural network into reconstructing the disorder potential from the SGM data. We have then used the reconstructed disorder potential to calculate a new set of SGM data and we compared it to the experimental results of [Iordanescu *et al.* 2020]. Our results have been published as [Percebois *et al.* 2023]. Section 7.5 shows the main results. Section 7.1 gives an introduction to SGM. Sections 7.2, 7.3 and 7.4 contain the technical details of the simulations.

Part I

Technical development

Chapter 2

The pure electrostatic self-consistent approximation

In quantum nanoelectronic devices, to a large extend it is the electrostatic energy that determines the charge distribution inside a device. In this chapter we introduce the pure electrostatic self-consistent approximation PESCA. PESCA provides the minimum level at which a semiconductor must be included in an electrostatic calculation to properly account for both screening and partial depletion due to e.g. field effect. We show that PESCA captures the salient features of interest in actual devices within a accuracy better than $\sim 1\%$ in many situations.

In Section 2.1 we introduce the small parameter $\kappa = C_g/C_q$, the ratio of the geometrical capacitance to the quantum capacitance. The parameter κ is what controls the validity of the PESCA. Then, in Section 2.2 we formulate PESCA and the algorithm we use to solve it. In Section 2.3 we apply PESCA to calculate the pinch-off phase diagrams in the gate voltages space for a split quantum wire defined in a GaAs/GaAlAs heterostructure. Notably, in Section 2.4 we show how to use experimental pinch-off diagrams to extract important information characterizing the device measured, such as the dopant concentration. Then, in Section 2.5 we extend the PESCA algorithm to solve the SCQE problem when the ILDOS is a step like function. We use as example the integer quantum hall effect ILDOS. Finally, we compare our calculation to the Thomas-Fermi results obtained by [Armagnat *et al.* 2019].

2.1 The small parameter of the SCQE problem

To get started, let's consider a SCQE problem simple enough to be solved in a few lines of algebra. We consider an infinite 2DEG, such as the one found in GaAS/-GaAlAs heterostructures. The 2DEG is situated at a distance d from the surface, where we place a metallic electrode. A voltage V_g is applied between the metallic electrode and the 2DEG with a voltage source. Setting the zero of the energy in the metallic gate, the gate voltage sets the electro-chemical potential in the 2DEG $\mu = eV_g$. In the effective mass approximation, the density of states in the 2DEG is a constant $\rho = m^*/(\pi\hbar^2)$. The solution of the quantum part of the problem reduces to a simple spatially independent ILDOS,

$$n = \rho(eV_g - eU) \tag{2.1}$$

where the electric potential U in the the 2DEG sets the position of the bottom of the conduction band. The electrostatic problem reduces to a simple planar capacitor,

$$n = \frac{\varepsilon}{ed} U \tag{2.2}$$

Introducing the geometrical capacitance (per unit surface) $C_g = \varepsilon/d$ and the quantum capacitance $C_q = e^2\rho$ one arrives at,

$$n = \frac{1}{1+\kappa} C_g V_g \quad \text{with} \quad \kappa = \frac{C_g}{C_q} \tag{2.3}$$

For most semiconducting systems, the dimensionless parameter κ is actually very small $\kappa \ll 1$. Therefore from Eq.(2.3) we conclude that to good approximation the density in the 2DEG is given by $n \approx C_g V_g/e$, i.e the density in the 2DEG is entirely controlled by the electrostatics.

Table 2.1 shows an estimate of κ in a few common situations (GaAs and Silicon devices). We find that $\kappa \approx 1\%$ is the common situation. This means the pure electrostatic approximation, $\kappa = 0$ limit, makes an error of $\approx 1\%$. The PESCA that we will develop in this article is essentially the generalization of the $\kappa = 0$ limit to a spatially dependent problem. Like the $\kappa = 0$, the PESCA result is independent of the details of the quantum problem, e.g. the effective mass of the material, and as such clarifies how these (geometrical and quantum) parameters affect the physics.

Semi.	$\frac{m^*}{m_e}$	$\frac{\varepsilon}{\varepsilon_0}$	Deg.	d $[nm]$	C_q $[mF/m^2]$	C_g $[mF/m^2]$	κ
GaAs (e)	0.067	12.9	2	100.	89.7	1.14	1.3%
Si (e)	0.2	11.7	4	20	535.4	5.18	0.97%
Si (e)	0.2	11.7	4	5	535.4	20.7	3.9%
Si (h)	0.49	11.7	2	20	655.9	5.18	0.79%

Table 2.1: Typical values of the κ parameter for a few common 2DEG. Deg. is the degeneracy (2 for spin, 4 for spin and valleys in the silicon conduction band). e stands for electron gas and h for holes.

2.2 The Pure Electrostatic Self-Consistent Approximation: PESCA

Before we introduce the PESCA, let's define a concrete electrostatic problem that will be used for illustrations. We consider an infinite split quantum wire defined in a GaAs/GaAlAs 2DEG. A side view of the system is shown in Fig.2.1. The device is directly inspired by the experiments [Bautze *et al.* 2014, Roussely *et al.* 2018, Bäuerle *et al.* 2018] on split wires aiming at demonstrating flying qubits. Using the top side electrodes, one can deplete the gas underneath and define a wire. Using the central electrode, one can further split this wire into two, potentially leaving

some tunneling coupling between the two sub-wires. Besides the 2DEG, the model contains positive dopants of concentration n_{dop} $[m^{-3}]$ in the green region and surface charges of concentration n_{sc} $[m^{-2}]$ at the free surface of the stack. The metallic gate correspond to electric equipotentials at voltage V_{side} and V_{middle} respectively. The dimensions of the different layers are, from top to bottom : 7.5 nm (GaAs cap layer), 88 nm (doped AlGaAs) and 49.3 nm (AlGaAs). The 2DEG is considered infinitely small on $\vec{z}$.

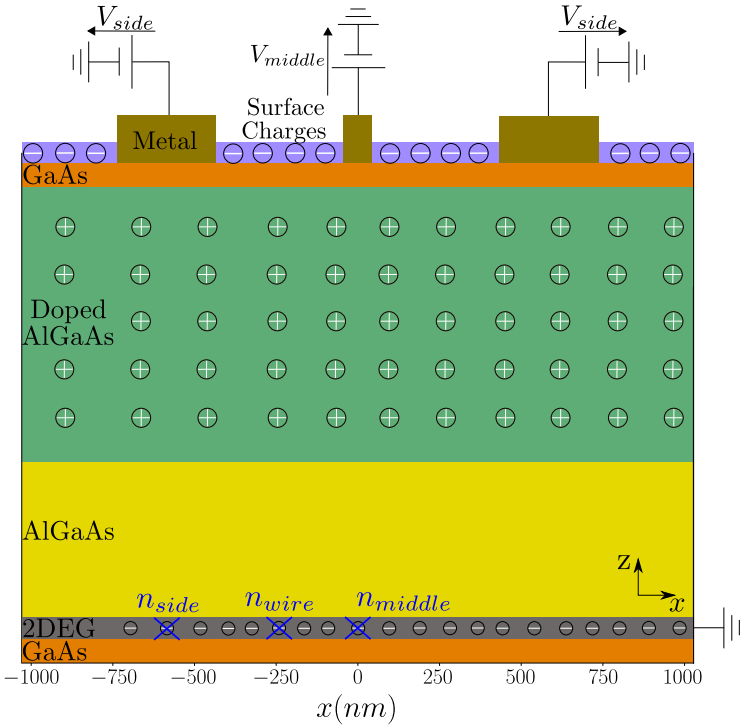


Figure 2.1: Side view of the split quantum wire system studied in this chapter. Various colors correspond to different regions. The active quantum region containing the 2DEG is shown in gray and is 10nm thick. The system is infinite along the y axis. The orange and yellow regions correspond to *GaAs* and *AlGaAs* respectively. The green region corresponds to the doped *AlGaAs* layer. A finite density of surface charges appears at the GaAs/vacuum upper interface and is shown in violet. Metallic electrodes are shown in brown and correspond to equipotential. The electron density at special points $x_{\text{side}} = -585\text{nm}$, $x_{\text{wire}} = -225\text{nm}$ and $x_{\text{middle}} = 0\text{nm}$ are called n_{side}, n_{wire} and n_{middle} respectively. We use different scales along the x and z directions (See text).

2.2.1 Formulation of PESCA

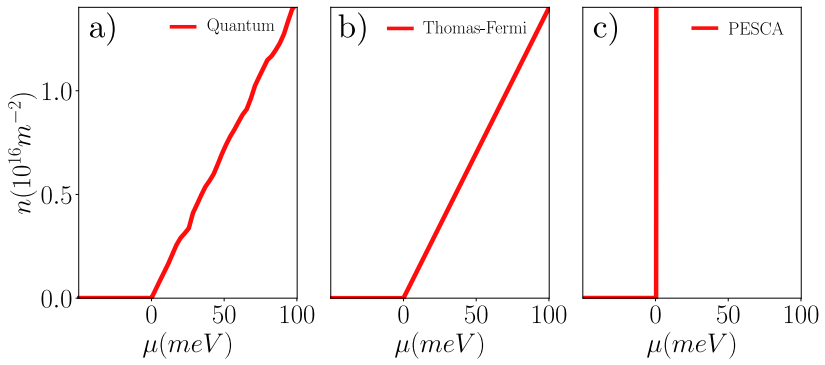


Figure 2.2: Example of ILDOS. a): ILDOS obtained from a quantum calculation. b) ILDOS corresponding to Thomas-Fermi for a 2DEG without magnetic field. c) ILDOS in the PESCA approximation. The latter is independent of the material.

Fig.2.2 (a) shows an example of the ILDOS calculated by solving Eq.(1.8) and Eq.(1.10) explicitly for a finite width 2DEG in the effective mass approximation. The curve is approximately linear above the bottom of the band with occasional small $\sqrt{\mu}$ kinks due to the opening of a conducting channel. It also has small variations between different positions x in the 2DEG. It should be noted however, that the biggest source of non-linearity is the large kink at the bottom of the 2DEG conduction band $\mu = 0$. Fig.2.2 (b) shows the ILDOS in the Thomas-Fermi approximation. Instead of solving the quantum problem for a finite system, one uses the bulk density of states calculated for a translation invariant 2DEG. Since the density of states of the infinite 2DEG is constant, the Thomas-Fermi ILDOS is linear for $\mu > 0$, with a slope equal to the averaged one of Fig.2.2 (a).

The PESCA approximation amounts to taking the slope of the ILDOS to be infinite. Therefore under PESCA the self-consistent problem is defined by the set of equations :

$$\nabla.(\varepsilon(\vec{r})\nabla U(\vec{r})) = en_d(\vec{r}) - en(\vec{r}, \mu(\vec{r})), \tag{2.4}$$

$$\mu(\vec{r}) = E_F + eU(\vec{r}), \tag{2.5}$$

where $n_d(\vec{r})$ is any constant charge in the system, e.g. dopants, and the ILDOS, only non-zero at the 2DEG for the current example, writes :

$$n(\mu) = \begin{cases} \infty & \forall \quad \mu > 0 \quad eV \\ 0 & \forall \quad \mu \leq 0 \quad eV \end{cases} \tag{2.6}$$

i.e. the ILDOS is made of a horizontal branch $n(\mu < 0) = 0$ and a vertical branch $\mu(n > 0) = \infty$, as shown in Fig.2.2 (c) (we remove the dependance on $\vec{r}$ for clarity).

For a bulk system it is equivalent to setting the small parameter $\kappa = 0$. Keeping only the vertical branch $\mu = 0$ would correspond to a good metal with a density of states high enough for the conductor to always remain an equipotential. Keeping only the horizontal branch would correspond to a fully depleted 2DEG. By keeping both branches, one allows for partially depleted devices, i.e. for a proper treatment of field-effect transistors. In particular, PESCA can predict the "pinch-off" phase diagrams, i.e. the values of the gate voltages where the gas is depleted underneath the side gates or the middle gates in Fig.2.1. Solving the PESCA problem amounts to finding which part of the 2DEG is on the horizontal/vertical branch and solve the associated electrostatic problem.

As argued in the preceding section, the PESCA is expected to be quantitative (within a few percent) for the calculation of thermodynamic quantities such as the density. Its validity for more subtle observables such as the conductance remains to be asserted. The relevance of PESCA lies in several features. (i) First, as we shall see, PESCA can be calculated very easily, at a much smaller computational cost than the full treatment of the SCQE problem. (ii) PESCA can help solving SCQE problems more accurately. For instance, suppose one wants to solve the SCQE problem of our split wire, c.f Fig.2.1, in a regime where the 2DEG is depleted below the side gates so that the wire is formed. One could treat the 2DEG inside the wire region $|x| < |x_{\text{side}}|$ by solving the corresponding quantum problem. However, it is inefficient to treat the 2DEG outside of the wire region in the same way, as it would be very costly computationally. It is not necessary to take into account that part of the 2DEG as it partially screens the effect of the gates. Treating this outside region $|x| > |x_{\text{side}}|$ within PESCA would provide a precise and efficient compromise with an automatic calculation of the position of the boundary of the wire. (iii) PESCA can quantitatively (i.e. within a few percent) predict the voltages needed to deplete the 2DEG in different regions (e.g. underneath the side or middle gates). With these depletion voltages we construct what we call "Pinch-off" phase diagrams, c.f. Fig.2.5. Pinch-off phase diagrams can be easily measured experimentally. They depend on the geometry of the device as well as on the charge distribution in the stack. As such, PESCA calculations may be very valuable to reconstruct the charge distribution of experimental devices and develop precise electrostatic models. Finding what is the correct distribution of e.g. ionized donors or surface charges, is indeed a prerequisite for predictive simulations of nanoelectronic devices, such as quantum point contacts [Chatzikyriakou *et al.* 2022] or the split wire devices studied in this chapter [Bautze *et al.* 2014]. (iv) PESCA forms the basis onto which more precise algorithms for treating Thomas-Fermi problems or SCQE problems will be built, c.f. Chapter 3.

2.2.2 PESCA algorithm

To solve the PESCA problem, one needs to extend traditional Poisson equation solvers s.t they allow one to dynamically update the status of the 2DEG so that some parts of it belong to the horizontal branch (depleted) while the rest belongs

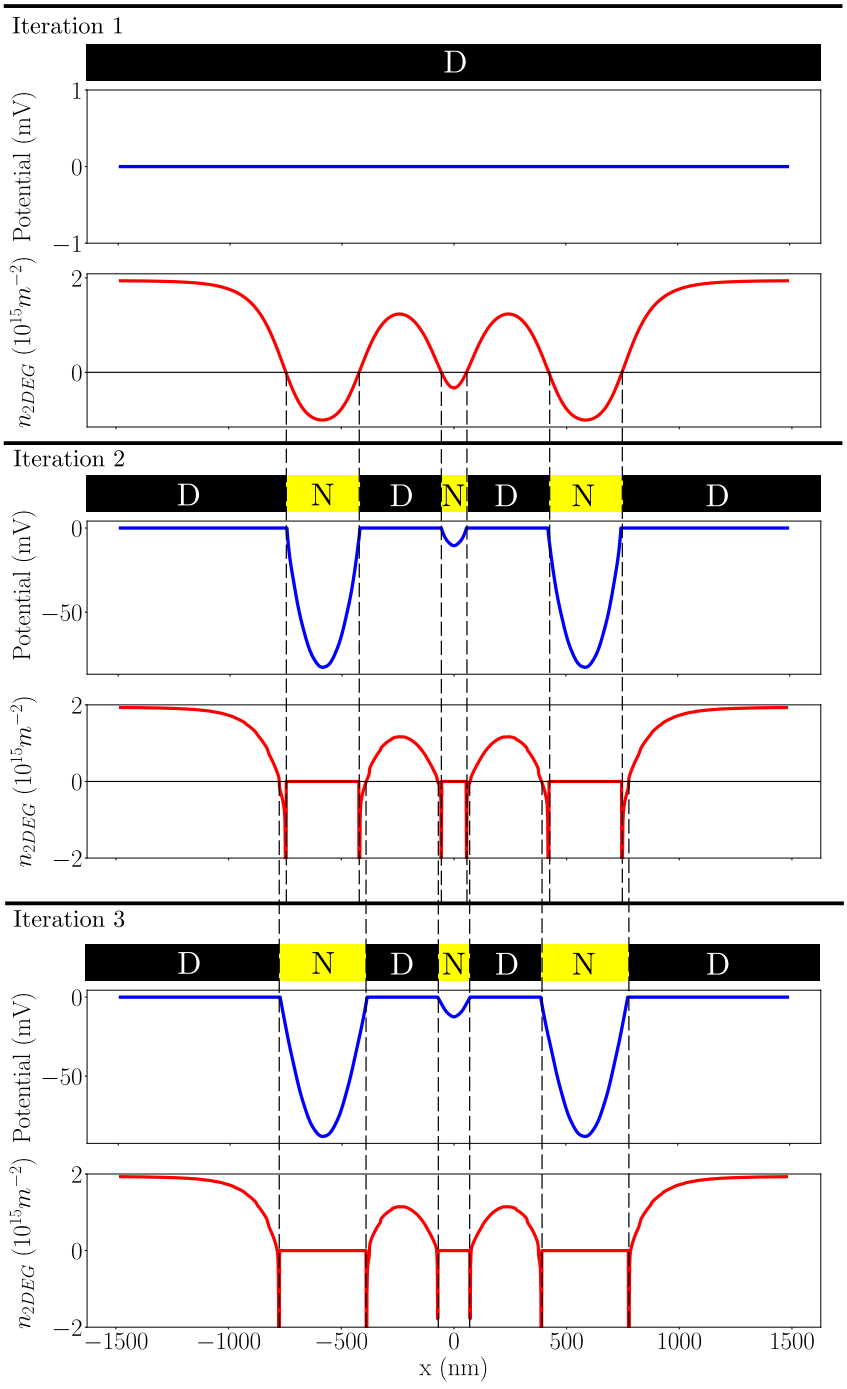


Figure 2.3: Illustration of the PESCA algorithm. For each iteration, the upper panel shows the $\mathcal{D}/\mathcal{N}$ partitioning (black for $\mathcal{D}$ cells, $\mathcal{N}$ for yellow cells), the middle panel shows the potential $U(x)$ for this partitioning (blue curve) and the lower panel shows the corresponding density profile $n(x)$ (red curve). The results shown here correspond to a full Dirichlet initialization. For this example, $V_{side} = -0.52V$ and $V_{mid} = -0.88V$.

to the vertical branch of the ILDOS (not depleted).

Following [Armagnat *et al.* 2019] we first discretize the Poisson equation into a finite volume discrete problem where the system is divided in to small cells i of finite volume. Other discretization schemes such as finite differences or finite elements could be use as well. One arrives at,

$$\sum_j C_{ij} U_j = Q_i. \tag{2.7}$$

where C_{ij} is a discretized version of the Laplacian operator, U_i the electric potential at the center of cell i and Q_i the number of electrons inside the cell :

$$Q_i = -\int_{C_i} \mathrm{d}\vec{r}[n(\vec{r}) - n_\mathrm{d}(\vec{r})] \tag{2.8}$$

A detailed derivation of the C_{ij} terms is shown in Chapter 3, see Section 3.1.

Cells that belong to the 2DEG can be sorted out in two categories. For those belonging to the horizontal part of the ILDOS, we know the density ($n_i = 0$) and want to calculate the potential U_i. We call these cells the Neumann cells ($\mathcal{N}$). The cells on the vertical branch of the ILDOS are of a different type that we call Dirichlet ($\mathcal{D}$). In these cells we know the electric potential $U_i = 0$ and want to calculate the density n_i. Writing Eq.(2.7) in a block form for the Dirichlet ($\mathcal{D}$) and Neumann ($\mathcal{N}$) blocks, it reads

$$\begin{bmatrix} C_{\mathcal{N}\mathcal{N}} & C_{\mathcal{N}\mathcal{D}} \\ C_{\mathcal{D}\mathcal{N}} & C_{\mathcal{D}\mathcal{D}} \end{bmatrix} \cdot \begin{bmatrix} U_\mathcal{N} \\ U_\mathcal{D} \end{bmatrix} = \begin{bmatrix} Q_\mathcal{N} \\ Q_\mathcal{D} \end{bmatrix}. \tag{2.9}$$

Rewriting the above equation so that the known inputs are on the right hand side and the unknowns on the left hand side, we get a new linear problem that can be solved using standard routines for solving linear problem with sparse matrices,

$$\begin{bmatrix} C_{\mathcal{N}\mathcal{N}} & 0 \\ C_{\mathcal{D}\mathcal{N}} & -1 \end{bmatrix} \cdot \begin{bmatrix} U_\mathcal{N} \\ Q_\mathcal{D} \end{bmatrix} = \begin{bmatrix} 1 & -C_{\mathcal{N}\mathcal{D}} \\ 0 & -C_{\mathcal{D}\mathcal{D}} \end{bmatrix} \cdot \begin{bmatrix} Q_\mathcal{N} \\ U_\mathcal{D} \end{bmatrix}. \tag{2.10}$$

The algorithm for solving PESCA solves Eq.(2.10) iteratively as follows. (I) First, we partition the 2DEG between $\mathcal{N}$ and $\mathcal{D}$ cells. For instance, one may set all the cells to be $\mathcal{D}$, which assumes the 2DEG is nowhere depleted. (II) Second, we solve Eq.(2.10) to calculate $U_\mathcal{N}$ and $Q_\mathcal{D}$. (III) Third, we determine the new partitioning of the cells. To do so we proceed as follows.

For the $\mathcal{D}$ cells,

- if $Q_i > 0$ then the cell remains a $\mathcal{D}$ cell.
- if $Q_i \leq 0$ then the cell is in the wrong branch of the ILDOS and one must change it to $\mathcal{N}$.

For the $\mathcal{N}$ cells,

- if $U_i < 0$ then the cell remains a $\mathcal{N}$ cell.

- if $U_i \geq 0$ then the cell is in the wrong branch of the ILDOS and one must change it to $\mathcal{D}$.

We repeat steps II and III until the partitioning is stable. Since there is a finite number of partitioning, the algorithm is guaranteed to converge in a finite number of iterations. In practice the convergence is extremely fast (see Fig.2.4) and when attained the result is exact.

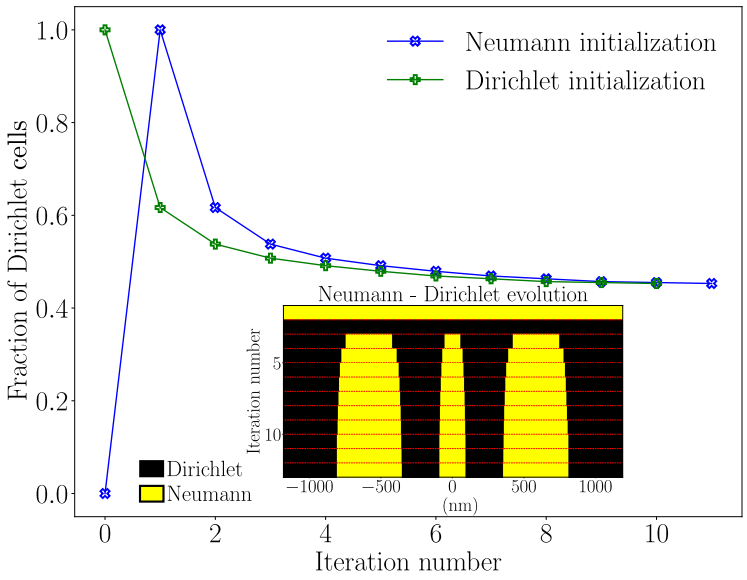


Figure 2.4: Fraction of Dirichlet cells ($\mathcal{D}$) in the 2DEG as a function of the number of PESCA iterations for $V_{side} = -52V$ and $V_{mid} = -0.88V$. In green all cells belonged to $\mathcal{D}$ at the first iteration. In blue all cells were set to Neumann ($\mathcal{N}$) in the initial configuration. The inset on the right corner shows the $\mathcal{D}/\mathcal{N}$ partitioning as a function of position x. The black regions correspond to Dirichlet cells and yellow to Neumann points.

Fig.2.3 shows the first three iterations of the above algorithm for the split wire geometry. Starting from a full Dirichlet initialization in iteration-1, we calculate the associated density (red curve) and find that it is negative below the gates that have been polarized with a negative voltage. In iteration-2, we first update the $\mathcal{D}/\mathcal{N}$ partitioning and assume that cells with $Q(x) < 0$ are actually depleted. Then we calculate a new density and potential profile. Since some cells are now depleted, the density close to them is slightly affected (those cells used to screen the metallic gates and do not do so anymore) and the position of the $\mathcal{D}/\mathcal{N}$ interface moves slightly in iteration-3. After a few iterations, the partitioning converges to its final form. The full set of iterations is shown in Fig.2.4, where the fraction of Dirichlet cells is shown as a function of the iteration number for two different initializations. At 5 iterations most cells are converged and both $U(x)$ and $Q(x)$ profiles are calculated with good precision. Within 12 iterations the cell partitioning is fully converged.

We find that for an initialization where there is an equal amount of Dirichlet and Neumann sites, convergence is achieved within 10 iterations as well. The inset of Fig.2.4 shows the evolution of the partitioning with the number of iterations. We identify in yellow the depleted regions and in black the non depleted regions.

2.3 Application to Pinch-off phase diagrams

We now turn to a practical application of PESCA for the split wire geometry. The electrostatic model of such a wire contains several parameters that are more or less well known. For instance the density of dopants is only approximately known by the grower, and even if known, the fraction of the dopants that are actually ionized is difficult to assess. Additionally, a large fraction of the electrons coming from the dopants actually go to surface states. Hence the actual fraction of ionized donor charge populating the conducting region of the device is difficult to estimate. Another parameter is the workfunction of the metal used in the gates, as it should give rise to offsets on the gate voltages.

These unknown parameters determine the charge distribution in the sample. We argue that, to a large extend, these parameters can be inferred from a combination of measurements and PESCA calculation. For an in depth discussion of the modeling and the comparison to experiments we refer to Chapter 6. Here we concentrate on how the PESCA calculation can be made and fitted to the experimental findings

To study the split wire device with PESCA we use a 2 stages model, a high temperature model and a low temperature model. At high temperature, the surface of the stack is modeled by an equipotential at a voltage V_{sc} and the metallic gate by an equipotential at a voltage V_{off}. On the other hand we describe the dopant region by its concentration of dopants n_{dop}. If we were to use an equipotential V_{dop}, it would amount to supposing the phenomena of "Fermi level pinning" happens in the dopant region. We do not believe Fermi level pinning to take place here. We first solve the high temperature Poisson problem and calculate the distribution of charge $n_{\text{sc}}(x)$ at the surface. At low temperature, we freeze the surface charge. That is, we turn the surface from $\mathcal{D}$ to $\mathcal{N}$ with a distribution of charges given by $n_{\text{sc}}(x)$. Among the regions outside of the 2DEG, only the gates remain $\mathcal{D}$ at the low temperature model. We note n_s the bulk 2DEG density far away from the gates. Although n_s is calculated, it is easily measured experimentally with e.g. Hall resistance. One can adjust the dopant density to obtain the correct value of n_s.

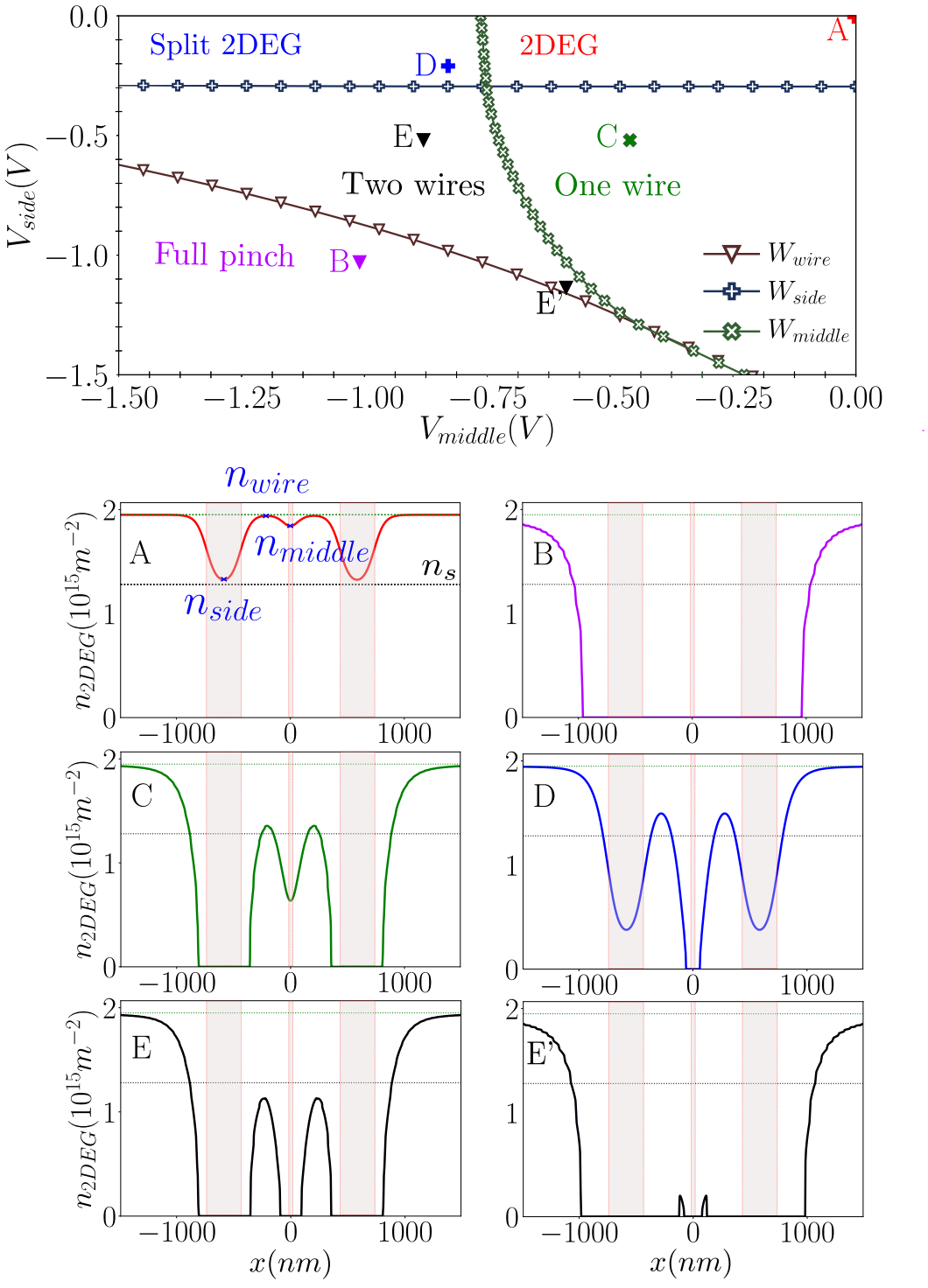


Figure 2.5: Top panel: PESCA Pinch-off phase diagram for the split wire device shown in Fig.2.1. $n_{dop} = 1.4310^{16}m^{-2}$, $n_s = 1.2810^{15}m^{-2}$ (dotted line), $V_{off} = -0.813V$ and $V_{sc} = -0.668V$. The different lines W_{wire}, W_{side} and W_{middle} separate the different regions A-E, see text. Bottom panels: 2DEG density profile calculated for the points A-E in the phase diagram. The labels n_{side}, n_{middle} and n_{wire} indicate the 2DEG density at specific points, respectively, underneath the side gate, the middle gate and in between, see Fig.2.1.

Fig.2.5 shows the PESCA "Pinch-off" phase diagram of the model for a particular set of parameters n_{dop}, V_{sc} and V_{off}. For each region, a density profile is shown in the bottom panels for a particular $(V_{\text{side}}, V_{\text{middle}})$ point. In region A, the 2DEG is present underneath all gates. Note however that even though no gate voltage is applied in point A, the density underneath the gates is reduced due to the workfunctions of the electrodes. In region B, the 2DEG is entirely depleted in the central part of the split wire.

When the side gate voltage is negative enough, the 2DEG underneath the side gates get depleted and the quantum wire is formed (region C). Upon further applying a negative voltage on the middle gate, one eventually cuts the wire into two, thus reaching the split wire regime (region E). In region D, only the gas underneath the central gate is depleted. The point of Pinch-off phase diagrams is that they can be easily measured experimentally. For example, they can be constructed by measuring the conductance between ohmic contacts separated by the different regions of the split wire device, see Fig.2.6 for an example. Then, each point in the diagram correspond to the gate voltages for which the conductance reaches zero. At the same time, the different lines of the phase diagram depend on the model parameters. Hence, pinch-off phase diagrams can put strong constraints on the underlying modeling.

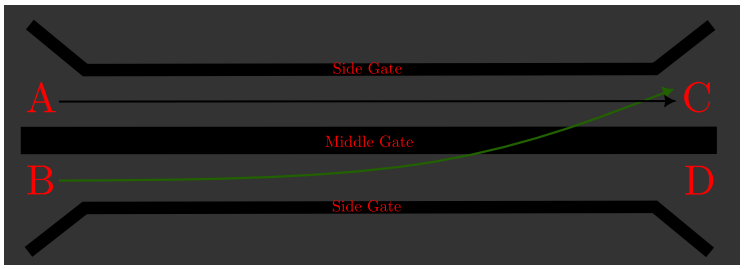


Figure 2.6: Sketch of the top view of the split wire device. In grey the 2DEG and black the gates. A, B, C and D indicate the electrodes. Current is sent through electrodes A or B, and received at C (D). If the device is in the one wire regime, then current from B to C will not be zero. If in the two wire regime, then B to C is zero.

Fig.2.7 shows the density at $x_{\text{wire}} = -225nm$ as a function of the gate voltage (the same voltage is applied to all the gates). First, these curves are not linear. This is a consequence of the self-consistent nature of PESCA. Indeed for a given, fixed $\mathcal{D}/\mathcal{N}$ partitioning, the density at every point depends linearly on the gate voltage. Hence the non-linearity is a consequence of the variation of this partitioning as some regions get depleted. The different curves of Fig.2.7 (black, green and blue) are calculated by varying the gate voltage applied at the high temperature stage of the calculation. This is aimed at mimicking a common experimental practice known as "bias cooling", where a positive bias is applied during the cooling of the sample. This bias attract electrons under the gate which in turn are screened by charges elsewhere

in the device. In our model, it is the surface charges that screen the electrons under the gate. At low temperature these screening charges get frozen, hence remain even if the bias voltage is removed. They contribute to depleting the 2DEG underneath the gate even at zero voltage. We find that this effect is qualitatively reproduced by our calculations. However, typically a one to one correspondence is found experimentally between the bias used during cooldown and the gate value at which pinch-off is found at low temperature [Pierre *et al.* 2022]. In our calculations the offset in the pinch-off value is only that of half the cooldown bias voltage. This points to the presence of additional screening effects in the donor layer not taken into account here. For instance, we have supposed that the donor ionization is fully frozen at high temperature.

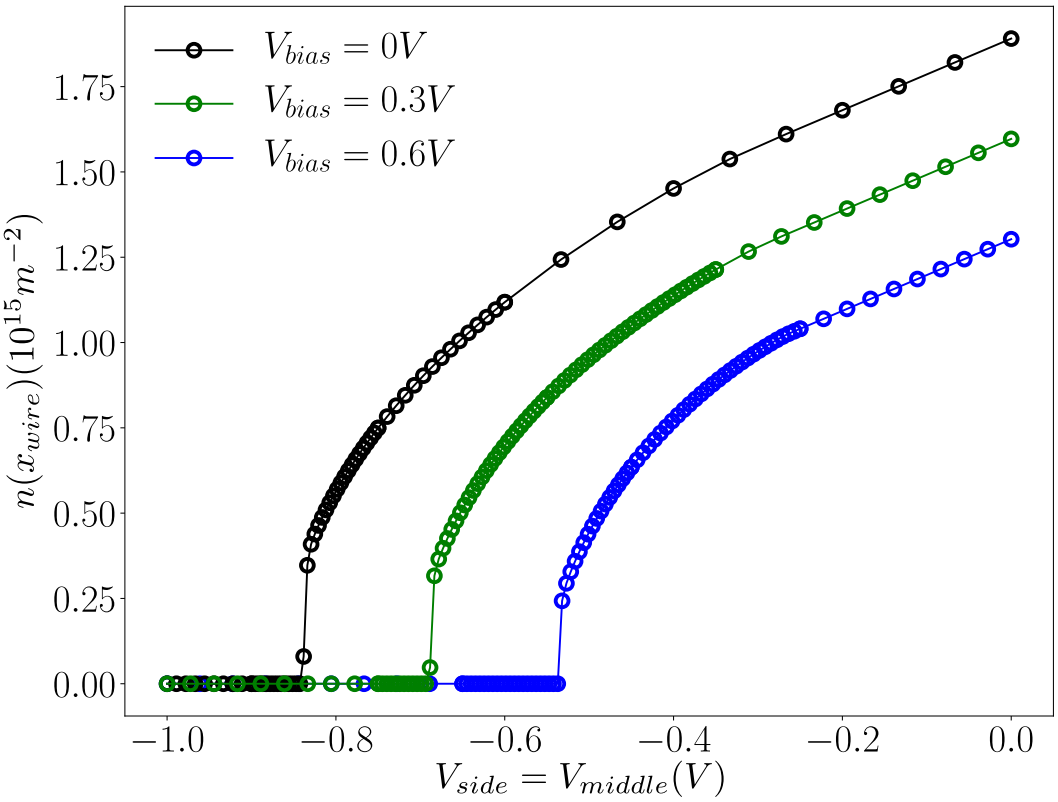


Figure 2.7: Effect of bias cooling. Density $n(x_{\text{wirel}})$ versus gate $V_{\text{side}} = V_{\text{middle}}$ at $x_{\text{wire}} = -225nm$ as a function of V_{bias}. The $n(V)$ profiles are all calculated at the low temperature stage but for different V_{bias} at the high temperature stage. The different curves correspond respectively to black for $V_{bias} = 0V$, green for $V_{bias} = 0.3V$ and blue for $V_{bias} = 0.6V$.

Fig.2.8 shows the pinch off voltage in a simpler geometry where only a single gate is present. When the gate width is larger than roughly 200-300 nm (about twice the distance from the 2DEG to the gate), one approaches the bulk value of the pinch-off (which can easily be calculated analytically). However, the Pinch-off value changes

quite strongly when the gate is made thinner. Since in actual devices this width may be as thin as 30 nm or perhaps even thinner, we find that the geometry must be accounted for precisely if one wants to make quantitative predictions.

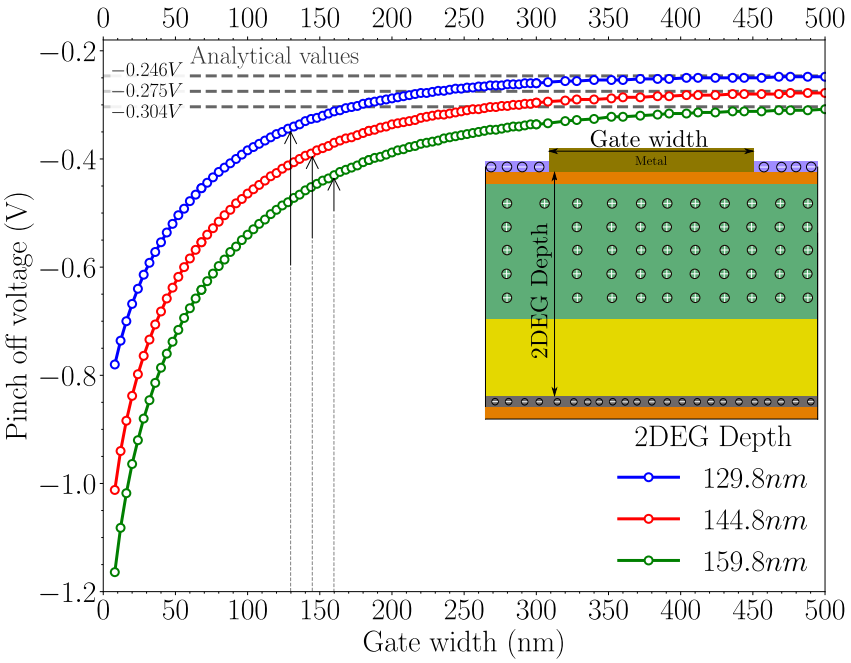


Figure 2.8: Pinch off voltage in a single gate geometry (see inset) versus gate width. The parameters are $n_{dop} = 1.43\ 10^{16} m^{-2}$, $V_{off} = -0.813V$ and $V_{sc} = -0.668V$. The different curves correspond to three different distances between the gate and the 2DEG obtained by increasing the thickness of the undoped $AlGaAs$ layer

2.4 Adjusting the Pinch-off phase diagram to experimental data

In this section we look at the inverse problem to the one from the preceding section: suppose that we are given an experimental Pinch-off phase diagram, how do we extract the microscopic parameters $n_{\rm dop}$, $V_{\rm sc}$ and $V_{\rm off}$ of the model ? Of course, one could design some optimization problem and minimize the difference between the target pinch-off phase diagram and the one calculated from PESCA. However such a step is rather computationally intensive and lacks physical insights. Besides, further insight can be obtained by using the particularly simple structure of PESCA in some limiting regimes, as we show below.

The first limit of interest corresponds to the region A of the phase diagram of Fig.2.5. In this region, where the gate voltages are only weakly negative, the 2DEG is depleted nowhere. Hence all the cells are $\mathcal{D}$ cells and as such no self-consistency is required. This is the fully metallic limit. In the Fully metallic limit, the 2DEG

Regions	PESCA		Region A Full metallic limit		Region B Depleted limit			
			s_{mid} s_{side}	s_{sc} s_{dop} s_{off}	s_{sc} s_{dop} s_{off}	U_{mid} U_{side}	U_{sc} U_{dop} U_{off}	
	High T	Low T	Low T	High & Low T	High T	Low T	High T	Low T
Surface charge	$\mathcal{D}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{N}$	$\mathcal{D}$	$\mathcal{N}$
Gates	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$
Dopants	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
2DEG	$\mathcal{D}/\mathcal{N}$	$\mathcal{D}/\mathcal{N}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{N}^*$	$\mathcal{D}/\mathcal{N}$	$\mathcal{D}/\mathcal{N}$

Table 2.2: Summary of the boundary conditions we fix at each region of the model throughout this chapter. $\mathcal{D}$ means the cells are of Dirichlet type. $\mathcal{N}$ means the cells are of Neumann type. $\mathcal{D}/\mathcal{N}$ means the cells can be of either Dirichlet or Neumann type.
* The 2DEG far from the side gates are of $\mathcal{D}$ type instead of $\mathcal{N}$ (See text)

density is the solution of a single linear system of the form of Eq.(2.10). From the additivity of different solutions, it results that $n(x)$ is a linear function of the different input parameters,

$$\begin{aligned} n(x) = \; & V_{\text{middle}} s_{\text{mid}}(x) + V_{\text{side}} s_{\text{side}}(x) \\ & + V_{\text{off}} s_{\text{off}}(x) + V_{\text{sc}} s_{\text{sc}}(x) + \frac{n_{\text{dop}}}{10^{16}} s_{\text{dop}}(x) \end{aligned} \tag{2.11}$$

where $s_{\text{mid}}(x)$, $s_{\text{side}}(x)$, $s_{\text{off}}(x)$, $s_{\text{sc}}(x)$ and $s_{\text{dop}}(x)$ are functions to be determined. Each function can be obtained from a single call to the linear solver. In other words we can get the full parametric dependance of $n(x)$ with respect to the input parameters. An example of the $s_{\text{mid}}(x)$ set of functions is given in Fig.2.9. The frontiers of the Pinch-off phase diagram that connect region A to another region can be obtained directly from Eq.(2.12). The frontier W_{middle} between region A and D is given from $n(x = x_{\text{middle}} \equiv 0) = 0$, which translates into

$$\begin{aligned} & V_{\text{middle}} s_{\text{mid}}(0) + V_{\text{side}} s_{\text{side}}(0) \\ & + V_{\text{off}} s_{\text{off}}(0) + V_{\text{sc}} s_{\text{sc}}(0) + \frac{n_{\text{dop}}}{10^{16}} s_{\text{dop}}(0) = 0 \end{aligned} \tag{2.12}$$

Similarly, the frontier W_{side} between region A and region C is given from $n(x = x_{\text{side}} \equiv -585nm) = 0$ in Eq.(2.12). If these frontiers are known experimentally, we directly obtain a set of linear constraints on the microscopic parameters that facilitate the fitting of the model to the experiments.

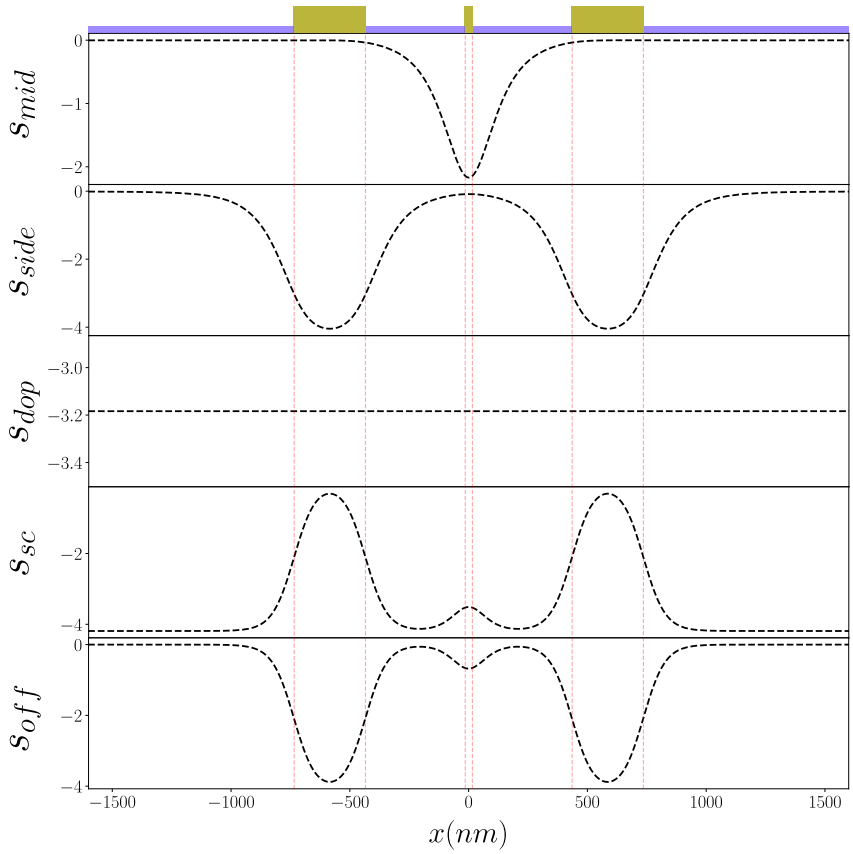


Figure 2.9: Functions $\mathbf{s}_{\text{mid}}(\mathbf{x})$, $\mathbf{s}_{\text{side}}(\mathbf{x})$, $\mathbf{s}_{\text{off}}(\mathbf{x})$, $\mathbf{s}_{\text{sc}}(\mathbf{x})$ providing the full solution of PESCA in region A of the phase diagram, see Eq.(2.12). The vertical red dotted lines show the positions of the electrostatic gates.

The calculation of the set of functions $s(x)$ require a little care with respect to the boundary conditions. For instance the workfunction V_{off} can naively be thought as a voltage which could be simply added to V_{side} and V_{middle}. However, V_{off} is present at high temperature where the surface charges are mobile (i.e. modeled by $\mathcal{D}$ cells) while the gate voltages V_{side} and V_{middle} are set at low temperature when the surface charges are frozen (i.e. modeled by $\mathcal{N}$ cells). Table 2.2 summarizes the boundary conditions we fix at each calculation we performed in this article. For instance, it shows that to calculate $\mathbf{s}_{\mathbf{mid}}$ we set the surface charge region to $\mathcal{N}$, the (middle and side) gates region to $\mathcal{D}$, the dopants region to $\mathcal{N}$ and the 2DEG region to $\mathcal{D}$.

Another interesting limit is the "depleted limit". It corresponds to region B of the Pinch-off diagram. In this regime there is no electron gas inside the central split wire region. However, some electron gas remains in the region $|x| > |x_{\text{dep}}|$ far away from the central region. In PESCA, the position x_{dep} is determined self-consistently. However, it is interesting to consider the case where x_{dep} is fixed, as it allows us to study the screening role of the electron gas far away from the central region. For a

fixed value of $x_{\rm dep}$, the potential inside the wire is given by

$$\begin{aligned} U(x) = \; & V_{\rm middle}U_{\rm mid}(x) + V_{\rm side}U_{\rm side}(x) \\ & + V_{\rm off}U_{\rm off}(x) + V_{\rm sc}U_{\rm sc}(x) + \frac{n_{\rm dop}}{10^{16}}U_{\rm dop}(x) \end{aligned} \tag{2.13}$$

where $U_{\rm mid}(x)$, $U_{\rm side}(x)$, $U_{\rm off}(x)$, $U_{\rm sc}(x)$ and $U_{\rm dop}(x)$ are functions to be determined. To determine the frontier between region B and the other regions from Eq.(2.13), one can use

$$\exists\, x, \text{ such that } |x| < |x_{\rm dep}| \text{ and } \; U(x) = 0. \tag{2.14}$$

Contrary to region A, in region B the position of depletion is not directly under the gates, it depends instead on the ratio $V_{\rm side}/V_{\rm middle}$. Therefore the curve delimiting the region B of the Pinch-off diagram, i.e. $W_{\rm wire}$, is not a simple straight line. Much like for $s(x)$, calculating the set of $U(x)$ functions require special care with respect to the boundary conditions, as summarized in Table 2.2. This is specially true at the 2DEG. At low temperature the 2DEG is made of $\mathcal{N}$ cells while at high temperature it is made of $\mathcal{D}$ cells. Therefore, calculating, e.g. $U_{\rm sc}(s)$, requires solving *two* different linear systems: one to determine $s_{\rm sc}$ and a second to calculate the effect of $s_{\rm sc}$ when the 2DEG is depleted.

Fig.2.10 shows the PESCA phase diagram together with the predictions obtained from the full metallic limit and the depleted limit. The prediction from the full metallic limit is essentially exact. Its appeal steems from the fact that (in contrast to a PESCA calculation) we have the full dependance of $n(x)$ on $n_{\rm dop}$, $V_{\rm sc}$ and $V_{\rm off}$. The prediction from the depleted limit depends strongly on the value of $x_{\rm dep}$. This means the occupied 2DEG outside of the split wire region plays an important screening role, which cannot be ignored if one wants to make quantitative predictions.

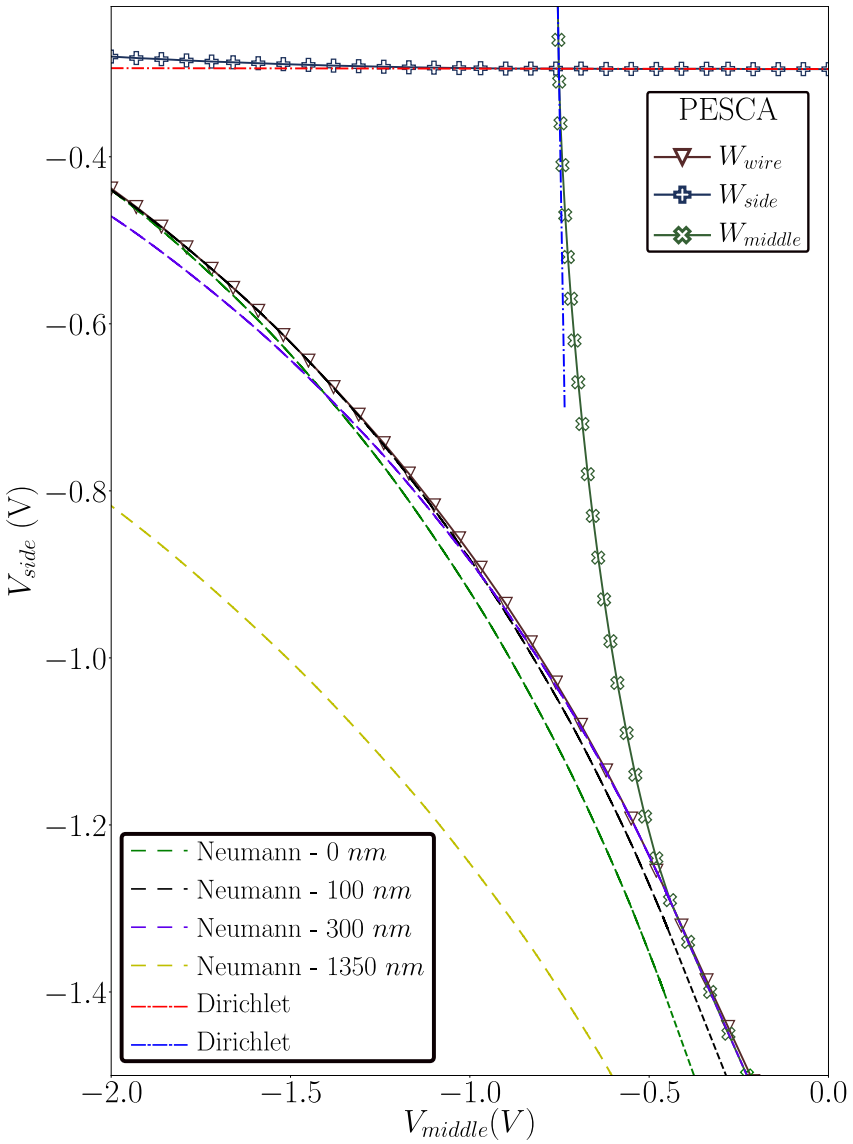


Figure 2.10: Comparison between PESCA Pinch-off phase diagram with the full metallic limit and the depleted limit. The full lines correspond to PESCA results obtained with $n_{dop} = 1.4310^{16}m^{-2}$, $n_g = 1.9510^{15}m^{-2}$, $n_s = 1.2810^{15}m^{-2}$, $V_{off} = -0.813V$ and $V_{sc} = -0.668V$. The red and blue dotted lines correspond to calculations in the full metallic limit. The green, black, violet and yellow dotted lines correspond to calculations in the depleted limit as one varies x_{dep}. Here x_{dep} delimits the distance between the last depleted 2DEG cell and the side gates.

2.5 Extension of PESCA to the integer quantum Hall effect

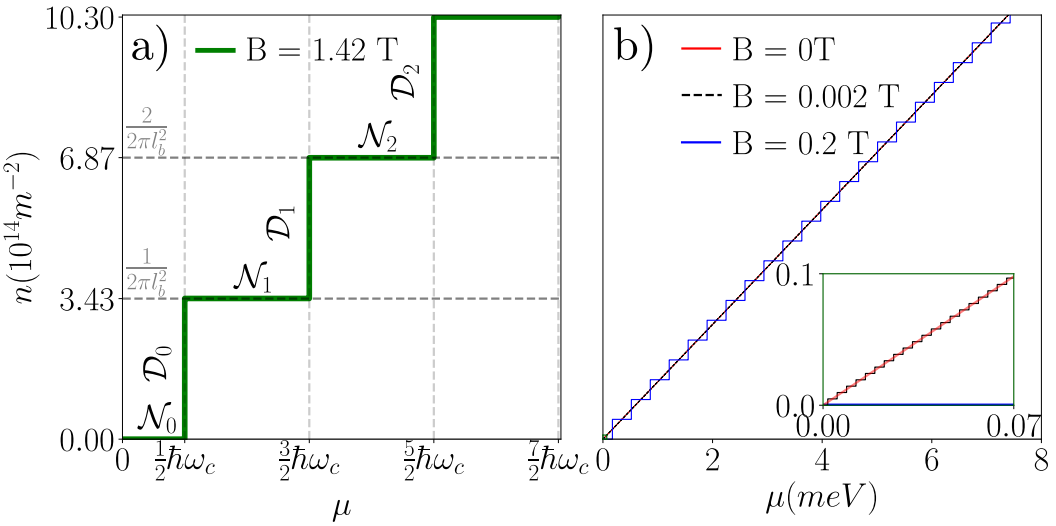


Figure 2.11: a) Thomas Fermi ILDOS for $B = 1.4T$. One can see the μ and n regions for which a site is of $\mathcal{N}_i$ or $\mathcal{D}_i$ type up to $i = 2$. For example, if $1/2\hbar\omega_c < \mu < 3/2\hbar\omega_c$ then the site is of $\mathcal{N}_1$ type with $n = 1.81.10^{14}m^{-2}$. b) Thomas Fermi ILDOS for $B = 0.2T$ and $B = 2mT$ in blue and black respectively. The ILDOS in red corresponds to Thomas-Fermi at $B = 0T$. At the energy scales relevant to the problem in question there is no discernable difference between the $B = 2mT$ and the Thomas-Fermi ILDOS

We end this chapter with the first extension of the PESCA algorithm to more complex situations. As we have seen, PESCA essentially relies on the ILDOS having two different branches. However, it is straightforward to generalize the algorithm s.t. it can handle an ILDOS with a discrete number of branches. Such a situation arises naturally in the quantum Hall regime when the 2DEG is put under a perpendicular magnetic field. The role of electron-electron interactions is known to have drastic effects in this regime and in particular leads to the reconstruction of the edge states into compressible and incompressible stripes [Chklovskii *et al.* 1992a, Chklovskii *et al.* 1992a, Chklovskii *et al.* 1993]. The density of states of a bulk 2DEG in the quantum Hall regime is highly non-linear. It is made of delta function peaks in the (highly degenerate) Landau levels separated by insulating regions of vanishing density of states. The resulting ILDOS is a step like function as shown in Fig.2.11 (a). In Fig.2.11 (a), $\hbar\omega_c = 2.45$ meV, with ω_c the cyclotron frequency and $l_B = 21.6$ nm the magnetic length. To solve the self consistent problem corresponding to this ILDOS, one updates the PESCA algorithm by introducing the cells $\mathcal{D}_p$ and $\mathcal{N}_p$. The integer $p \geq 0$ indicates on which

branch of the ILDOS the cell is. The $\mathcal{D}_p$ cells are Dirichlet cells with fixed potential $U = \hbar\omega_c(p + 1/2)$ while the $\mathcal{N}_p$ cells are Neumann cells with fixed density $n = p/(2\pi l_B^2)$. Step III of the PESCA algorithm also needs to be extended as follows. For the $\mathcal{D}_p$ cells,

- if $n_i > (p + 1)/(2\pi l_B^2)$ then the cell must change to $\mathcal{N}_{p+1}$
- if $n_i < (p/(2\pi l_B^2))$ then the cell must change to $\mathcal{N}_p$
- otherwise the cell remains in $\mathcal{D}_p$.

For the $\mathcal{N}_p$ cells,

- if $U_i \geq -\hbar\omega_c(p + 1/2)$ then the cell must change it to $\mathcal{D}_{p-1}$.
- if $U_i \leq -\hbar\omega_c(p + 3/2)$ then the cell must change it to $\mathcal{D}_p$.
- otherwise the cell remains in $\mathcal{N}_p$.

Fig.2.12 shows the convergence of the the $\mathcal{N}_p$ / $\mathcal{D}_p$ partition for the extended PESCA. On the top Fig.2.12 an initial zero magnetic field PESCA (two-branch ILDOS Fig.2.2 (c)) calculation is performed before switching on the magnetic field and performing a second calculation with extended PESCA. On the bottom Fig.2.12 the iterations are directly done with a finite magnetic field. We find that both schemes converge although starting with a simple PESCA initialization seems to speed up the convergence. This is particularly the case at very low magnetic fields where many Landau levels are filled. The converged partition gives direct access to the so-called compressible ($\mathcal{D}_p$) and incompressible ($\mathcal{N}_p$) stripes [Chklovskii *et al.* 1992a, Chklovskii *et al.* 1992a, Chklovskii *et al.* 1993].

Fig.2.13 shows the converged profile of electric field (top) and density (bottom). We refer to [Armagnat & Waintal 2020, Armagnat *et al.* 2019] for a discussion of the physics of the different stripes present in the system. We note that, even though the magnetic field is fairly high B=1.42T (cyclotron energy $\hbar\omega_c = 2.45meV$), the density profile is only weakly affected by the field with respect to PESCA ($\approx 5\%$). Also, the modification of the electric potential of a few mV is small compared to the larger scales at play within the rest of the sample, of the order of 1V. These few mV might be associated to important physics, but on the other hand they only weakly affect the Pinch-off gate voltages. This confirms that PESCA is an appropriate level of approximation for reconstructing the charge distribution inside the sample.

This can be further seen in Fig.2.14 where a similar calculation is performed at low $B = 0.2T$ and very low $B = 0.002T$ magnetic field. We also add a direct Thomas Fermi calculation corresponding to the ILDOS of Fig.2.2b that was performed using the algorithm of [Armagnat *et al.* 2019]. The direct Thomas-Fermi calculation is indistinguishable from the $B = 0.002T$ one. This can also be seen in Fig.2.11b, where the different ILDOS are plotted. Besides being an independent validation of the calculation (the two algorithms only share the Poisson solver), this points to

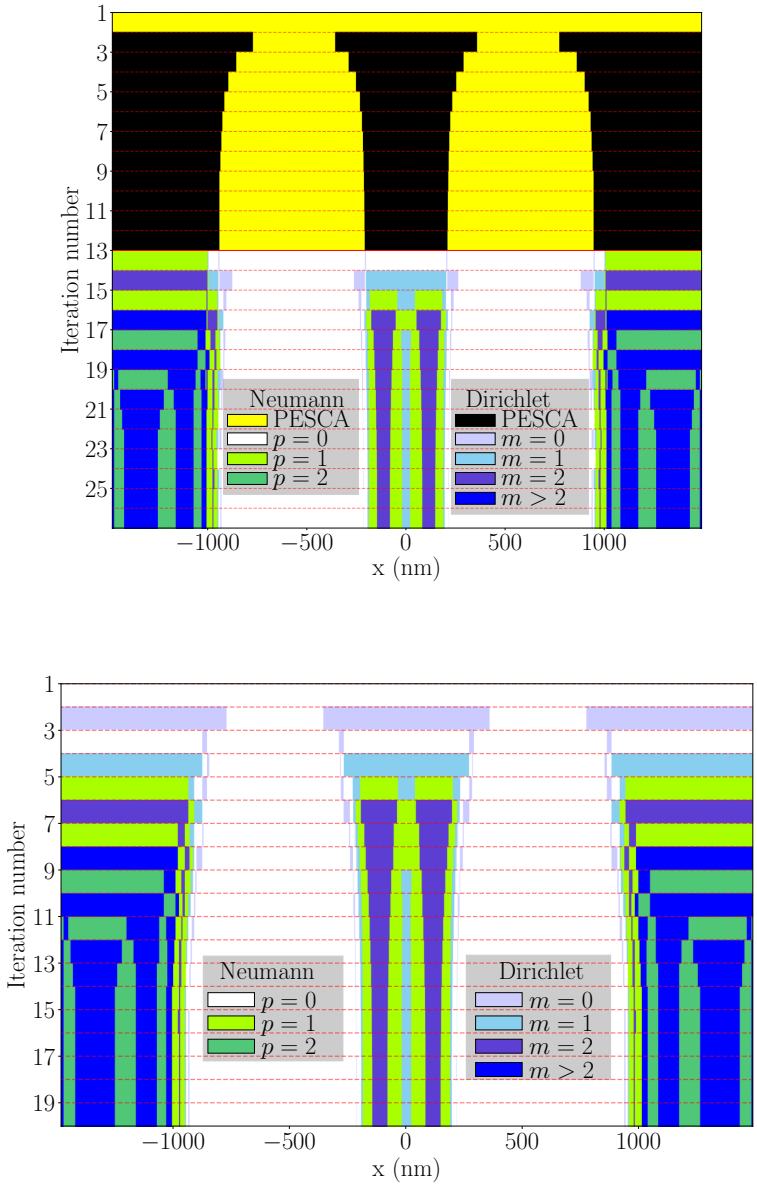


Figure 2.12: $\mathcal{N}_p/\mathcal{D}_p$ partitioning as a function of the iteration number for the results in Fig.2.13. (a) Up to iteration 12 the PESCA approximation was used. At the latter iteration the PESCA $\mathcal{N}/\mathcal{D}$ partitioning is stable. The resulting potential and charge profile are then used as input to iteration 13. For the latter and onwards thomas-fermi approximation is used for $B = 1.42T$. The yellow and black regions correspond to $\mathcal{N}$ and $\mathcal{D}$ cells under PESCA approximation. Under thomas fermi the cells located within the first five landau levels are shown in varying degrees of green for $\mathcal{N}_p$ cells and blue for $\mathcal{D}_p$ cells. p is the filling factor s.t. $n = p/(2\pi l_b)$ and m is defined s.t. $p = (1 + m)/2\hbar\omega_c$. (b) The thomas fermi approximation for $B = 1.42T$ is used for all iterations.

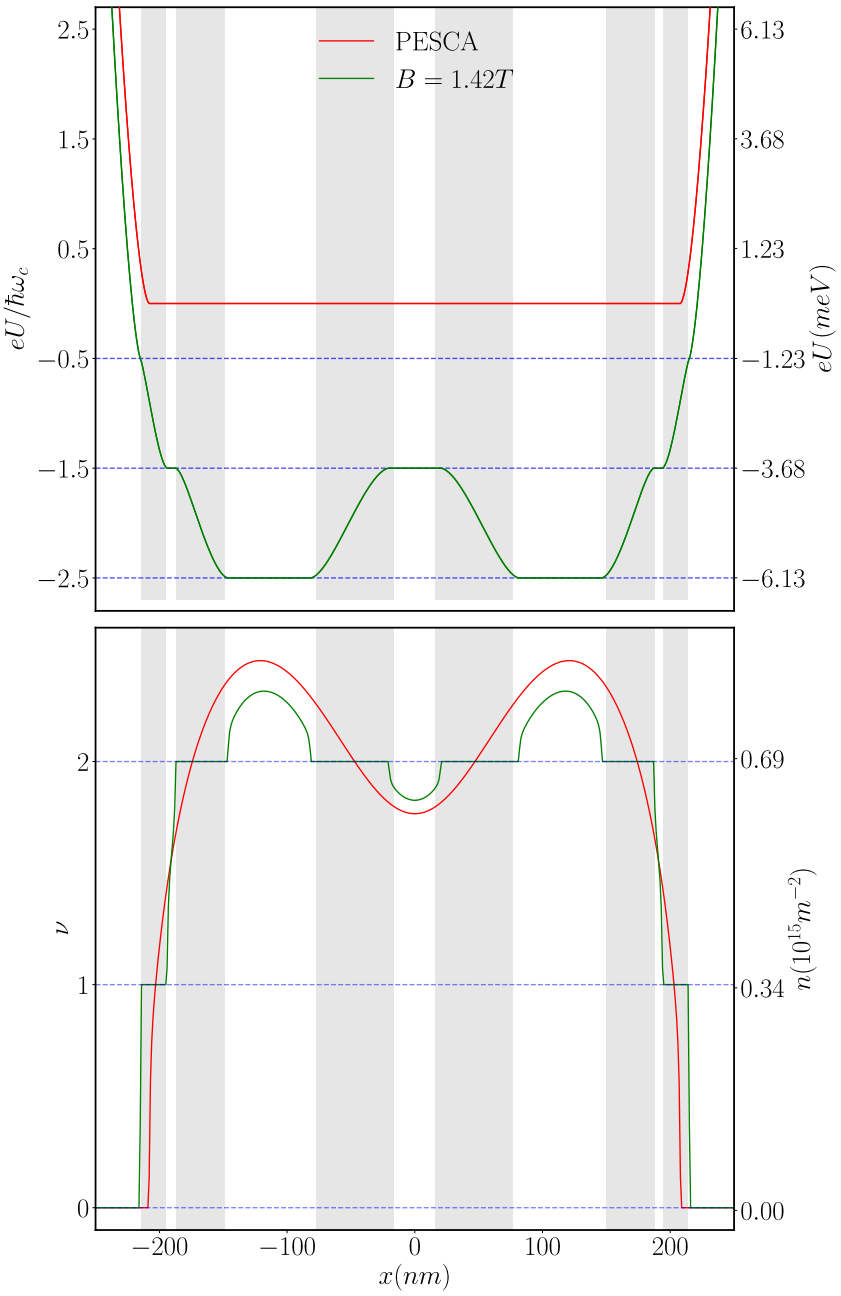


Figure 2.13: Top panel : Chemical potential profile at the 2DEG as a function of x. The profile in green was obtained for $B = 1.4T$ with the ILDOS on Fig.2.11 (a). In red the profile was obtained with the PESCA approximation, i.e. the ILDOS on Fig.2.2 (c). The gray (white) stripes correspond to the incompressible (compressible) region. Lower panel : Charge density profile for $B = 1.4T$ and with the PESCA approximation. At the regions where the charge density is depleted, the red curve converges towards the green one. For this calculation $V_{side} = -1V$ and $V_{mid} = -0.35V$.

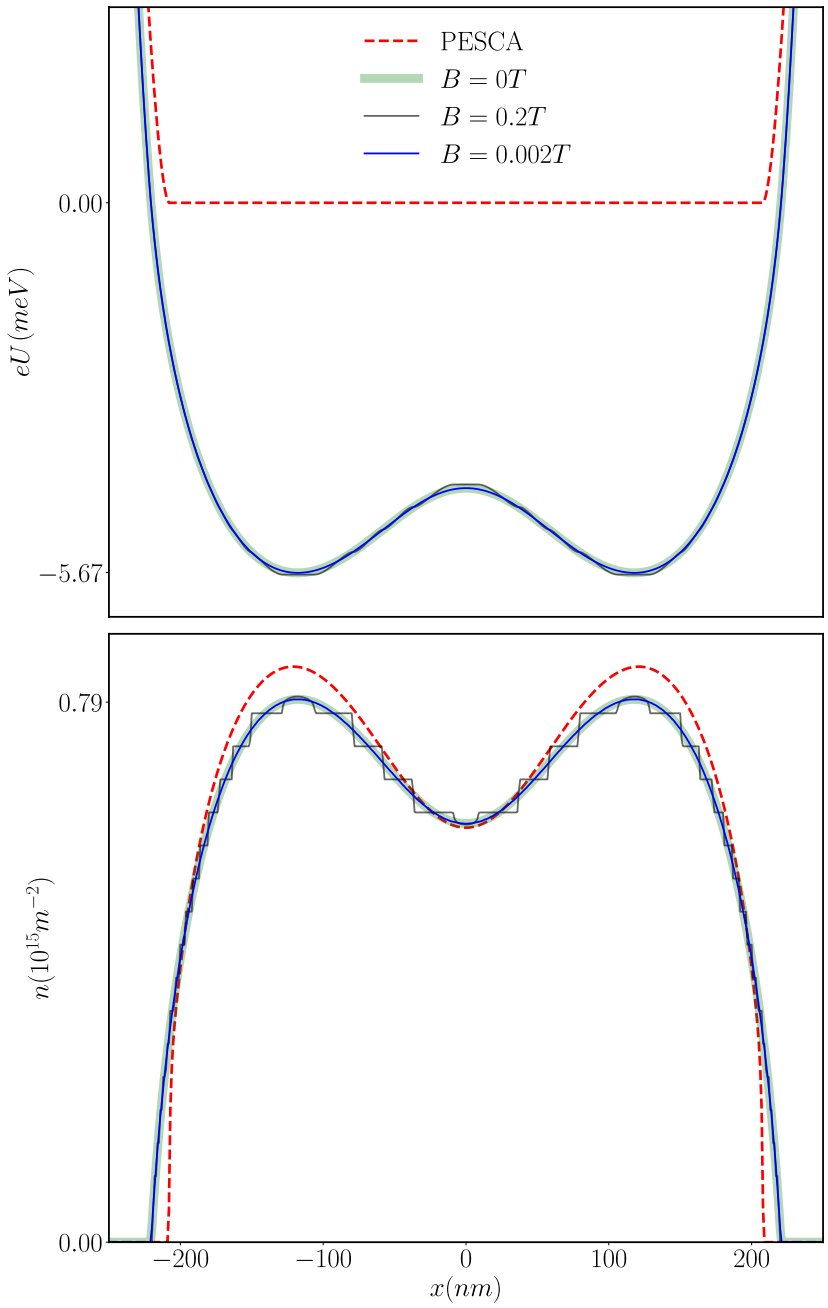


Figure 2.14: Top panel : Chemical potential profile at the 2DEG as a function of x. In blue and grey the profile was obtained for with $B = 2mT$ and $B = 0.2T$ respectively . They were calculated using the ILDOS on the right panel of Fig.2.11. In red the profile was obtained with the PESCA approximation, i.e. the ILDOS on Fig.2.2 (c). The thicker green line corresponds to a Thomas Fermi calculation for $B = 0T$ using the algorithm of [Armagnat *et al.* 2019]. Lower panel : Charge density profile. For this calculation $V_{side} = -1V$ and $V_{mid} = -0.35V$.

the fact that *any* ILDOS could be approximated by step like functions with $\mathcal{N}_p$ / $\mathcal{D}_p$ partitions, which would allow the PESCA algorithm to be extended to arbitrary Thomas-Fermi approximations (i.e. with fixed ILDOS).

2.6 Conclusion

In this chapter we have first used the small parameter in the SCQE problem (ratio of the geometric to quantum capacitance) to develop the PESCA. We have shown that PESCA is enough to reconstruct the charge distribution in a sample, a prerequisite for predictive simulations. Second we have extended the algorithm we used for PESCA towards solving the SCQE with a step-like ILDOS, that of the integer quantum hall effect. We have shown its results to be equivalent to the Thomas-Fermi calculations of [Armagnat *et al.* 2019].

The PESCA algorithm, and its extension to the IQHE ILDOS, are guaranteed to converge and form a first step into a new type of solver for SCQE problems. The next step is to extend $\mathcal{N}_p$, $\mathcal{D}_p$ branches to allow for arbitrary slopes of the branches and address any piece-wise linear ILDOS. This is addressed in the next chapter.

Chapter 3

Solving the self-consistent quantum electrostatic problem

In this chapter we discuss the method we have developed to solve the self-consistent quantum electrostatic problem. It consists on approximating the SCQE into a NLH equation, and then correcting the approximation iteratively. The local density of states, updated at the end of each iteration, contains key information about the energy dependance of the quantum problem. Therefore it is a fundamental quantity to be transferred as input to the new iteration of the algorithm. More precisely, at each iteration we solve a NLH equation, and the LDOS allows us to isolate the non-linearities of the problem and address them accordingly.

In Section 3.1 we first formulate a discrete version of the continuous SCQE defined in the Introduction. Then in Section 3.2 we introduce the quantum adiabatic approximation, it approximates the SCQE problem into a NLH equation. In Section 3.3 we detail the algorithm to solve the SCQE and in Section 3.4 how to solve the NLH equation using two different algorithms. In particular, we show that one of them is guaranteed to converge for any non-linear ILDOS. Both schemes are inspired from the PESCA algorithm of Chapter 2. Lastly in Section 3.5 we illustrate both our algorithms by solving the NLH for three different Piecewise-ILDOS and one continuous ILDOS.

3.1 Discrete SCQE problem definition

Here we formulate a discrete version of the continuous SCQE problem defined in Section 1.2. We shall start with the electrostatic problem an then move on to the quantum problem. The Poisson equation can be discretized in many ways using e.g. finite difference or finite elements. In this thesis we use finite volume discretization. Finite volume has the advantage of defining a physically valid discrete electrostatic problem for any grid coarseness. It does so by ensuring electric flux conservation, i.e. charge conservation, within the simulated region.

The electrostatic problem is defined in terms of capacitance matrix C_{ij}. We discretize the simulation volume into a set of cells $\mathcal{C}_i$ centered on point $\vec{r}_i$, see Figure 3.1 for an example. Each cell has a few neighbors j at distance $d_{ij} = |\vec{r}_i - \vec{r}_j|$ and S_{ij} is the surface that separates them. We use Voronoi cells so the surface is planar. Gauss theorem for a given cell takes the form :

$$\sum_j \Phi_{ij} = -eQ_i \tag{3.1}$$

where Q_i is the total number of charge inside the cell ($Q_i = -1$ for one electron, $+1$ for one hole), it writes :

$$Q_i(\mu) = -\int_{\mathcal{C}_i} d\vec{r}\ [n(\vec{r}) - n_d(\vec{r})], \tag{3.2}$$

and Φ_{ij} the flux of the electric field :

$$\Phi_{ij} = \int_{S_{ij}} \varepsilon(\vec{r})\vec{\nabla}U(\vec{r}) \cdot \vec{n}\ \mathrm{d}S \tag{3.3}$$

where $\vec{n}$ is the unit vector perpendicular to the surface S_{ij} (parallel to $\vec{r}_i - \vec{r}_j$ for Voronoi cells). We approximate $\vec{\nabla}U(\vec{r}) \cdot \vec{n}$ with $(U_j - U_i)/d_{ij}$ where U_i the electric potential at the center of cell i. We arrive at :

$$\sum_j C_{ij}U_j = Q_i(\mu) \tag{3.4}$$

with the capacitance matrix given by

$$C_{i\neq j} = -\frac{\varepsilon_{ij}S_{ij}}{ed_{ij}} \leq 0 \tag{3.5}$$

for neighboring cells and

$$C_{ii} = -\sum_{j(i)} C_{ij} \geq 0 \tag{3.6}$$

for the diagonal part, where $j(i)$ stands for the neighbors of cell i ($C_{ij} = 0$ otherwise). Beware of the slight abuse of notations since we have incorporated the electric charge e inside the definition of C_{ij}. The dielectric constant ε_{ij} is averaged over neighboring sites according to :

$$\varepsilon_{ij} = \frac{2\varepsilon_i\varepsilon_j}{(\varepsilon_i + \varepsilon_j)} \tag{3.7}$$

where ε_i is the dielectric constant inside cell i.

Finally, defining the discrete LDOS as :

$$\rho_i(E) = \int_{\mathcal{C}_i} d\vec{r}\ \rho(\vec{r}, E), \tag{3.8}$$

we have for electrons

$$\frac{\partial Q_i}{\partial \mu} = -\rho_i(\mu) \leq 0. \tag{3.9}$$

Regarding the quantum problem, there are also various ways to obtain a discretized model, e.g. tight-binding model, finite difference from a k.p Hamiltonian

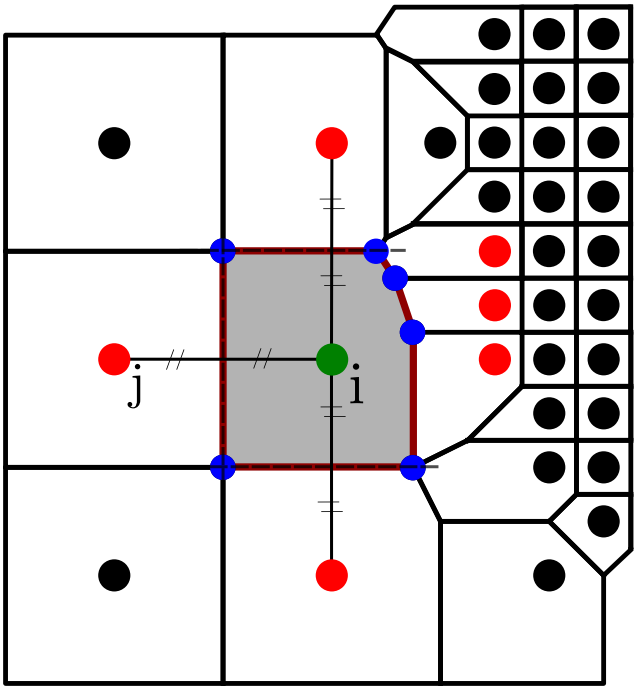


Figure 3.1: Schematics of a 2D voronoi diagram. Each voronoi cell is delimited by the black lines. For the cell i, its central point is in green, its region (in grey) is delimited by the red lines. In blue are the vertices of the cell. In red are the central point of the cells j first neighbor of cell i. Here d_{ij} is defined as the distance from the green point i to the red point in cell j. The S_{ij} is defined by the red line separating cell i from cell j, a line since it is a 2D voronoi diagram (for a 3D diagram it would be a surface). This figure was generated using *PESCADO* , see Chapter 4.

etc... Here we suppose that this discretized model is obtained on the same grid as the electrostatic model, possibly with additional local degrees of freedom on each site (spin, orbitals, different atoms per unit cell etc.) in the usual framework of quantum transport [Groth *et al.* 2014]. $\Psi_{\alpha E}$ becomes a vector with $\Psi_{\alpha E}(i)$ the sub-vector on site i (whose components span the local degrees of freedom). The discrete Schrödinger equation reads:

$$\sum_j [(H_0)_{ij} - eU_i\delta_{ij}\mathbb{1}]\Psi_{\alpha E}(j) = E\Psi_{\alpha E}(i) \tag{3.10}$$

where $\mathbb{1}$ is the identity matrix of the local degrees of freedom. With these notations, the discrete LDOS reads :

$$\rho_i(E) = \frac{1}{2\pi}\sum_\alpha \Psi_{\alpha E}(i)^\dagger \Psi_{\alpha E}(i). \tag{3.11}$$

Together, the above set of equations form the discrete SCQE problem.

3.2 Approximating the SCQE as a NLH equation

We already wrote a continuous form of the NLH problem in Eq.(4.3), when we described the Predictor-Corrector approach. In this section we shall give a more detailed explanation on how to approximate the discrete SCQE problem into a generalized (discrete) NLH equation. In order to achieve this, we introduce the quantum adiabatic approximation (QAA). In essence, the QAA assumes the LDOS $\rho_i(E)$ independent of the electrostatic potential up to a small shift $e\delta U_i$. It can be seen as a generalization of the Thomas-Fermi approximation, in [Armagnat *et al.* 2019] we discuss it in more detail.

Suppose that for an electrostatic potential U_i we have computed the LDOS $\rho_i(E)$. Now consider a different potential $U_i' = U_i + \delta U_i$. QAA approximates the corresponding LDOS as :

$$\rho_i'(E) = \rho_i(E + e\delta U_i) \tag{3.12}$$

i.e. it supposes that the electrostatic potential simply shifts the different energy bands. The approximation is exact in the limit where the difference of potential δU_i varies infinitely smoothly with position. Let E_F be the electro-chemical potential of the quantum problem, Eq.(3.4) reduces to an equation for U_i' :

$$\sum_j C_{ij}U_j' = Q_i(E_F + eU_i' - eU_i) \tag{3.13}$$

which is the NLH equation mentioned earlier. Eq.(3.13) can be recast into a slightly more convenient form :

$$\sum_j C_{ij}\delta U_j = Q_i(E_F + e\delta U_i) + n_i \tag{3.14}$$

where the source term n_i is given by $n_i = -\sum_j C_{ij}U_j$. Hereafter, we ignore the n_i term that we absorb in the definition of Q_i. In contrast to the SCQE problem for which, as far as we are aware, there is no proof of convergence of the various iterative schemes that are commonly used, in Section 3.4.1 we show that the NLH equation has very nice properties both theoretically and in practice.

It is also possible to get a linear version of the NLH. For small δU_i, we can linearize Eq.(3.14) and obtain a discretized version of the LH equation :

$$\sum_j C_{ij}\delta U_j = Q_i(E_F) - e\rho_i(E_F)\delta U_i. \tag{3.15}$$

The LH equation is a (sparse) linear equation that can be solved by standard numerical approaches. As we shall see in Section 3.4.2, all algorithms developed in this thesis eventually reduce to sequentially calling the LH equation solver. Lastly, if $U_i = 0$ and the LDOS is the bulk DOS (ρ^{bulk}) at the Fermi energy (site independent), then the LH equation reduces to a generalization of the well-known Thomas Fermi approximation :

$$\sum_j C_{ij}U_j = Q_i(E_F) - e\rho^{\text{bulk}}U_i. \tag{3.16}$$

3.3 The secret lies in the density of states

Considering we can solve the NLH equation, how to use it to find the solution to the SCQE problem ? The answer is the following iterative scheme :

1. Start with an initial LDOS $\rho_i(E)$. A common choice is to use the bulk DOS of the material on all sites $\rho_i(E) = \rho^{\text{bulk}}(E)$. Alternatively, one can solve the quantum problem with $U_i = 0$ on all sites.
2. Given this LDOS, solve the NLH problem and obtain U_i.
3. Given this potential U_i, solve the quantum problem and obtain a new LDOS.
4. Repeat steps (2) and (3) until convergence (no mixing scheme has been necessary in our experience so far).

The main difference between this strategy and standard iterative schemes is that the input of the electrostatic problem is not the density but the LDOS. The local density of states *contains information* about the energy dependance of the quantum problem. Therefore the NLH equation already captures the main sources of non-linearities of the problem. It *knows* about the LDOS at the Fermi level, about the presence of gaps in the spectrum, etc ... In other words, it knows about the cusps in the ILDOS that cause havoc in many modern approaches that solve the NLH relying solely on the charge density. As an added bonus, if we use the bulk DOS at initialization, then the first solution of the NLH equation (iteration 1 step 2) gives

a generalization of the Thomas-Fermi potential of the problem. The rest of this chapter is consecrated to solving the NLH.

3.4 Solving the NLH equation

In this section we shall explain how to solve the NLH formulated by Eq.(3.14). First we discuss some interesting properties of this equation. In particular, we show the existence of a functional $F(\{U_i\})$ that that (i) admits the solution of the NLH equation as its global minimum and (ii) has no local minimum or saddle points. The existence of such functional guarantees the unconditional convergence of a scheme seeking to solve NLH by minimizing F. With this in hand, we then propose two algorithms to solve the NLH equation.

3.4.1 Properties of the NHL equation

The NHL equation (3.13) takes the generic form,

$$\sum_j C_{ij}U_j = Q_i(U_i) \tag{3.17}$$

with the following properties:

- C_{ij} is symmetric.
- C_{ij} is semi-definite positive.

  Indeed, $\sum_j C_{ij} = 0$ and $C_{i\neq j} \leq 0$ imply that $\forall U_i$,

$$\sum_{ij} U_i C_{ij} U_j = -\frac{1}{2}\sum_{i\neq j}(U_i - U_j)^2 C_{ij} \geq 0 \tag{3.18}$$

- $Q_i(E)$ is a decreasing function of the energy ($dQ_i/dE = -\rho_i(E)$ and the LDOS is positive).
- $\rho_i(E) = 0$ for $E < E_B$, the bottom of the lowest band.

The goal of this section is to construct a functional $F(\{U_i\})$ that (i) admits the solution of the NLH equation as its global minimum and (ii) has no local minimum or saddle points.

We define $F(\{U_i\})$ as,

$$F(\{U_i\}) = \frac{1}{2}\sum_{ij} U_i C_{ij} U_j - \int_{-\infty}^{U_i} dE\ Q_i(E). \tag{3.19}$$

F is the sum of two convex functions and is therefore convex itself. The gradient of F is given by

$$\frac{\partial F}{\partial U_i} = \sum_j C_{ij} U_j - Q_i(U_i), \tag{3.20}$$

and its Hessian is,

$$\frac{\partial^2 F}{\partial U_i \partial U_j} = C_{ij} + \rho_i(U_i)\delta_{ij} \tag{3.21}$$

i.e. a semi-definite positive matrix. This functional admits a global minimum which is the solution of the NLH equation. When all bands are empty, NLH can have degenerate global minimums that differ by a global shift of the potential $U_i \to U_i + U$. However, when at least one band is partially occupied, one can explicitly check that there is a unique global minimum. Indeed, if U_i^* is a minimum of F then for any variation δU_i^* around this minimum,

$$\begin{aligned} F(\{U_i^* + \delta U_i^*\}) = F(\{U_i^*\}) + \frac{1}{2}\sum_{ij} \delta U_i^* C_{ij} \delta U_j^* \\ - \int_{U_i^*}^{U_i^*+\delta U_i^*} dE\ [Q_i(E) - Q_i(U_i^*)] \end{aligned} \tag{3.22}$$

which is the sum of two positive terms. $F(\{U_i^* + \delta U_i^*\}) = F(\{U_i^*\})$ implies that δU_i^* does not depend on i (first term) and the local density of states vanish on all sites at $E = U_i$.

To summarize, we are in a very comfortable situation fo solve the NLH equation: it is the global minimum of a convex functional of which we know both the gradient and the Hessian. This is a much more satisfactory situation than the SCQE problem we started with.

3.4.2 Practical algorithms for solving the NLH equation

We now turn to the practical schemes used to solve the NLH equation. In practice, there remains one small but crucial difficulty that we must treat with care: the LDOS is usually a smooth function of energy but it has cusps or discontinuities at the band edges. To illustrate this problem, let's consider a simple quadratic band $H \propto p^2$. The corresponding DOS has the form $\rho(E) \propto E^{d/2-1}$ and has a discontinuous derivative ($d = 3$), is discontinuous ($d = 2$) and even diverges ($d = 1$) at the band edge $E = 0$. The existence of these singular points make traditional gradient descent sort of methods unstable unless one treats these points explicitly. This is particularly true in low dimension.

We handle this difficulty by explicitly tracking the problematic points in energy on each site. Physically, it means that we are tracking the regions of space that are depleted (due to e.g. a gate) and those that are not. Below, we explain the algorithm to deal with this aspect. Once this is taken care of, we have found most approaches

to converge quickly to the solution. We present two of them: the piecewise Newton-Raphson algorithm (a variation of the eponym algorithm) and the piecewise linear Helmholtz algorithm.

3.4.2.1 Breaking the ILDOS into piecewise-smooth regions

The input of all solvers is the ILDOS $Q_i(E)$. On each site i, we break the energy regions into different intervals $[E_i^\alpha, E_i^{\alpha+1}]$ where the energies E_i^α mark the position of the (possibly) singular point in energy of the ILDOS ($E_i^0 = -\infty$ by convention). A different function $Q_i^\alpha(E)$ is used on each subinterval. For example, in the case of the quadratic band, one would use $E^0 = -\infty$, $E^1 = 0$ and $E^2 = +\infty$. For $E \in [-\infty, 0]$, one would use $Q_i^0(E) = 0$ while inside the band $E \in [0, +\infty]$, one uses $Q_i^1(E) \propto E^{d/2}$. These sub-intervals are used in the two different NLH solvers described below.

3.4.2.2 Piecewise Newton-Raphson algorithm

We first describe the Piecewise Newton-Raphson algorithm, a simple adaptation of the Newton-Raphson algorithm. The algorithm works as follows:

1. We initialize the potential on all sites U_i. A typical choice is $U_i = 0$ everywhere. On each site, we identify the energy interval $\alpha(i)$ such that $U_i \in [E_i^\alpha(i), E_i^{\alpha(i)+1}]$.

2. We linearize the NLH equation at U_i and form Eq.(3.15). This is a linear equation that can be solved e.g. with a sparse LU solver such as MUMPS [Amestoy *et al.* 2001, Amestoy *et al.* 2006]. We obtain U_i'.

3. for all sites, if $U_i' \in [E_i^\alpha(i), E_i^{\alpha(i)+1}]$ then we set $U_i \to U_i'$. However, if $U_i' < E_i^\alpha(i)$ then the corresponding point has switched to the previous branch (e.g. the corresponding site is depleted). We set $U_i \to E_i^\alpha(i)$ and $\alpha(i) \to \alpha(i) - 1$. Likewise, if $U_i' > E_i^{\alpha(i)+1}$ then we switch to the next branch. We set $U_i \to E_i^{\alpha(i)+1}$ and $\alpha(i) \to \alpha(i) + 1$.

4. We repeat steps (2) and (3) until convergence.

Keeping track of the index $\alpha(i)$ of the solution on each site is the key to prevent the band edges from destabilizing the algorithm.

3.4.2.3 Piecewise Linear Helmholtz algorithm

In practice the piecewise Newton-Raphson algorithm is very stable in almost all the situations that we have encountered. When the ILDOS is particularly non-linear, it can nevertheless fail to converge. In this situation, the following, somewhat slower but very stable, "Piecewise Linear Helmholtz algorithm" usually solves the problem.

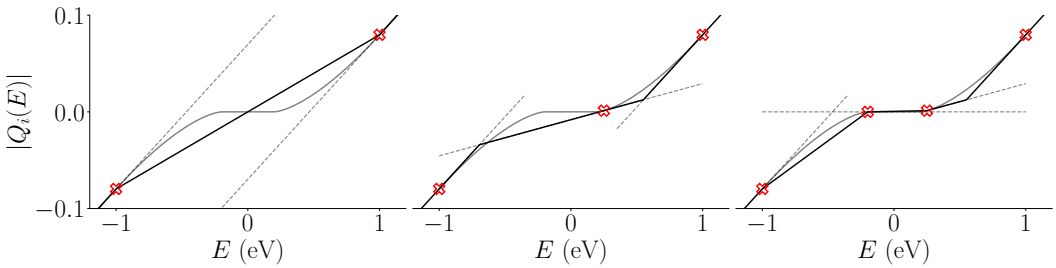


Figure 3.2: Construction of the piecewise linear ILDOS $\bar{Q}_i(E)$ (black line) from the continuous one $Q_i(E)$ (thin grey line). The red points correspond to the initial list of points E_i^α. Left: the two tangents (dotted line) do not intersect inside the segment, one interpolates linearly between the two red points. Middle panel: the tangents do intersect, one uses two linear segments (the two tangents) to interpolate between the two red points. Right panel: when a new red point is inserted, the piecewise linear ILDOS $\bar{Q}_i(E)$ is updated.

This algorithm can also be used in cases where the quantum solver provides the ILDOS $Q_i(E)$ but not (its derivative) the LDOS $\rho_i(E)$.

The Piecewise Linear Helmholtz algorithm uses a piecewise-linear-ILDOS $\bar{Q}_i(E)$ that approximates the actual ILDOS $Q_i(E)$. Here, the points E_i^α are not only used to separate smooth regions; they also correspond to a discretization of the ILDOS. The algorithm to construct $\bar{Q}_i(E)$ works as follows (see Fig.3.6 for an illustration):

- On each point E_i^α, we set $\bar{Q}_i(E_i^\alpha) = Q_i(E_i^\alpha)$.
- On each point E_i^α, we calculate the tangent $y = Q_i(E_i^\alpha) - \rho_i(E_i^\alpha)(E - E_i^\alpha)$.
- If the tangent at point E_i^α and $E_i^{\alpha+1}$ *do not* intersect inside the segment $[E_i^\alpha, E_i^{\alpha+1}]$ (left panel of Fig.3.6), we set $\bar{Q}_i(E)$ to simply interpolate linearly between E_i^α and $E_i^{\alpha+1}$:

$$\bar{Q}_i(E) = Q_i(E_i^\alpha) + (E - E_i^\alpha)\frac{Q_i(E_i^{\alpha+1}) - Q_i(E_i^\alpha)}{E_i^{\alpha+1} - E_i^\alpha} \tag{3.23}$$

- If the tangent at point E_i^α and $E_i^{\alpha+1}$ *do* intersect (middle panel of Fig.3.6), we set $\bar{Q}_i(E)$ to be equal to the tangents up to the intersection point. To do so we insert an extra temporary point $E_i^{\alpha'}$ at the intersection in between E_i^α and $E_i^{\alpha+1}$.

The idea of the algorithm is to solve the piecewise-linear-NLH equation defined by $\bar{Q}_i(E)$ and at the same time refine our description of $\bar{Q}_i(E)$ so it becomes a fair approximation of $Q_i(E)$. We refine $\bar{Q}_i(E)$ by inserting new points E_α. The algorithm works as follows:

1. We initialize the potential on all sites U_i. We also initialize the points E_i^α. We construct the corresponding piecewise linear ILDOS $\bar{Q}_i(E)$. On each site, we identify the energy interval $\alpha(i)$ such that $U_i \in [E_i^{\alpha(i)}, E_i^{\alpha(i)+1}]$.

2. We solve the linear Helmholtz equation associated with $\bar{Q}_i(E)$. We obtain U_i'.

3. For all sites, if $U_i' \in [E_i^{\alpha(i)}, E_i^{\alpha(i)+1}]$ then we set $U_i \to U_i'$. If $U_i' < E_i^\alpha(i)$ then the corresponding point has switched to the previous branch. We set $U_i \to E_i^{\alpha(i)}$ and $\alpha(i) \to \alpha(i) - 1$. Likewise, if $U_i' > E_i^{\alpha(i)+1}$ then we switch to the next branch. We set $U_i \to E_i^{\alpha(i)+1}$ and $\alpha(i) \to \alpha(i) + 1$.

4. If we have not switched branch, then the new point U_i' is used to update our piecewise linear ILDOS $\bar{Q}_i(E)$. The value U_i' is added to the list of energies $\{E_i^\alpha\}$ cutting the previous interval $[E_i^{\alpha(i)}, E_i^{\alpha(i)+1}]$ into two subintervals $[E_i^{\alpha(i)}, U_i']$ and $[U_i', E_i^{\alpha(i)+1}]$. We reconstruct $\bar{Q}_i(E)$ using this new point (see the right panel of Fig.3.6 for an example).

5. We repeat steps (2)-(4) until convergence.

In this algorithm, the intervals E_i^α are not static, they evolve along the iterations. Furthermore, since we split the intervals $\alpha(i)$ at the position of the previous iteration energy solution, $\bar{Q}_i(E)$ will be more refined near the actual solution of the NLH problem. Note that if $Q_i(E)$ is actually piecewise linear, then step (4) above is omitted.

3.5 Solving the NLH problem for a hexagonal nanowire device

In this section, we illustrate the different algorithms described above on a practical use case. More examples can be found, see the split wire example in Chapter 4, the quantum point contact example of Chapter 2, the application to graphene pn-junction [Flór *et al.* 2022] of Chapter 5 and the simulations of scanning gate microscopy [Percebois *et al.* 2023] of Chapter 7. All the numerics shown here were performed using the open source software *PESCADO* described in Chapter 4.

3.5.1 Piecewise linear ILDOS

We consider an infinitely long hexagonal nanowire. We suppose it is invariant by translation and therefore model it in two dimension. A back gate (Dirichlet boundary condition at $U_i = -6V$) positioned below the nanowire depletes it while a top gate (Dirichlet boundary condition at $U_i = +2V$) placed over two of the nanowire edges attract electrons into the system. The LDOS vanishes outside the nanowire.

We start with the simplest situation where the ILDOS is piecewise-linear with only two branches: an horizontal and a vertical branch. Fig. 3.3a shows the ILDOS and Fig. 3.3b the geometry together with the final result (color plot shows the charge

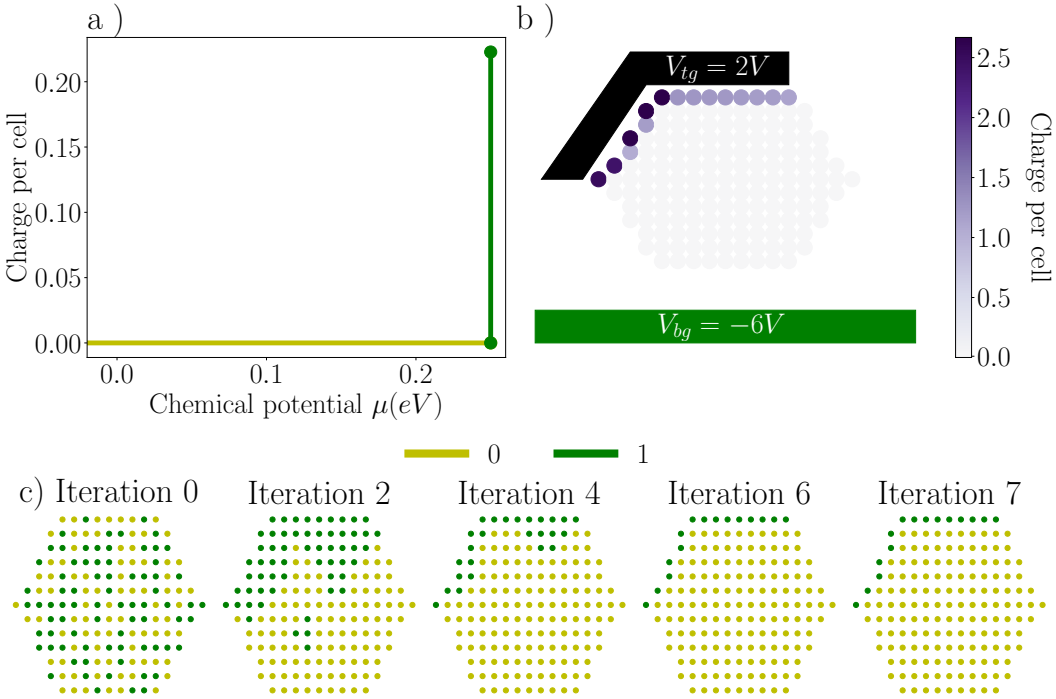


Figure 3.3: Solving the NLH equation using the piecewise linear Helmholtz algorithm: PESCA. a) Input ILDOS : a piecewise linear ILDOS with two branches. This corresponds to the PESCA approximation. b) Colormap and schematics of a side view of the the hexagonal nanowire with top (black) and back gate (green). The color code corresponds to the charges in each cell of the nanowire. c) Convergence of $\alpha(i)$ on each site versus different iterations. $\alpha = 0$ (yellow, first branch) or $\alpha = 1$ (green, second branch). The initial configuration $\alpha(i)$ is random here.

on each cell). In fact, this NLH problem corresponds to the PESCA approximation studied in Chapter 2.

For this first example we have used the piecewise Linear Helmholtz algorithm. However, since the ILDOS is *actually* piecewise linear, step (4) can be omitted. There is no need to refine something which is already exact. Fig. 3.3c shows the different iterations until convergence (iteration 6) for an initially random $\alpha(i)$ configuration (iteration 0). We observe that convergence is reached in very few (six) iterations. Contrary to the Newton-Raphson algorithm, the convergence here is not a continuous process. It is only when all the sites are in the correct branch that the piecewise Linear Helmholtz algorithm has converged (there is no precision criteria). The electrons only accumulate on a single layer of cells, this is due to the PESCA approximation. Since the density of states is infinite (vertical branch of the ILDOS), the corresponding Thomas-Fermi length vanishes and charge accumulates purely on the surface (purely metallic limit).

In our second example we replace the vertical line of the ILDOS (PESCA) by a line of finite slope:

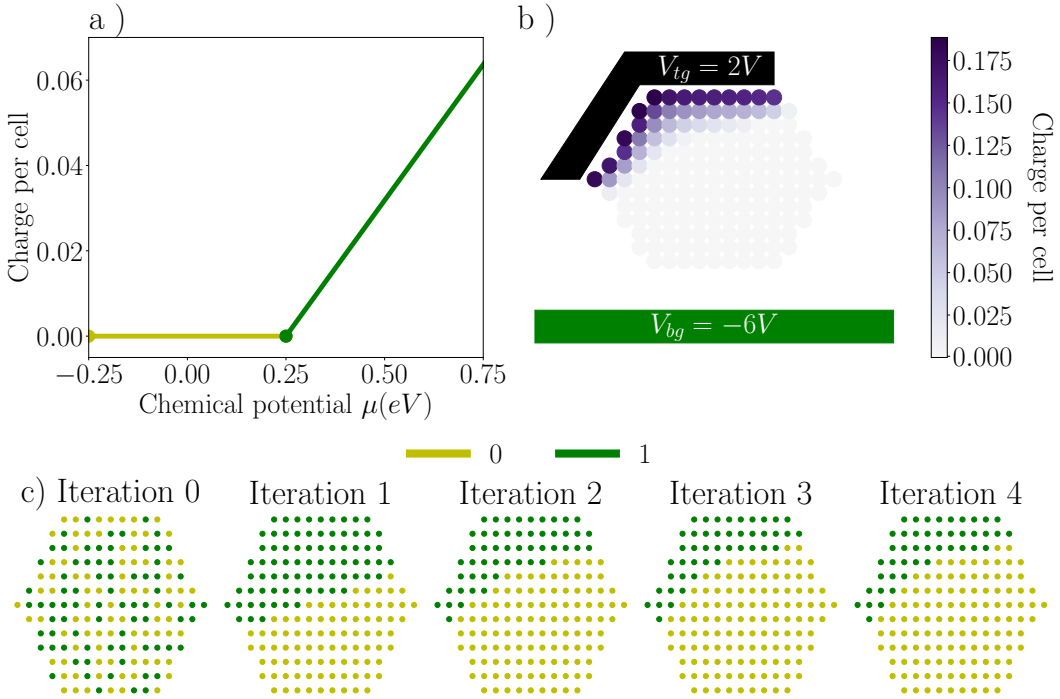


Figure 3.4: Solving the NLH equation using the piecewise linear Helmholtz algorithm: Thomas-Fermi. a) Input ILDOS: a piecewise linear ILDOS with two branches. This corresponds to the Thomas-Fermi approximation. b) Colormap and schematics of a side view of the the hexagonal nanowire with top (black) and back gate (green). The color code corresponds to the charges in each cell of the nanowire. c) Convergence of $\alpha(i)$ on each site versus different iterations. $\alpha = 0$ (yellow, first branch) or $\alpha = 1$ (green, second branch). The initial configuration $\alpha(i)$ is random here.

$$Q_i(E) = \begin{cases} 0 & \forall \quad E < 0.25eV \\ \rho E & \forall \quad E \geq 0.25eV \end{cases} \tag{3.24}$$

This corresponds to the Thomas-Fermi approximation. Fig. 3.4 shows the results. Convergence is even faster than in PESCA, it is more physically accurate and all of it at no additional computing cost. Thomas Fermi usually leads to a good, quantitative, description of the electronic density. The main difference with PESCA is that the finite density of states means the electric field can now penetrate inside the wire over a finite (Thomas-Fermi) length (see Fig. 3.4b).

In the last example of this series, we use an ILDOS with three branches. The first describes the valence band, the second the gap and the third the conduction band of a semi-conductor, see Fig.3.5. We have,

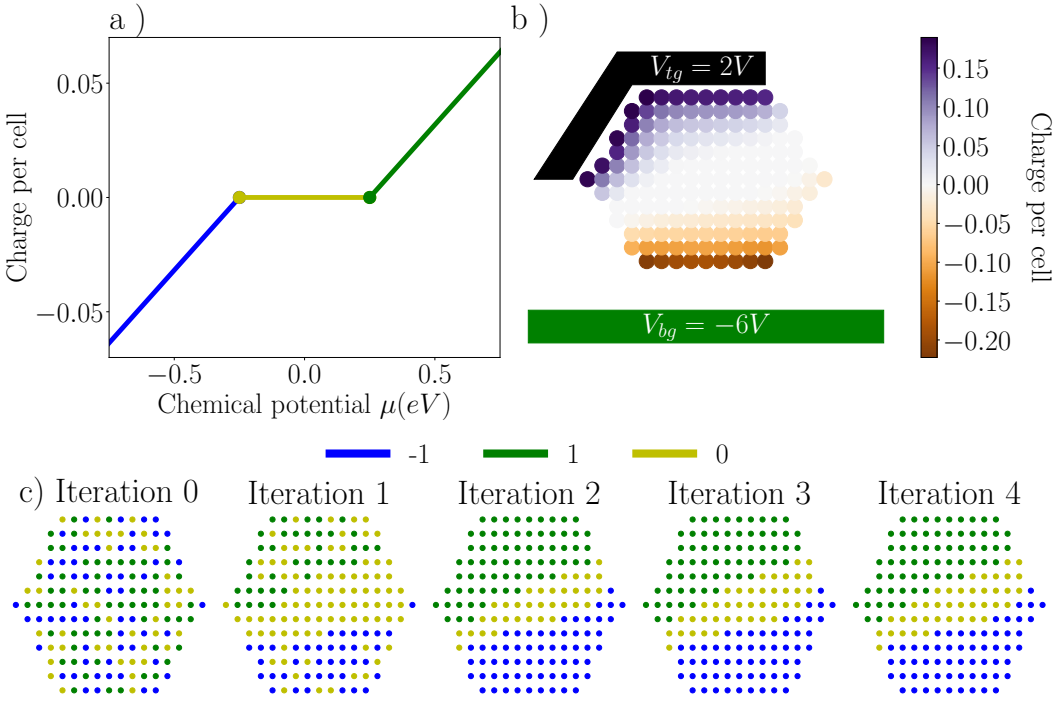


Figure 3.5: Solving the NLH equation using the piecewise linear Helmholtz algorithm: Valence-Conduction band model. a) Input ILDOS: a piecewise linear ILDOS with three branches. They correspond respectively to the valence band, the gap and the conduction band. b) Colormap and schematics of a side view of the the hexagonal nanowire with top (black) and back gate (green). The color code corresponds to the charges in each cell of the nanowire. c) Convergence of $\alpha(i)$ on each site versus different iterations. $\alpha = 0$ (yellow, first branch) or $\alpha = 1$ (green, second branch). The initial configuration $\alpha(i)$ is random here.

$$Q_i(E) = \begin{cases} \rho E & \forall & E \leq -\Delta \\ 0 & \forall & -\Delta \leq E \geq \Delta \\ \rho E & \forall & E \geq \Delta \end{cases} \tag{3.25}$$

Due to the valence band and the large negative voltage applied to the back gates, it is now possible to attract holes at the lower part of the nanowire (in red, see Fig.3.5b).

3.5.2 Continuous ILDOS

We now turn to a genuine NLH equation where the ILDOS varies continuously. We describe the valence and conduction bands using a free 3D density of states:

$$Q_i(E) = \begin{cases} -a|E + \Delta|^{3/2} & \forall & E \leq -\Delta \\ 0 & \forall & -\Delta \leq E \geq \Delta \\ a|E - \Delta|^{3/2} & \forall & E \geq \Delta \end{cases} \tag{3.26}$$

Fig.3.6 illustrates the Piecewise Linear Helmholtz algorithm for this ILDOS. The upper panels show the evolution of the charge in the nanowire at different iterations. The middle and lower panels show the corresponding piecewise linear ILDOS $\bar{Q}_i(E)$ on two different representative sites (green and black cross in the upper panel). Fig.3.7 shows te charge profile evolution with iteration. After just two iterations, the result is indistinguishable from the converged result. On each plot in Fig.3.6 (b) and (c), the small circle correspond to the new solution U_i' obtained after the call to the LH solver. This new solution is used to improve $\bar{Q}_i(E)$. Observe how these new points accumulate close to the final solution (right panel) such that, in fine, $\bar{Q}_i(E)$ is a extremely good approximation of $Q_i(E)$ for E close to the solution $E = U_i$. On the last column the results have been zoomed to show the (tiny) difference between $\bar{Q}_i(E)$ and $Q_i(E)$ close to the solution.

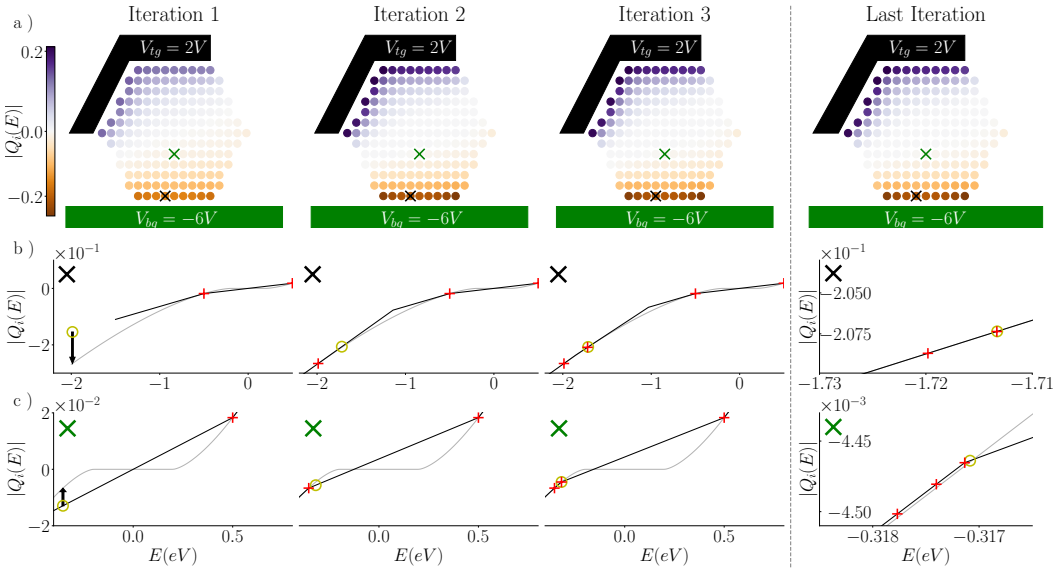


Figure 3.6: Solving the NLH equation using the piecewise linear Helmholtz algorithm: continuous ILDOS. (a) Charge profile for the first three and last iterations. (b) Evolution of $\bar{Q}_i(E)$ (black line) for same iterations and for the site tagged by a black cross in (a). Thin grey line: continuous ILDOS $Q_i(E)$ that is being approximated. Red crosses: points E_i^α added to the list, yellow circle: value U_i' given by the LH solver. (c) Same as (b) for the site tagged by a green cross in (a)

3.6 Conclusion

In this chapter we have explained how to solve the SCQE problem. We first applied the quantum adiabatic approximation to the SCQE problem to transform it into a NLH equation. Then, we have introduced two iterative schemes, the Piecewise Newton-Raphson and the Piecewise Linear Helmholtz algorithm. The first solves the NLH equation quickly and works for most cases. The second is slightly slower,

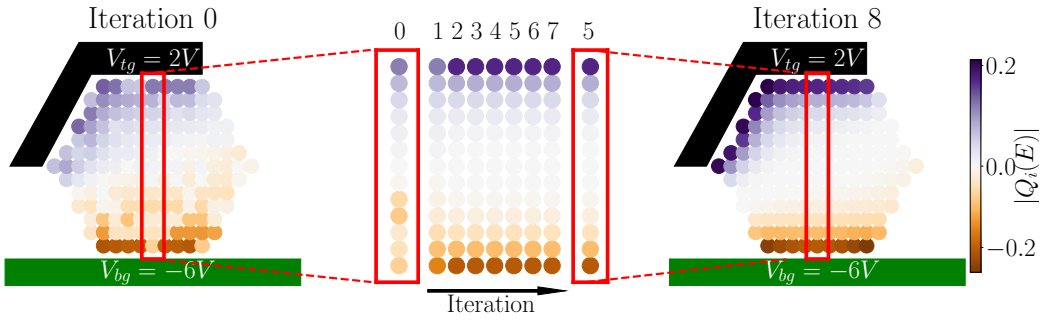


Figure 3.7: Evolution of the charge profile as a function of iteration for the continuous ILDOS of Eq.(3.26). The colormap on the left corresponds to the charge profile at iteration zero. It was obtained by solving the Helmholtz equation with a random initial guess for the ILDOS interval. The colormap on the right corresponds to the converged solution of the self consistent problem. The data on the middle corresponds to the charge value at $x = 0$ for each iteration of the solver. One notices that after the second iteration the result becomes indistinguishable from the converged result.

however its convergence is guaranteed. The key difference of our methods compared to traditional approaches is that we use the local density of states as input to the electrostatic problem instead of the charge density. Then, we have applied the piecewise linear algorithm for three examples where we solve a NLH problem defined with a piecewise linear ILDOS. We have also solved a NLH problem with a continuous ILDOS using both methods and compared their performance. For an example of a self-consistent quantum electrostatics calculation obtained using the Piecewise Newton Raphson algorithm, see Figure 4.4. In the next chapter we shall explain in detail the linear-helmholtz equation solver, *PESCADO* - the backbone of both algorithms.

Chapter 4

PESCADO : An open source software

The two algorithms we have developed in Chapter 3 rely on the existence of a LH equation solver. Therefore, in this chapter we shall explain *PESCADO* , an open source python library for solving electrostatic problems. It solves the LH problem and implements the Piecewise Newton-Raphson and Piecewise Linear Helmholtz algorithms. Overall, *PESCADO* offers:

- A lightweight geometrical engine. The first step to solve a partial differential equation is to describe the geometry of the system. For example, defining the shapes and positions of the different layers of materials, charge dopants, metallic gates, etc .... This is usually done using a complex hierarchy of tools for defining volumes, surfaces, edges and points. The choice made in *PESCADO* was to use a very light, yet surprisingly powerful, solution: for each volume, the user defines a function of space $\vec{r} = (x, y, z)$ that returns 1 if $\vec{r}$ is inside the volume and zero outside (what we call a "Shape"). For instance to define a spherical gate, one associates it with a function $f(\vec{r})$ returning 1 if $x^2+y^2+z^2 < 1$. All surfaces, edges and points are defined implicitly. Different Shapes can be combined through the usual logical operators (and, or, xor...) to create more complex Shapes. We have found that the shape system is an interesting compromise between expressivity and effectiveness. In practice it easily covered all the geometries that we have encountered while being very lightweight to use in practice. *PESCADO* geometrical engine is not specific to electrostatics and could be used for other problems.

- A finite volume mesher. The second step of the solver is to discretize the geometry. *PESCADO* uses finite volume which is particularly well adapted for electrostatics. Finite volume provides a valid, even for a coarse, discrete electrostatic problem. In practice, one uses a list of points $\vec{r}_i$ and *PESCADO* constructs the Voronoi cells centered around these points, together will the needed information to discretize the Poisson equation (volume of a cell, list of its neighbors with the corresponding surfaces of contact...). There are well established general algorithms (and libraries) to perform these computations for an arbitrary set of points $\vec{r}_i$. However, these calculations can be computationally expensive, time and memory wise. In many situations, it is enough to use a regular grid where the Voronoi cell can be computed analytically. *PESCADO* intends to get the best of both worlds by providing algorithms to combine, in

a single mesh, the different discretization techniques in different regions of the simulation. *PESCADO* mesher is also not specific to electrostatics.

- An electrostatic solver. In its basic form, it solves the Poisson equation

$$\nabla \cdot (\varepsilon(\vec{r})\nabla U(\vec{r})) = -en(\vec{r}), \tag{4.1}$$

and computes the electric potential $U(\vec{r})$ for a given input density of charge $n(\vec{r})$ (with $\varepsilon(\vec{r})$ the dielectric function). An important feature of this solver is that it also solves the Helmholtz equation *i.e.* one can replace the right hand side of Eq.(4.1) with $n(\vec{r}) \to n(\vec{r}) + \rho(\vec{r})U(\vec{r})$ where $\rho(\vec{r})$ is the local density of states at the Fermi level. The solver supports arbitrary Dirichlet and Neumann boundary conditions: on each cell the unknown variable can either be the potential (Neumann cells, the standard type) or the density (Dirichlet cells, for these cells the electric potential is an input). The solver works in one, two and three dimensions.

- A NLH solver. A quantum system is described by its LDOS $\rho(\vec{r}, E)$. When this system is at equilibrium, described by a electro-chemical potential E_F (the Fermi level), one has $\mu(\vec{r}) \pm eU(\vec{r}) = E_F$, with $\mu(\vec{r})$ the chemical potential (the minus sign applies to electrons, plus for holes). The electronic density is given by,

$$n(\vec{r}) = \int_{-\infty}^{\mu(\vec{r})} \rho(\vec{r}, E)dE \tag{4.2}$$

so that the electrostatic problem becomes the NLH equation,

$$\nabla \cdot (\varepsilon(\vec{r})\nabla U(\vec{r})) = \mp e \int_{-\infty}^{E_F \mp eU(\vec{r})} \rho(\vec{r}, E)dE \tag{4.3}$$

PESCADO solves this equation robustly, with theoretically guaranteed convergence for any arbitrary input $\rho(\vec{r}, E)$, see Section 3.4.1. The main source of non-linearity in Eq.(4.3) are the cusps of the LDOS, at *e.g.* band edges. *PESCADO* ensures convergence by treating theses cusps explicitly, see chapter 3.

At the time of writing, *PESCADO* does not provide an API for solving the full self-consistent quantum problem. More precisely, it does not implement a tool to calculate the LDOS. This is by design, since there are many techniques to calculate the LDOS for a given electric field. Each method has its place depending on the physics being modeled. However, *PESCADO* has been designed to integrate seamlessly with the software Kwant [Groth *et al.* 2014]. Hence performing such calculations is relatively straightforward. From an initial LDOS (typically the bulk DOS of the material, in that case the NLH equation corresponds to the Thomas Fermi approximation), one computes the electric potential. The quantum solver takes the potential and recomputes the LDOS which is then given as an input to *PESCADO* . In our experience, the results are converged after 2-3 plain iterations.

Note that this is very different from schemes where the self-consistency is performed on the electronic density $n(\vec{r})$: here the electrostatic solver is *aware* of the energy dependance of the quantum system. the self-consistency is performed on $\rho(\vec{r}, E)$ which makes the convergence of the scheme much more robust.

Prototypes of *PESCADO* were used to calculate compressible and incompressible stripes in the quantum Hall effect [Armagnat *et al.* 2019, Armagnat & Waintal 2020]. The current version of *PESCADO* has been used to calculate the quantum hall edge state position in graphene pn junctions [Flór *et al.* 2022] and in machine learning reconstruction of scanning gate microscopy experiments [Percebois *et al.* 2023]. Also related are simulations of a large data set of quantum point contact experimental data [Chatzikyriakou *et al.* 2022].

This chapter is organized as follows. First, we provide a few examples of how *PESCADO* works in practice. We start with a simple simulation where we solve the Poisson equation in 2D. This first example showcases the simplicity of the API. Then we move on to more complex situations solving Helmholtz, NLH problems and end with a quantum calculation (conductance in a quantum point contact). After that, we describe the different features of *PESCADO* one after the other (geometrical engine, mesher, electrostatic solver, NLH solver).

4.1 Examples of *PESCADO* usage

This section contains examples of increasing complexity illustrating the usage of *PESCADO* .

4.1.1 Classical electrostatics of quantum nanoelectronic devices

Our first example is a split wire, a geometry inspired from electronic flying quantum bit experiments [Bäuerle *et al.* 2018, Bautze *et al.* 2014]. In the upper panel of Fig. 4.1 is a sketch of a side view along the $x - z$ plane of the device (the system is infinite along y). A two-dimensional electron gaz (2DEG, in red) is formed at the interface between a layer of GaAlAs (beige) and a layer of GaAs (not shown). The green region is implanted with some dopant density ($n_{\mathrm{d}} = 10^{23} m^{-3}$). Three metallic gates are deposited on the top of the device (in blue). Applying a negative voltage on these gates depletes the 2DEG underneath them and creates the quantum wire. With the central gate, one can further split the wire in two and/or allow tunneling between the two subwires. In this first example we make a very crude approximation and model the 2DEG with an equipotential $U(\vec{r}) = 0$ purely at the electrostatic level (Dirichlet boundary condition). This corresponds to an infinitely large LDOS in the 2DEG. This approximation should only be used when the electronic density is positive as it cannot handle the depletion of the 2DEG. The metallic gates are also treated with a Dirichlet boundary condition $U(\vec{r}) = V_g$ with a unique gate voltage V_g applied to all of them.

```
1 from pescado.mesher import patterns, shapes
2 from pescado.poisson import ProblemBuilder
```

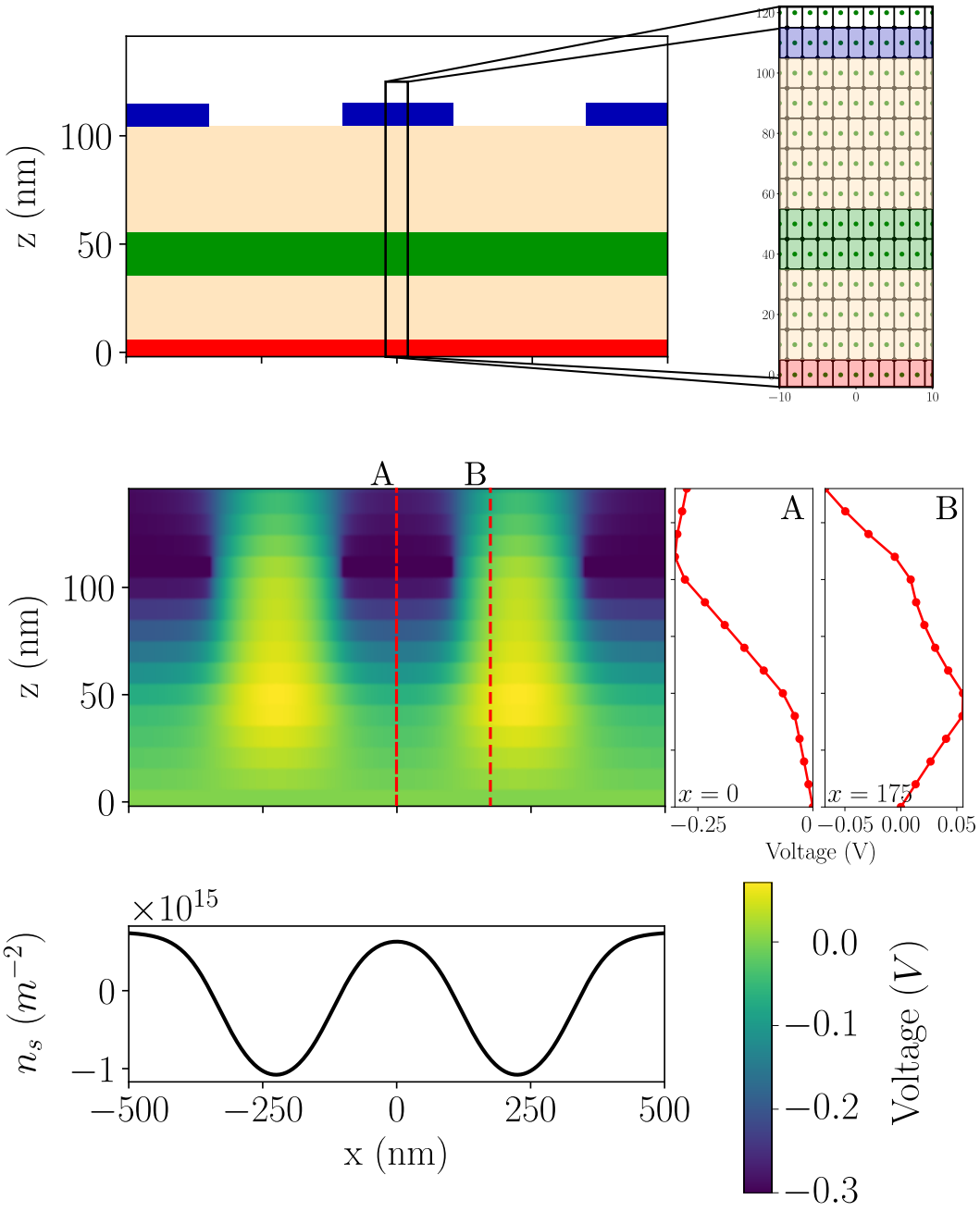


Figure 4.1: Electrostatics of a split wire. Upper left: side view of the device simulated in Script 4.1 and Script 4.2. The different regions are the 2DEG (red), the doped dielectric (green), the undoped dielectric (beige), the metallic gates (blue) and the air (white). Upper right: Voronoi cells in the central part after meshing. Middle left: calculated potential in the (x, y) plane. Middle right: potential versus z along the two cuts A ($x = 0$) and B ($x = 175nm$) indicated on the left. Lower panel: density of charge $-en(\vec{r})$ in the 2DEG.

```
3
4  # Define the device regions
5  dielectric = shapes.Box(lower_left=[-1e3, -55], size=[2e3, 160])
6  dopants = shapes.Box(lower_left=[-1e3, 35], size=[2e3, 20])
7  twodeg = shapes.Box(lower_left=[-1e3, -4.9], size=[2e3, 9.8])
8
9  gate = (shapes.Box(lower_left=[-1e3, 105], size=[650, 10])
10         | shapes.Box(lower_left=[-100, 105], size=[200, 10])
11         | shapes.Box(lower_left=[350, 105], size=[650, 10]))
12
13 # Define the simulation region
14 system = shapes.Box(lower_left=[-1e3, -55], size=[2e3, 600])
15 # Define the mesh point spacing
16 rect_pattern = patterns.Rectangular.constant(element_size=(2, 10))
17
18 # Define the ProblemBuilder
19 pb = ProblemBuilder()
20
21 # Make the mesh
22 pb.initialize_mesh(simulation_region=system, pattern=rect_pattern)
23
24 # Define material properti
25 pb.set_metal(region=gate, setup_name='gate')
26 pb.set_relative_permittivity(val=12, region=dielectric)
27
28 # Define local boundary conditions for the dielectric and tag it
29 pb.set_dirichlet(region=twodeg, setup_name='2deg')
30 pb.set_neumann(region=dopants, setup_name='dopants')
31
32 # Discretize the problem
33 pd = pb.finalized()
34
35 # Save the discrete problem
36 pd.save('Problem_phys_example_2D')
```

Script 4.1: This code defines the electrostatic problem for the system shown in Fig.4.1.

The construction of this system is shown in Script 4.1. The geometry is constructed in Lines 4-14. Since all the regions are rectangular, we can use the Box shape predefined in *PESCADO* , but we shall see that handling more complex shapes is not much more difficult. Notice the usage of the or operator | on Lines 9-11 to combine the three rectangular shapes into a larger one containing the three gates. Lines 16-22 initialize the Poisson problem and perform the initial discretization (and unique in this simple example) with a regular grid. Lines 25-30 set the boundary conditions at each region and the dielectric permittivity value (here $\varepsilon = 12\varepsilon_0$). Line 33 finalizes the problem, i.e. precomputes the matrix elements needed for actual simulations. Line 36 saves the finalized problem into a file for future use.

```
1
2  pd = Problem.load('Problem_phys_example_2D')
3
4  # Set the voltage and charge values
5  charge_density_in = pd.sparse_vector(val=1e-4, name='dopants')
6
7  voltage_in = pd.sparse_vector(val=-.3, name='gate')
8  voltage_in.extend(pd.sparse_vector(val=0, name='2deg'))
9
10 # Solve
```

```
11 voltage_res, charge_res = pd.solve(
12   voltage=voltage_in, charge_density=charge_density_in)
13
14 # Transform into charge density -nm^{-3}
15 charge_res[charge_res.indices] = (
16     charge_res[charge_res.indices] / pd.volume[charge_res.indices])
17
18 # 2DEG density
19 charge_res[pd.points(name='2deg')] = (
20   charge_res[pd.points(name='2deg')] * 10 * 1e18)
```

Script 4.2: Solves the electrostatic problem (poisson equation) for the sketch at the top of Fig.4.1. It calculates the potential profile and charge shown in the middle and bottom of Fig.4.1.

We solve the electrostatic problem in Script 4.2. We start by by loading the problem we have constructed and saved in Script 4.1. Saving/loading the problem to/from file is not necessary but is a useful feature when the same electrostatic problem must be solved several times on *e.g.* different processors. To solve a problem, one starts by defining the values of the input (see Lines 5-8), here $V_g = -0.3V$ (Line 7), $U = 0$ at the 2DEG (Line 8) and the dopant density $n_d = 10^{-4}\text{nm}^{-3}$ (Line 5, note that all lengths are in nm in *PESCADO*). In this simple example we have used constant values for the inputs, but it is easy to replace that with spatially varying values. For instance, one can use functions on the right hand side of the val=? assignment. Line 11 calls the actual solver. How to change from *PESCADO* to more conventional units, c.f. Lines 15-20, will be explained later. The results are plotted in the middle and lower panels of Fig. 4.1. In the lower panel, we observe that the electron density actually changes sign. Such a behavior could be possible in *e.g.* graphene but not in a 2DEG. This is an indication that one should go beyond the crude approximation of treating the 2DEG as an equipotential. We shall come back to this problem in Sec.4.5.2.

4.1.2 Constriction in a 2DEG at the Thomas Fermi level

```
1
2 import numpy as np
3
4 from pescado.mesher import shapes, patterns
5 from pescado.poisson import ProblemBuilder, Problem
6
7 ############ A) Create the PoissonProblem
8
9 ##### A.0) Define the device regions
10
11 size = np.array([1100, 1400, 1450])
12
13 sim_region = shapes.Box(lower_left=size * -.5, size=size)
14
15 device_region = shapes.Box(
16     lower_left=(size[0] * -.5, size[1] * -.5, -50),
17     size=(size[0], size[1], 200))
18
19 dielectric = shapes.Box(
20     lower_left=(size[0] * -.5, size[1] * -.5, -725),
```

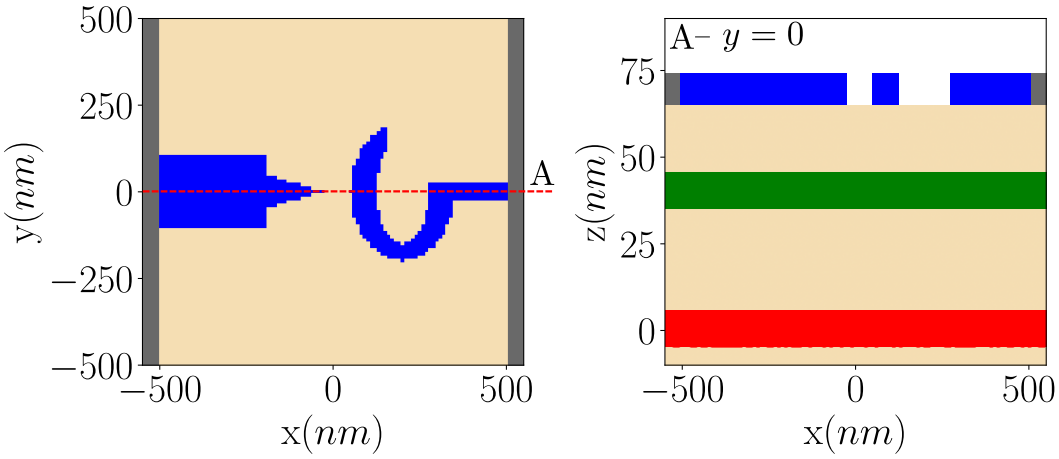


Figure 4.2: Schematics of the device defined in Script 4.3. On the left top view of the device. On the right cut along the x-z plane for $y = 0$. In blue are the constriction gates, on gray the side gates. In salmon the dielectric layer, in green the doped dielectric and white air. In red is the 2DEG.

```
21      size=(size[0], size[1], 790))
22
23  doped_dielectric = shapes.Box(
24      lower_left=(size[0] * -.5, size[1] * -.5, 35),
25      size=(size[0], size[1], 10))
26
27  gas = shapes.Box(
28      lower_left=(size[0] * -.5, size[1] * -.5, -5),
29      size=(size[0], size[1], 10))
30
31  side_gate_left = shapes.Box(
32      lower_left=(-551, size[1] * -.5, 65), size=(50, size[1], 10))
33
34  side_gate_right = shapes.Box(
35      lower_left=(501, size[1] * -.5, 65), size=(50, size[1], 10))
36
37  gate_left = shapes.extrude(
38      (shapes.Delaunay(
39          coordinates=np.array([[-200, 50], [-200, -50], [-30, 0]]))
40       | shapes.Box(lower_left=(-500, -100), size=(300, 200))),
41      axis=2, bounds=np.array([[65], [75]]))
42
43  gate_right = shapes.extrude(
44      ((shapes.Ellipsoid(center=(200, 0), radius=(150, 200))
45        - shapes.Ellipsoid(center=(200, 0), radius=(75, 150)))
46       - shapes.Box(lower_left=(153, 25), size=(1000, 400))
47       | shapes.Box(lower_left=(325, -25), size=(175.9, 50))),
48      axis=2, bounds=np.array([[65], [75]]))
49
50  ##### A.1) Make the mesh
51
52  pb = ProblemBuilder()
53
54  step_coarse = 50
55  coarse = patterns.Rectangular.constant(
56    element_size=(step_coarse,) * 3)
```

```
pb.initialize_mesh(simulation_region=sim_region, pattern=coarse)

step_fine = 10
fine = patterns.Rectangular.constant(
  element_size=(step_fine, ) * 3)
pb.refine_mesh(region=device_region, pattern=fine)

pb.mesh.save('Mesh_QPC_3D_DISC')

##### A.2) Define the device materials

# Dielectric
pb.set_relative_permittivity(val=12, region=dielectric)

# Metal
pb.set_metal(region=(gate_right | gate_left),
             setup_name='gate')
pb.set_metal(region=(side_gate_left | side_gate_right),
             setup_name='side_gate')

##### A.2) Define the Boundary Conditions
pb.set_flexible(region=gas, setup_name='gas')

pb.set_neumann(region=doped_dielectric, setup_name='dopant')

pd = pb.finalized()

pd.save('Problem_QPC_3D_DISC')
```

Script 4.3: define the electrostatic problem associated with the constriction shown in Fig.4.2

In our second example, we will increase the complexity of the problem in two ways. First, we will treat the 2DEG properly, at the Thomas Fermi level. Instead of using Dirichlet boundary condition in the 2DEG as in the previous example, we will use our NLH solver with $\rho(\vec{r}, E) = \theta(E)\rho_0$. Here $\theta(E)$ is the Heaviside function and $\rho_0 = m_e^*/(\pi\hbar^2)$ the bulk density of states of a 2DEG, with $m_e^* = 0.067m_e$ the effective electron mass. The crucial aspect here is not that ρ_0 is finite (as opposed to infinite when the 2DEG is treated at the Dirichlet level) but that $\rho(\vec{r}, E < 0) = 0$, *i.e.* the 2DEG can actually get depleted; and depleted regions do not screen the electric field anymore.

Second, we will consider a three dimensional geometry. The device layout is shown in Fig.4.2 : on the left is a top view showing the two metallic gates (in blue) that form a constriction with a pointed shape for the left gate and a question mark shape for the right gate. The gray region also correspond to (side) gates, they prevent an abrupt drop of density on the edges.On the right of Fig.4.2 is a side view that shows the different layers (2DEG in red, dopants in green, metallic gates in blue on top). Note that this figure is easy to generate using *PESCADO* , as we shall see later in Section 4.4.5. The device of Fig.4.2 is a sort of QPC showing quantized conductance plateaus as we deplete the electrons underneath the gate and force the current to flow in the constriction region.

Script 4.3 defines the geometry of the problem. There we use more advanced **Shapes** to describe the various regions: in addition to the **shapes.Box** (a rect-

angular box), we make use of **shapes.Delauney** (take a list of coordinates and construct a convex polyhedron from it), **shapes.Ellipsoid** (an ellipsoid shape), **shapes.extrude** (extrudes a 3D shape from a 2D one) as well as arithmetic operations between **Shapes** (minus sign (−) to remove some parts, or (|) to glue two parts together).

The problem needs to be discretized with a mesh size a smaller than the characteristic length scales of the problem. As characteristic length scales we have those given by the geometry (typically $\approx 70nm$, the gate-2DEG distance) and the Fermi wave length $\lambda_F = \sqrt{2\pi/n_s} \approx 88nm$. We use a coarse discretization with rectangular elements of size $50 \times 50 \times 50nm^3$ except at the active device region, where we use a fine mesh of size $10 \times 10 \times 10nm^3$. Using two different meshes instead of just one fine mesh provides a reduction of the number of cells from roughly 2. 10^6 to 4. 10^5, which provides a significant gain in simulation time. The rest of the script assigns the cell within each region to a specific type. A metal type cell is one with Dirichlet boundary condition *and* an infinite dielectric constant. A Neumann type cell is the default, it models a dielectric with some possible charge density. A Flexible type cell is used for the active 2DEG region for the Helmholtz solver, see Section 4.4.6.

```
1
2  import copy
3  import pickle
4  import numpy as np
5  from scipy import constants
6
7  from pescado.poisson import Problem
8  from pescado.tools import SparseVector, meshing
9  from pescado.self_consistent import problem as sp_problem
10
11 ############ B) Load the Problem and define the Discrete ILDOS
12
13 #### B.1) Load the Problem
14
15 pd = Problem.load('Problem_QPC_3D_DISC')
16
17 #### B.2) Define the ILDOS
18
19 def disc_ildos(surf):
20
21     dos = (1 * (0.067 * constants.m_e)
22           / (np.pi * constants.hbar ** 2))
23     dos *= 1e-18 * surf
24     dos *= constants.elementary_charge
25
26     coord = np.empty((3, 2), dtype=float)
27     coord[:, 0] = np.array([-1, 0, 1])
28     coord[:, 1] = np.array([0, 0, dos])
29
30     return coord
31
32 ildos = [disc_ildos(surf=10 * 10),]
33
34 gas_idx = pd.points(name='gas')
35
36 site_ildos = SparseVector(
37     values=np.zeros(len(gas_idx)), indices=gas_idx, dtype=int)
38
```

```
diff_vol_idx = gas_idx[
  (np.abs(pd.volume[gas_idx] / 10 - 10**2) > 1e-10)]

for idx in diff_vol_idx:
    ildos.append(disc_ildos(surf=pd.volume[idx] / 10))
    sites_ildos[idx] = len(ildos) - 1

############ C) Define the SelfConsistent Problem

###### C.0) Initialize an instance of ScrodingerPoisson

sp_problem = solver.thomas_fermi(
        ildos=ildos, sites_ildos=site_ildos, poisson_problem=pd)

###### C.1) Solve for a given gate voltage / dopant configuration

u_d = pd.sparse_vector(val=-.274, name='gate')
u_d.extend(pd.sparse_vector(val=-1., name='side_gate'))

q_n = pd.sparse_vector(val=4 * 1e-4, name='dopant')

poisson_input = {'voltage': u_d, 'charge_density': q_n}

initial_guess = SparseVector(
    values=np.zeros(len(gas_idx)), indices=gas_idx, dtype=float)

sp_problem.solve(
    poisson_input=poisson_input,
    initial_guess=initial_guess)

qcharge_btf = sp_problem.quantum_charge(iteration=-1)
chem_pot_btf = sp_problem.chemical_potential(iteration=-1)
```

Script 4.4: How to solve the thomas fermi problem for the 3D *PESCADO* problem of Fig. 4.2 and the 2DEG bulk ILDOS

Script 4.4 performs the self-consistent calculation. We need to provide *PESCADO* with the ILDOS inside the 2DEG. The ILDOS is defined as $n(\vec{r},\mu) \equiv \int_{-\infty}^{\mu} \rho(\vec{r},E)dE$ which becomes, for our simple bulk DOS, $n(\vec{r},\mu) = \rho_0\mu\theta(\mu)$. Since this ILDOS is piecewise linear, we will use our solver dedicated to this particular case. It takes as an input a list of values of μ (*ildos[:,0]*) and a corresponding list of values of $n(\mu)$ (*ildos[:,1]*).

Fig.4.3 shows the calculated potential (left) and electronic density (right) inside the 2DEG. At this gate voltage ($V_g = -0.274V$), the 2DEG is depleted beneath the gate so that the current only flows through the constriction.

4.1.3 Conductance calculation - going beyond Thomas Fermi

To proceed, we need to make some quantum calculations. We do so using the Kwant package [Groth *et al.* 2014]. We describe the 2DEG with a simple effective mass Hamiltonian given by

$$H = \frac{p_x^2 + p_y^2}{2m} - eU(x,y) \tag{4.4}$$

which we discretize on the same grid as the electrostatic problem. We only treat the part referred to as the quantum active region, zone delimited by the red square

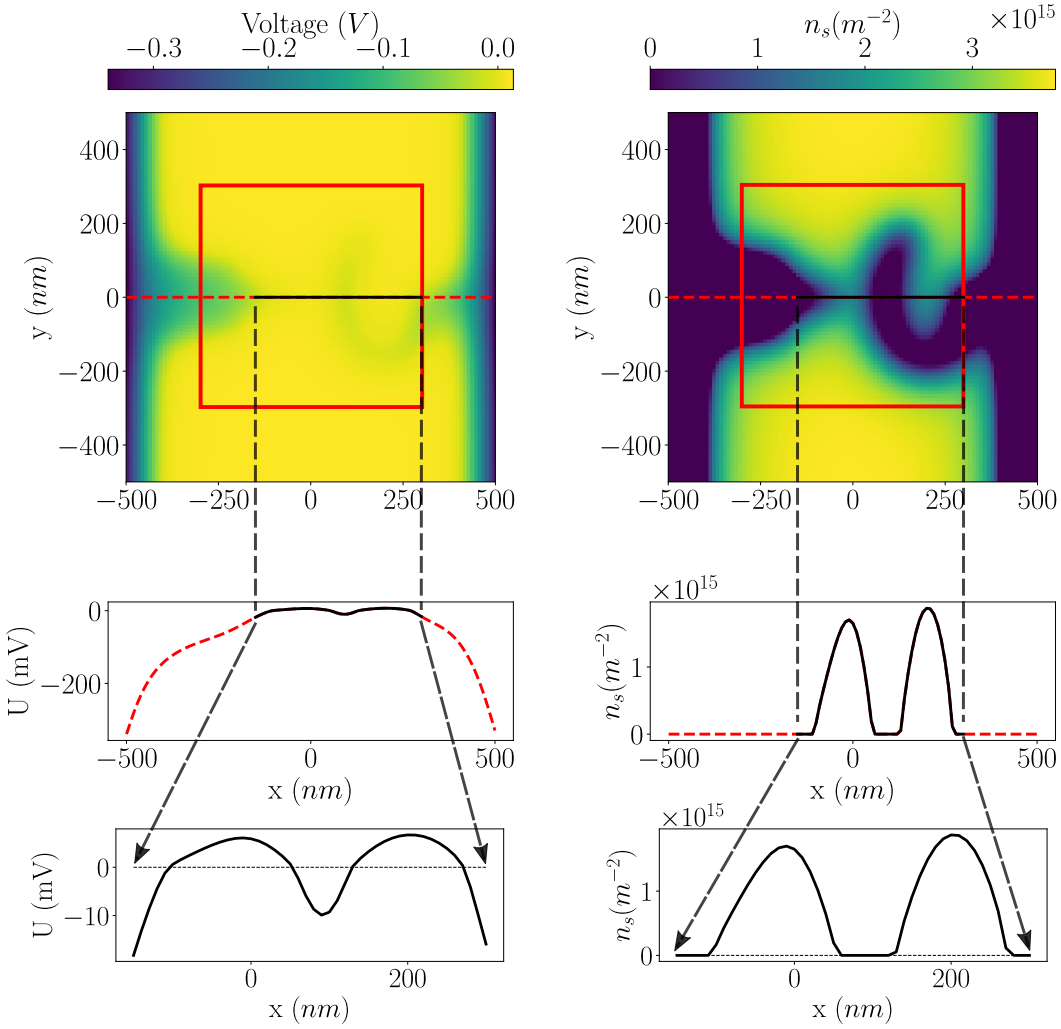


Figure 4.3: Result of the Thomas Fermi calculation for the device of Fig.4.2. Left panel: electric potential, right panel: electronic density. Top panels: cut in the x-y plane at the 2DEG region ($z = 0nm$). Middle panels, one-dimensional cut along the black line shown in the top panels. Lower panels: zoom inside the quantum active region (square region delimited by the red line).

in in Fig.4.3. The scripts for the quantum calculation are given in supplementary material. From the calculation of $U(x, y)$ performed in Section 4.1.2, we can already compute the conductance $G(V_g)$ versus gate voltage V_g. The results are shown in Fig. 4.4. We recover the expected conductance plateaus which are gradually smeared out as one opens the constriction (an experimental curve would look very similar).

Kwant also allow us to recalculate the ILDOS from a given potential profile $U(x, y)$. There are various methods to do so, here we use the Kernel polynomial method (KPM, see Appendix B and **kwant.kpm**) which has a good balance between speed and accuracy. Then the self-consistent potential is recalculated from the new ILDOS. We observe that the potential and density are only slightly affected by this step. The conductance $G(V_g)$, which is more sensitive, is essentially shifted by around $50mV$ but its shape is barely affected. One quantum update is sufficient to obtain a (visually) converged result. There is very little noticeable difference between the conductance obtained after one quantum iteration (black) and three (green), c.f. Fig. 4.4.

This concludes the introductory part of this chapter. We shall now go back from the very beginning and slowly go through the different concepts of *PESCADO* . As we do so, we shall introduce the API.

4.2 Describing geometries within *PESCADO* : a lightweight geometrical engine

The first feature an electrostatic solver must provide — actually that any partial differential solver must provide — is a way to describe the volumes, surfaces, lines and points that define the geometry of the problem one wants to solve. This includes describing the electrostatic gates, regions with different dielectric constants or dopant concentration, quantum region of interest...This is the role of the “ geometrical engine ”.

PESCADO geometrical engine is based entirely in the description of the volumes. The surfaces are defined implicitly through the boundaries between the different volumes (and lines are the intersections of different surfaces). This is in contrast to other approaches where one defines the low dimensional objects first. These latter approaches are suitable when the solution of the problem is very sensitive to the precise nature of the low dimensional objects. For instance, in magnetism it is important that a disk object is precisely described as a disk with a circular boundary otherwise the numerical solution might show some spurious anisotropy. The electrostatic problem is however not very sensitive to these details. The description by volumes used in *PESCADO* has the advantage of being very versatile as well as easy to use and parametrize. Note that *PESCADO* simulations work in one, two and three dimensions but we use the vocabulary of 3D objects everywhere. In d dimensions, we use the world "volume" for d dimensional object, "surface" is a $d-1$ dimensional object, and "line" for a $d-2$ object. Hence, an object like e.g. a disk in 2D will be called a volume.

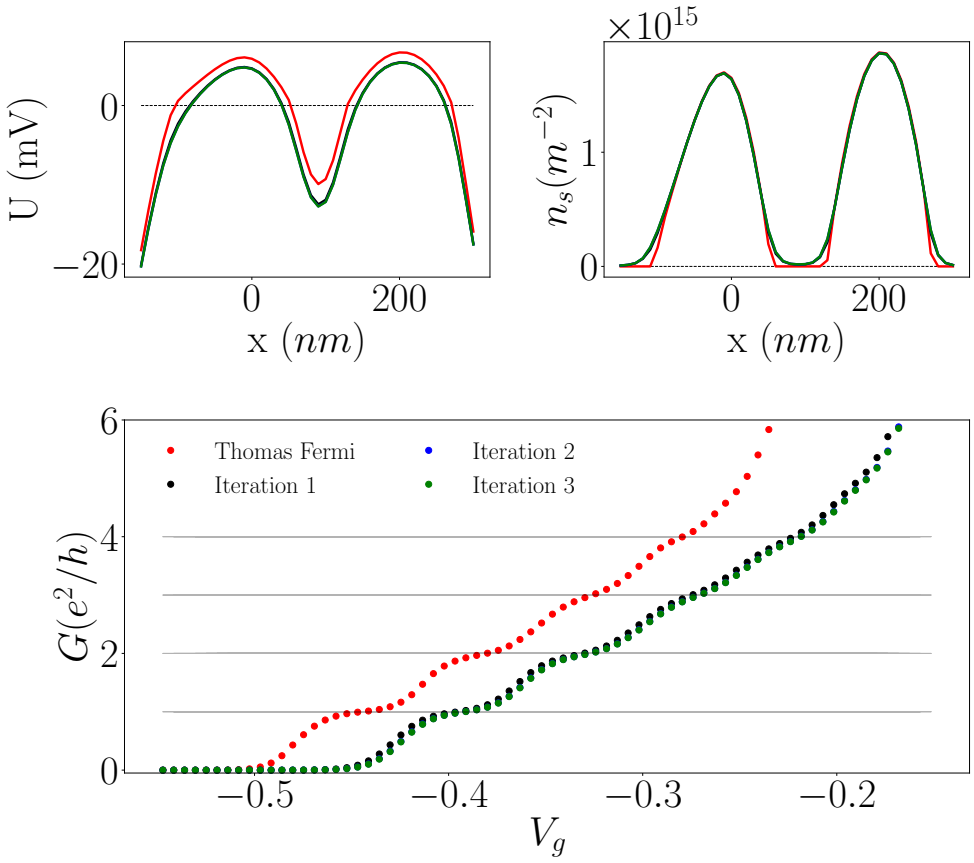


Figure 4.4: Upper panels: electric potential (left) and electronic density (right) in the 2DEG at $y = z = 0nm$ (same cut as in Fig.4.3) for $V_g = -0.274V$. bottom panel: conductance G versus gate voltage V_g showing the quantized conductance plateaus. The curves in red correspond to a potential $U(x, y)$ calculated at the Thomas-Fermi approximation while the curves in black correspond to a single quantum update (see text). Those in blue and green correspond to, respectively, two and three quantum updates.

4.2.1 The concept of "shape"

We define a volume, hereafter referred to as a "region" simply by a function $f(\vec{r})$ such that $f(\vec{r}) = 1$ when $\vec{r}$ is inside the region and $f(\vec{r}) = 0$ when the point $\vec{r}$ is outside. The user simply needs to define various Python functions and associate each of them to a region and that's it, one has defined the geometry of the problem. For instance, to describe a spherical electrostatic gate centered in $\vec{r}_0$ and of radius R, one would define the function $f(\vec{r}) = \theta(R - |\vec{r} - \vec{r}_0|)$ where $\theta(x)$ is the Heaviside step function.

In practice this concept is implemented using the Python class **Shape**. A " shape" (an instance of the class Shape) is a thin wrapper around the above mentioned function. For implementation reasons, a shape also needs to know, at least vaguely, where the volume is. This becomes handy later when one tries to mesh the system to know where to generate points. Hence, a shape also contains a " bounding box" (a rectangle in 2D, a rectangular parallelepiped in 3D) that encapsulates the region of interest. Strictly speaking, the function $f(\vec{r})$ is the intersection of the function provided by the user and the bounding box.

The **shapes** submodule contains many tools to create, manipulate and combine shapes. One can simply create a shape by providing an arbitrary function $f(\vec{r})$ and the bounding box (defined by the lower left corner and upper right corner of the rectangular parallelepiped). Common geometries such as rectangles, triangles, or circles can be created with their own methods for convenience. A given shape can also be rotated, translated or dilated. Last, several shapes can be combined to form a new shape: the logical *and* operator provides the intersection of two shapes, the logical *or* provides the union and one can also remove a shape from another. Using these tools, it is usually a matter of very few lines of code to define a given region.

4.2.2 Examples of defining shapes

Let us illustrate the above concepts by defining a few shapes within *PESCADO* . The different regions that we will construct are shown in Fig.4.5 while the corresponding script is shown in Script.4.5.

Example I creates a shape by providing explicitly the function $f(\vec{r})$, see Script.4.5 lines 6 to 14 and the resulting shape in Fig.4.5 panel I. This shape corresponds to a rectangle from which we "subtract" two disks. The bounding box is an array with the format $[[x_{\min}, y_{\min}], [x_{\max}, y_{\max}]]$. Note that the function $f(\vec{r})$ must be vectorized for efficiency reasons: its input is a 2-dimensional (numpy) array whose different columns correspond to the different coordinates x,y and z (the number of coordinates fix the dimension of the problem) and the different rows correspond to different point $\vec{r}_1, \vec{r}_2, \vec{r}_3$.... We refer to such a 2-dimensional (numpy) array as a "set of points". The Python function *f(r)* takes a set of points and return a one-dimensional numpy array of booleans that indicate which point is in (1) or out (0) of the corresponding region.

```
1 import numpy as np
2 from pescado.mesher import shapes
```

```
3
4  # Example I: providing f(r) explicitly
5  def f(r):
6      x, y = r[:, 0], r[:, 1]
7      circ_1 = (x ** 2 + (y + 5.5) ** 2) >= 5 ** 2
8      circ_2 = (x ** 2 + (y -5.5) ** 2) >= 5 ** 2
9      rect = (np.abs(x) <= 5 / 2) * (np.abs(y) <= 10 / 2)
10     return circ_1 * circ_2 * rect
11
12 bounding_box = np.array([[-5, -5], [5, 5]])
13 region = shapes.General(func=f, bbox=bounding_box)
14
15 # Example II: a rectangle
16 rectangle = shapes.Box(lower_left=[-14, -4], size=[12, 8])
17
18 # Example III: an ellipse
19 ellipse = shapes.Ellipsoid(center=[0, 0], radius=(4.2, 2.75))
20
21 # Example IV: a convex polyhedron
22 triangle = shapes.Delaunay(coordinates=np.array(
23     [[-8, -5], [-2, -5], [-5, 5]]))
24
25 # Example V: rotation of a shape
26 rotated_triangle = shapes.rotate(
27     shape=triangle, angle=3 * np.pi / 2,
28     axis=(0, 0, 1), origin=[-5, 0])
29
30 # Example VI: dilatation of a shape
31 dilated_rectangle = shapes.dilatate(
32     rectangle, dilatation=(0.4, 0.2), center=[-8, 0])
33
34 # Example VII: substraction of one shape from another
35 lead_l = rotated_triangle - dilated_rectangle
36
37 # Example VIII: reflection around the y-axis
38 lead_r = shapes.reflect(
39     shape=lead_l,
40     origin=np.array([0, 0]), normal=np.array([1, 0]))
41
42 # Example IX: combining various shapes
43 operations = (rectangle | lead_l | lead_r) - sphere
```

Script 4.5: Tutorial ilustrating how to define the 2D regions of Fig.4.5 using instances of **Shape**. Lines 7-30 show how to define standard shapes (regions I-IV). Lines 34-56 ilustrate some of the operations that can be performed to and in between instances of **Shape** (regions V-IX).

Examples II, III and IV use convenient methods to build respectively a rectangle, an ellipse and a (arbitrary complex) convex polyhedron (here a triangle). For the convex polyhedron, the input of the method is a list of the points $\vec{r}_1, \vec{r}_2$... that form the vertices of the polyhedron. This method is quite versatile to build relatively complex shapes.

Examples V, VI and VIII showcase methods that transform a shape into another using respectively rotations, dilatations and reflection with respect to an axis. Shapes can also be translated and extruded (not shown).

Examples VII and IX show how several shapes may be combined to form a new shape. The "and" operator $a\&b$ performs the intersection between two shapes (not

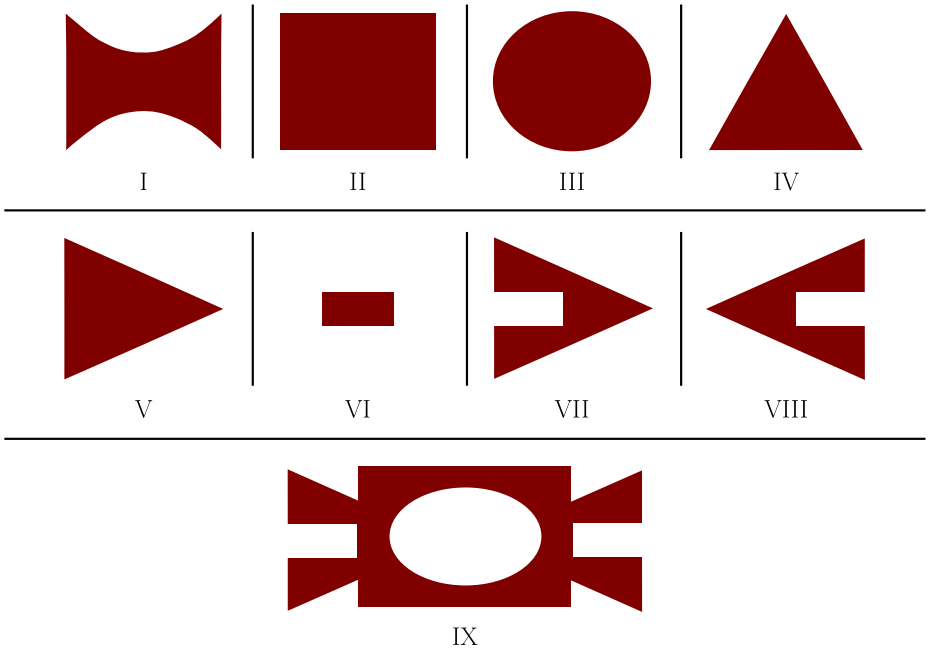


Figure 4.5: Regions defined in the Script.4.5 above. Regions I-IV are obtained using standard shapes. Regions V to VIII are obtained by applying geometrical and logical operations to regions II-IV. Region IX is obtained by applying logical and arithmetic operations between regions II, III, VII and VIII.

shown), the "or" operator $a|b$ performs the union (see IX), the subtraction $a - b$ remove the second shape b from the first a (see VII and IX and the "xor" operator is defined as $a \wedge b = a|b - a\&b$.

Using a combination of all the methods shown above, one can quickly build any desired shape. The examples above were given in two dimensions. The same methods also work in one and three dimensions, one just need to adapt the number of components to the value of the dimension, see the examples X-XIII in Script.4.6.

To use the function $f(\vec{r})$ of a shape, one simply calls it with a "set of points" as argument (in the form of a two dimensional numpy array as described in example I) and the shape returns a one dimensional numpy array of zeros and ones. In example XIV, the call to *pyramid(pts)* returns an array indicating which of the points in *pts* belong to the pyramid. Indexing *pts* with this array (*pts[pyramid(pts)]*) returns the set of points that belong to the pyramid.

```
1
2  import numpy as np
3  from pescado.mesher import shapes
4
5  # Example X: a one dimensional box (a segment)
6  pescado.mesher.shapes.Box(lower_left=[0, ], size=[5, ])
7
8  # Example XI: a three dimensional box
9  shapes.Box(lower_left=[-14, -4, -2], size=[12, 8, 5])
10
11 # Example XII: a 3D three dimensional ellipse
12 shapes.Ellipsoid(center=[0, 0, 0], radius=(4.2, 2.75, 3))
13
14 # Example XIII: a 3D pyramid
15 pyramid = shapes.Delaunay(coordinates=np.array(
16     [[-8, -5, 0], [8, -5, 0], [-8, 5, 0], [8, 5, 0],
17      [0, 0, 10]]))
18
19
20 # Example XIV: how one uses the f(r) function of a shape
21 pts = np.array([[-8, -5, 2], [-4, -3, 1], [-5, 5, 0],
22                 [0, 0, 3], [0, 0, -2]])
23
24 pts_inside_pyramid = pts[pyramid(pts)]
```

Script 4.6: Line 7 show how to create a 1D box. Lines 5-13 shows three examples on how to create 3D regions using instances of **Shape**. Lines 18 to 21 shows how to use an instance of **Shape** to obtain the coordinates inside the region it describes.

4.3 A lightweight finite volume mesher.

Once the continuum geometry has been defined, the second feature a electrostatic solver must provide is *some* way to discretize the problem. *PESCADO* implements a finite volume discretization scheme, i.e. the volume of the simulation is discretized into a set of small cells. Finite volume is particularly well suited for the electrostatic problem since it allows one to conserve the charge exactly irrespectively of the coarseness of the discretization (the main source of errors in an electrostatic simulation). In fact, for any discretization, however coarse, the resulting discretized

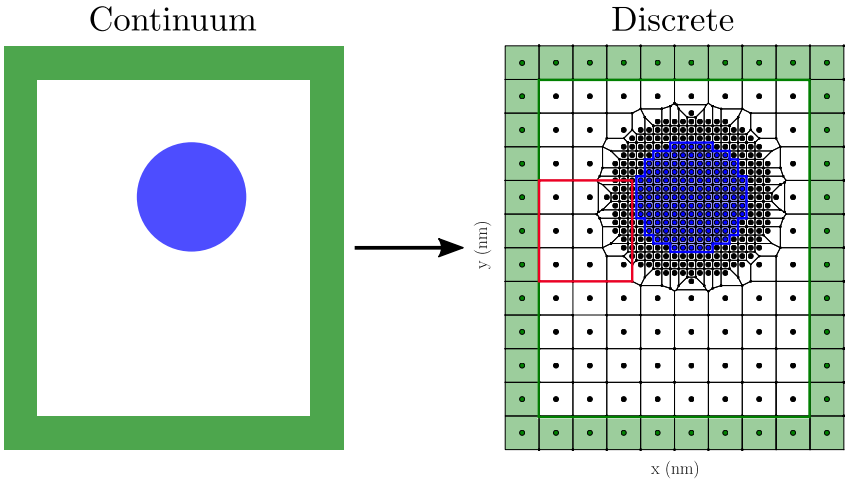


Figure 4.6: Example of discretization with finite volumes. Left: a 2D problem in the continuum with two regions of interest (green and blue). Right: Voronoi diagram obtained after discretization. The black circles indicate the mesh points. The Voronoi cells are separated by thin black lines. This mesh is generated by Script 4.7. The red line indicates a region for which a zoom is shown in Fig.4.7

problem is a valid discrete electrostatic problem defined in terms of a lamp system of capacitors. Hence the results are always guaranteed to be physical (although to be precise, one must also use a fine enough grid).

The small cells used in *PESCADO* are "Voronoi cells". One discretizes the geometry by defining a set of mesh points $\vec{r}_i$ which are the centers of the Voronoi cells. A voronoi cell $\mathcal{C}_i$ centered around a given point $\vec{r}_i$ consists of all the points $\vec{r}$ that are closer to $\vec{r}_i$ than to any other mesh point: $\vec{r} \in \mathcal{C}_i$ if and only if $\forall j \neq i |\vec{r}-\vec{r}_j| > |\vec{r}-\vec{r}_i|$. In 2D, a Voronoi cell $\mathcal{C}_i$ is a polyhedron separated from its neighbors $\mathcal{C}_j$ by the perpendicular bisector of $(\vec{r}_i, \vec{r}_j)$; it trivially extends to 3D. There exists standard algorithms that can construct a set of Voronoi cells from the list of mesh points $\vec{r}_i$ (as implemented in e.g. the Qhull library). However this construction can be memory/-computationally intensive for large systems. The approach taken in *PESCADO* is to precalculate the Voronoi cells for some regular set of points (these will be referred as a "pattern", see below) and only use the explicit Voronoi construction to combine several patterns in order to e.g. refine the grid in a region of particular interest. This strategy allows one to construct a wide variety of regular meshes (rectangular, spherical etc ...) and meshes with a wide variety of voronoi cells (within the same mesh, different discretization steps and voronoi cell shapes) at a relatively low computational cost. An example of Voronoi diagram used in *PESCADO* is shown in Fig. 4.6. On the left of Fig. 4.6 is the continuum geometry of the problem, a rectangular box with two colored regions: a blue circle and a green rectangular "belt". For instance, each region could be e.g. doped dielectric, a 2DEG where we would solve the quantum problem or metallic regions described by an equipotential. On the right of Fig. 4.6 is the Voronoi diagram after discretization. The mesh points

are represented as black circles. Each mesh point is surrounded by its Voronoi cell; different Voronoi cells are separated by thin black lines. The Voronoi cells are simple squares when the mesh points are themselves on a square lattice, e.g. close to the green region or inside the blue region. At the intersection between two regular regions, the Voronoi cells are more complex. Fig. 4.6 shows a typical example where a relatively coarse grid is sufficient in the bulk of the system but one requires a finer resolution in some region of interest (here the blue region).

4.3.1 The concepts of pattern and mesh

The *PESCADO* mesher uses two concepts, implemented in two different python classes: **Mesh** and **Pattern**.

An instance of **Mesh** represents the current discretization of a system. It is defined by the set of coordinates of the mesh points $\vec{r}_i$. Internally, a mesh also keeps track of the Voronoi diagrams associated to the mesh points. In particular, it can provide the list of neighbors of a given point and the corresponding distances or surfaces.

A pattern (an instance of **Pattern**) is the primary way to define a mesh in *PESCADO* . A pattern is a precalculated Voronoi diagram in a common situation, i.e. for a common voronoi cell shape or grid, e.g. squared. The simplest (and most most common) pattern used in *PESCADO* is the rectangular pattern (**patterns.Rectangular**) where the $\vec{r}_i = (x_i, y_i, z_i)$ are positioned on a regular rectangular grid, $x_i = na_x$, $y_i = ma_y$ and $z_i = pa_z$ where (n, m, p) are integers and (a_x, a_y, a_z) the discretization step along the three axis. For such pattern, the Voronoi cells are simple rectangles. The main advantage provided by a pattern is the ability to construct a mesh from the said pattern and a shape: the set of mesh points is just given by the points of the pattern that belong to the shape.

PESCADO possesses a powerful tool that allows one to build meshes that are more complex than simple patterns: the ability to merge two meshes together. Last, a special kind of pattern, the finite pattern (instances of of **patterns.Finite**) builds a pattern from an arbitrary list of points $\vec{r}_i$ (using the scipy python API to the Qhull library). Finite patterns allows one to build fully general patterns, including irregular ones, but can be computationally costly and are rarely needed in practice.

The rest of this section is organized as follows. We first present, in Section 4.3.2, the minimum python script that was used to generate the Voronoi diagram shown in Fig. 4.6. Section 4.3.3 provides some background material on Voronoi Diagrams. Section 4.3.4 goes through the API for accessing the different properties of the Voronoi diagram once it has been constructed. Section 4.3.5 shows some advanced functionalities to build more complex Voronoi diagrams. Section 4.3.6 describes the algorithm used internally for merging two meshes. Finally, Section 4.3.7 contains implementation details of the **Pattern** class. In particular, it explains how a user may define new patterns.

4.3.2 A simple example of mesh construction.

```python
from pescado.mesher import shapes, patterns, Mesh

# 1. Define the coarse mesh region and spacing
coarse = patterns.Rectangular.constant(element_size=(2, 2))
box = shapes.Box(lower_left=[-20, -20], size=[40, 40])
mesh = Mesh(simulation_region=box, pattern=coarse)

# 2. Refine the mesh in the fine mesh region
fine = patterns.Rectangular.constant(element_size=(0.5, 0.5))
sphere = shapes.Ellipsoid.hypersphere(center=(0, 0), radius=5)
mesh.refine(region=sphere, pattern=fine)

# 3. Save the mesh
mesh.save('mesh_minimal_example')
```

Script 4.7: Minimal example to construct an irregular mesh using *PESCADO* .

Script 4.7 generates the mesh shown in Fig.4.6. The mesh consists of a fine spherical region inside a larger and coarser rectangular region. The construction is done in two steps. First, one defines the pattern of the coarse region (*coarse*) and the shape defining the full simulation box (*box*). Here *element_ size* gives the discretization step of the rectangular pattern, $(a_x, a_y) = (2, 2)$. The initial mesh (*mesh*) is directly obtained from *coarse* and *box*.

In the second step, one defines a finer pattern (*fine*, now $(a_x, a_y) = (0.5, 0.5)$ is four times smaller) and the associated shape where the refinement will be performed (*sphere*). The refine method of mesh performs all the important calculations: (i) it removes all the existing mesh points inside *sphere*, (ii) it adds all new mesh points from the *fine* pattern that are inside *sphere* and (iii) it updates the Voronoi diagram of *mesh*. For most applications it is sufficient to call the refine method once but it can be called as many times as needed. Note that in practice the refinement is done with the intersection of *sphere* and the *simulation_ region* (which is in fact *box*), as the latter cannot be changed once *mesh* is defined.

4.3.3 More on Voronoi Diagrams

A voronoi diagram is defined implicitly by its set of mesh points $\{\vec{r}_i\}$. A point $\vec{r}$ belongs to the Voronoi cell $\mathcal{C}_i$ of mesh point $\vec{r}_i$ if it is closer to $\vec{r}_i$ than to any other point $\vec{r}_j$ $\forall j \neq i$, i.e. verifies

$$\left[\vec{r} - \frac{\vec{r}_i + \vec{r}_j}{2}\right] \cdot (\vec{r}_i - \vec{r}_j) > 0 \tag{4.5}$$

In other words, a Voronoi cell is a Convex polyhedron separated from its neighbors by intersection of planes (defined by Eq.(4.5) with an equal sign). Fig. 4.7 shows an example of a Voronoi cell $\mathcal{C}_i$ (in gray) centered on a mesh point $\vec{r}_i$ (in green). The neighbors of $\vec{r}_i$, in the Voronoi sense, are the mesh points $\vec{r}_j$ (in red) whose Voronoi cell touch $\mathcal{C}_i$ (have a finite surface of contact with $\mathcal{C}_i$ called a ridge). The

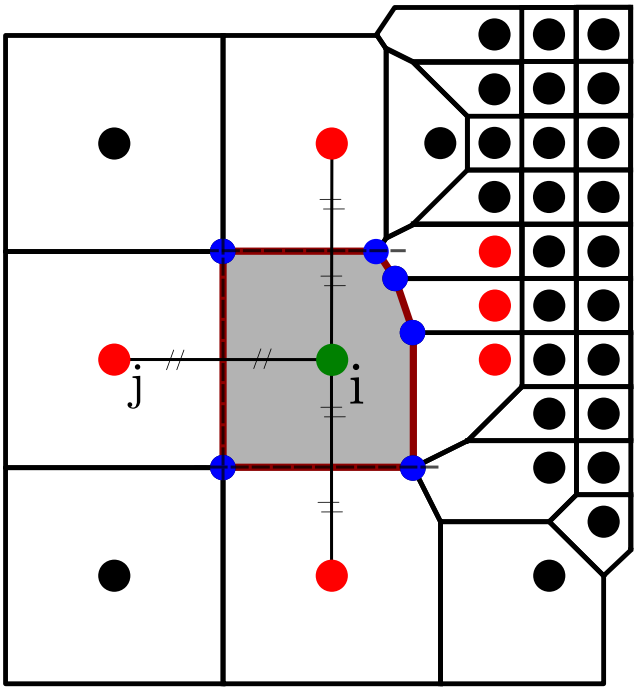


Figure 4.7: Zoom of the Voronoi diagram of Fig.4.6 (inside the rectangle delimited by the red line). The voronoi cell $\mathcal{C}_i$ of the point $\vec{r}_i$ (in green) is shown in gray. The red points are the points r_j that are the neighbors of the mesh points $\vec{r}_i$. The blue points are the vertices of $\mathcal{C}_i$ that define the ridges (dark red lines) that separate $\mathcal{C}_i$ from its neighbors $\mathcal{C}_j$.

borders of the Voronoi cell are defined by the vertices (blue points) which allow one to reconstruct its ridges (red lines in Fig.4.7). The set of all voronoi cells pave the simulation space.

Calculating a Voronoi diagram from the mesh points $\{\vec{r}_i\}$ amounts to computing (i) the list of neighbors j of each mesh point i and (ii) the Voronoi cells $\{\mathcal{C}_i\}$. Each cell is further described by (a) its set of vertices and (b) its set of ridges, each ridge being an ordered subset of the vertices. There are efficient algorithms for the construction of the voronoi diagram from an arbitrary set of points $\{\vec{r}_i\}$. *PESCADO* relies on the Qhull library for this purpose [Barber *et al.* 1996]. However, even these efficient algorithms can be computationally expensive (time wise or memory wise). Hence, the strategy of *PESCADO* is to rely on these algorithms only for small subsets of the mesh points, as we shall see in Section 4.3.6.

4.3.4 Extracting the Voronoi diagram from a mesh

Once a mesh has been constructed it can be used to recover the information needed to construct the discretized electrostatic problem, e.g. the list of neighbors of a given site, the distance to a given neighbor, the volume of the cell... This section illustrates the API for extracting such information. Each mesh point $\vec{r}_i$ is labeled by its index i, index which is also used to label the Voronoi cells. The API facilitates accessing these indices.

Let's consider the examples shown in Script 4.8. We suppose that the mesh of Fig. 4.6 has already been constructed with Script 4.7.

There are two methods for selecting a set of mesh points from a mesh, either through a shape or by giving the coordinates explicitly. Let us consider the first method. Line 10 of Script 4.8 defines the shape corresponding to the red region of Fig.4.6. The method **plot()** used in line 13 produces a plot of the selected region (the shape *zoom*). The output looks much similar to Fig.4.7 - albeit with fewer details. To obtain the list of the indices (more precisely a numpy array) of the mesh points inside the shape *zoom*, one uses the method **inside()** as in Line 18. Once this list of indices (*ele_ in_ zoom*) has been obtained, one can get: (i) the coordinates of the mesh points using **coordinates()** (a numpy array of $\vec{r}_i$ with one column per dimension, c.f. line 19); (ii) the indices of the neighbors using **neighbors()** (a list of numpy array of indices, c.f. line 22); (iii) the surfaces separating the mesh point from its neighbors using **surface()**(a list of numpy array of the surfaces, c.f. line 27); (iv) the distances to the neighbors using **distance** (a list of numpy array of distances, c.f. line 28) and (v) the volumes of the cell using **volume** (line 29). More explicitly $i = ele\_in\_zoom[k]$ is the index of the $(k+1)^{\text{th}}$ mesh point inside *zoom*, and $j = neig\_zoom[k][m]$ is the index of the $(m+1)^{\text{th}}$ neighbor of i (with no specific rule for ordering them). The surface that the Voronoi cells of indices i and j have in common is given by $surf\_zoom[k][m]$.

The second selection method is to provide explicitly the coordinates of the points one is looking for. An example is provided line 35 and 36. The method **points()** returns the indices associated to a numpy array of coordinates. Here there is a

unique point of coordinate $\vec{r}_i = (-6, -2)$, the green point of Fig.4.7.

Besides the basic properties of the Voronoi cell (surface, distances, volumes and neighbors one can also retrieve the entire Voronoi cell, described by its vertices and ridges. This is done line 41 using the **voronoi()** method, it returns an instance of **VoronoiCell** for a single mesh point index. The voronoi cell (*vor_ce*) allows one to access the basic mesh properties: the volume, surfaces and distances; but also more detailed information: the list of the vertices of the cell (line 44) and the ridges (line 45, the vertices of each ridge, so a list of list of vertices).

```
1
2  import numpy as np
3  from pescado.mesher import shapes
4
5  import matplotlib.pyplot as plt
6
7  # Accessing the properties of a mesh
8  # Method 1: the mesh points are defined by a shape
9  # Define the region of interest
10 zoom = shapes.Box(lower_left=[-9, -5], size=[5.5, 6])
11
12 # Plot a region of mesh (only for 2D meshes)
13 mesh.plot(
14   regions=[zoom, ], c='g', mew=5, ratio=0.7, lw=2)
15 plt.show()
16
17 # Select the elements inside it and get its coordinates
18 ele_in_zoom = mesh.inside(zoom)
19 coord_in_zoom = mesh.coordinates(ele_in_zoom)
20
21 # Find its neighbors
22 neig_zoom = mesh.neighbors(ele_in_zoom)
23 problem_builder_tuto
24 # Here surface() and distance() also return neig_zoom.
25 # They correspond to the mesh point index sharing
26 # the surface / distance with the element in 'points'
27 surf_zoom, neig_zoom = mesh.surface(points=ele_in_zoom)
28 dist_zoom, neig_zoom = mesh.distance(points=ele_in_zoom)
29 volumes_zoom = mesh.volume(points=ele_in_zoom)
30
31 # Method 2. the mesh points are defined by their coordinates
32 # Lets focus on the central point 'i'. It has the coordinates
33 # (-6, -2), lets select it.
34 central_coord = np.array([[-6, -2]])
35 central_ele = mesh.points(coordinates=central_coord)[0]
36
37 # Lets find its first neighbors
38 neig_ce = mesh.neighbors(central_ele)
39
40 # Lets recover its voronoi cell
41 vor_ce = mesh.voronoi(central_ele)
42
43 # Then its properties
44 vertex_ce = vor_ce.vertices
45 ridges_ce = vor_ce.ridges
46
47 # Acess a cell volume, distances and surfaces
48 volumes_ce = vor_ce.volume
49 # With one specific neighbour
50 surfaces_ce = vor_ce.surface(neig_idx=[0, ])
51
```

```
52 # Or all of its neighbors
53 distances_ce = vor_ce.distance(neig_idx=np.arange(len(neig_ce)))
```

Script 4.8: API to recover the mesh properties from a **Mesh** instance.

4.3.5 Advanced mesh construction

This section describes three more advanced features of mesh construction beyond the simple one used to build Fig.4.6. The first feature allows one to control in more detail the meshing at the boundary between a coarsely meshed region and a refined one - c.f. Script.4.9. The second feature shows how to use Rectangular patterns with a discretization that varies along the different dimensions - c.f. Script.4.10. The last feature is the Finite pattern, it allows one to construct a mesh using an arbitrary set of points - c.f. Script.4.11. The Finite pattern construction time is long and requires considerable memory, hence it should only be used if the required mesh eludes construction by other means.

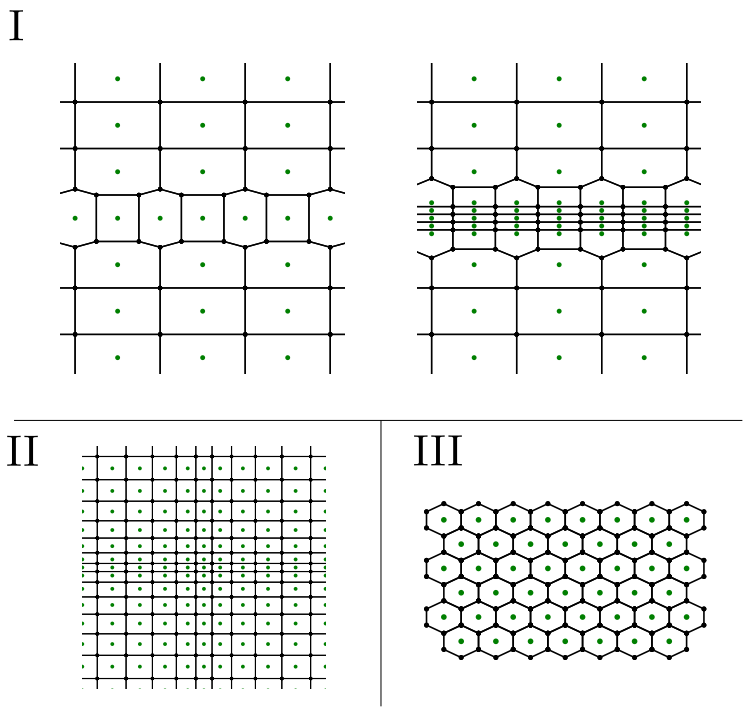


Figure 4.8: Three advanced features for mesh construction. (I) illustrates the use of the **refine()** method, c.f. Script.4.9. Left: nth=0 (default), Right: nth=2 (the second nearest neighbors of the refined mesh are also added). (II) Illustrates a rectangular mesh with a continuously varying mesh point spacing, c.f. Script.4.10. (III) illustrates a hexagonal voronoi cell mesh built using Finite, c.f. Script.4.11.

Script.4.9 shows a typical situation where one first defines the simulation region and meshes it somewhat coarsely (lines 6-8). Then one refines the mesh close to a region of interest (lines 11-12). The mesh constructed in this example is shown

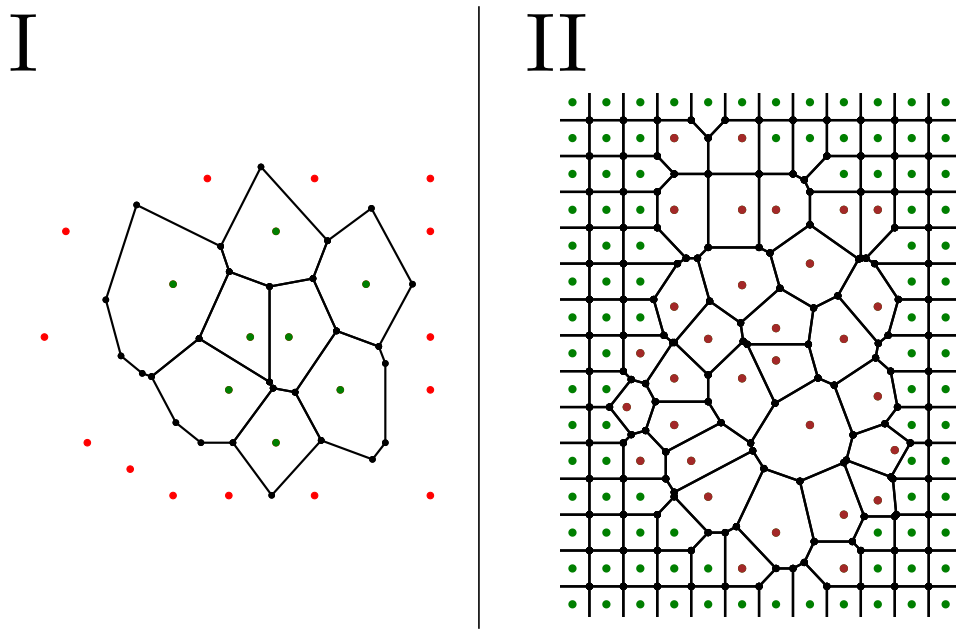


Figure 4.9: Two advanced features for amorphous mesh construction. (I) Illustrates an amorphous mesh created from a set of coordinates using the **Finite** pattern, c.f. Script.4.12. The green points are those inside *mesh_ amorhpous*. The red points are the first neighbors of the green points according to *amorphous_finite*. (II) Illustrates an amorphous mesh (brown points) embedded within a larger rectangular mesh. To create this mesh refine the region occupied by the amorphous mesh with the brown points. This is done by calling the **refine()** method with the brown points coordinates, c.f. Script.4.13.

on the left of Fig.4.8-I. The shape *fine_reg* is chosen such that when filled with *fine* points it contains a single row of points along the x-axis. See Fig.4.8-I left, it corresponds to using the **refine()** method as in lines 15-17. Notice that the voronoi cells in the *fine_reg* region have different shapes depending on the x coordinate. This might not be desirable in the simulation as it adds some sort of disorder due to the irregularity of the cells. To circumvent this issue, one may use a larger shape for the refined region (larger *fine_reg*). Alternatively, one may set the nth argument of the **refine()** method; this argument adds the nth$^{\text{th}}$ neighbors of the points to the refined region. In line 22 of Script.4.9 *nth=2*. The resulting mesh extends the refinement region by adding the first and second neighbors of the points that are inside *fine_reg* to the mesh (according to the *fine* pattern). The result is shown on the right side of Fig.4.8-I.

```
1
2 import numpy as np
3 from pescado.mesher import shapes, patterns, mesh
4
5 # Define the coarse pattern
6 coarse = patterns.Rectangular.constant((6, 6), center=(0, 0))
7 simulation_region = shapes.Box(
```

```
  lower_left=[-90, -20], size=[180, 40])

# Define the fine pattern
fine = patterns.Rectangular.constant((3, 1), center=(0, 0))
fine_reg = shapes.Box(lower_left=[-70, -0.9], size=[140, 1.8])

# Refine the coarse pattern without using the 'nth' parameters
mesh_s_ext = mesh.Mesh(
  simulation_region=simulation_region, pattern=coarse)
mesh_s_ext.refine(region=fine_reg, pattern=fine)

# With the 'nth' parameter
mesh_wext = mesh.Mesh(
  simulation_region=simulation_region, pattern=coarse)
mesh_wext.refine(region=fine_reg, pattern=fine, nth=2)
```

Script 4.9: The nth parameter in **refine()** ensures all voronoi cells inside *fine_ reg* are of the shape defined by the *fine* pattern.

Script.4.10 shows how one may construct a **Rectangular** pattern with a non-constant discretization step. In general a **Rectangular** pattern has its points $\vec{r}$ labeled by two integers n and m (in 2D, 3 in three dimensions): $\vec{r} = (x_n, y_m)$ with x_n and y_m arbitrary sequences of floats. The tuple (n, m) is referred to the **tag** of the pattern point $\vec{r}$. A simple choice $x_n = an$ provides a constant discretization step but one may use an arbitrary sequence. To define such a generalized pattern, one must provide both the sequences x_n, y_m *and* their inverses $n(x)$, $m(y)$ defined as $x_{n(x)} = x$ and $y_{m(y)} = y$ (or equivalently, $n(x_n) = n = m(y_n)$). This two way definition $x \leftrightarrow n$ ($y \leftrightarrow m$) is redundant but technically necessary since the pattern is defined for any (arbitrarily large) shape.

An example of such a construction with an exponentially increasing discretization step as one gets away from the center (with two different exponential along the x and y directions) is shown in Script.4.10. The resulting mesh is drawn in in Fig.4.8-III. Here x_n is *ticks_ X*, y_m is *ticks_ Y*, $n(x)$ is *tag2ticks_ X* and $m(y)$ is *tag2ticks_ Y*. The **Rectangular** object is initialized by providing its function $[x_n, y_m]$ (the **ticks** parameter) and its inverse $[n(x), m(y)]$ (the **tag2ticks** parameter). Notice the use of **np.round** in lines 12 and 24, it ensures the correct float to integer conversion.

```

import numpy as np
from pescado.mesher import shapes, patterns, mesh

# Define the ticks function along x
def ticks_X(tag):
    x = np.sign(tag) * np.abs(tag) ** 1.25  / 5
    return np.around(x, 2)

# Define the inverse ticks function along x
def tag2ticks_X(x):
    tag = np.round(
        np.sign(x) * (np.abs(x) * 5) ** (1/1.25),
        decimals=0)
    return (tag).astype(int)

# Define the ticks function along y
def ticks_Y(tag):
```

```
19      x = np.sign(tag) * np.abs(tag) ** 1.4  / 5
20      return np.around(x, 2)
21
22  # Define the inverse ticks function along y
23  def tag2ticks_Y(x):
24      tag = np.round(
25          np.sign(x) * (np.abs(x) * 5) ** (1/1.4),
26          decimals=0)
27      return (tag).astype(int)
28
29
30
31  # Define the Pattern and initialize the Mesh
32  fin_down = patterns.Rectangular(
33      ticks=[ticks_X, ticks_Y],
34      tag2ticks=[tag2ticks_X, tag2ticks_Y], center=(0, 0))
35  down_edge = shapes.Box(lower_left=[-53, -53], size=[106, 106])
36  mesh.Mesh(simulation_region=down_edge, pattern=fin_down)
```

Script 4.10: A **Rectangular** pattern can be used to construct a mesh whose discretization step varies arbitrarily along each direction.

Finally, it is possible to define a pattern from a set of points. To do so we use the **Finite** pattern. We shall illustrate this with three examples, c.f. an hexagonal mesh in Fig.4.8-III and two amorphous meshes in Fig.4.9-I-II.

The mesh in Fig.4.8-III is built by making a **Finite** pattern from the points in green, c.f. Script.4.11. To initialize a **Finite** one uses a numpy array of points $\vec{r} = (x, y)$, with x, y two floats defining the real space coordinates of $\vec{r}$ (3 floats in 3D). To obtain an hexagonal voronoi cell the points must define a triangular grid (green points Fig.4.8-III). One can think in terms of Bravais lattice: to obtain an hexagonal unit cell, one disposes the points in a triangular lattice. Lines 7 to 9 make the numpy array defining $\vec{r}$ (*coordinates*). Line 12 defines the hexagonal **Finite** pattern (*hex_finite*). Notice a **Finite** pattern is only defined within the region spanned by the coordinates given in the **points** parameter. For instance, *coordinates* span from $x = -41.89$ to $x = 41.89$ and $y = -76.18$ to $y = 76.18$. Therefore *hex_finite* is only defined within $|x| \leq 41.89$ and $|y| \leq 76.18$. Line 13 defines the rectangular region to be meshed. Line 16 makes the mesh of Fig.4.8-III.

In Fig.4.9-I-II we make two amorphous meshes. Lets start with Fig.4.9-I. First, the amorphous mesh is only defined for the green points, their voronoi cells are drawn by the black lines. The red dots do not belong to the mesh, however they ensure all the voronoi cells of the green points are closed. Line 8 of Script 4.12 defines the red points coordinates (*boundary_coord*). Line 9 defines the green points coordinates (*inside_coord*). Line 22 makes the **Finite** instance from *boundary_coord* and *inside_coord*. Notice, even though the *boundary_coord* points were used to define the **Finite** instance, they can not be used as mesh points. This is because they do not form a closed voronoi cell. Line 25 defines a rectangular region capturing only the *inside_coord*. Line 27 makes the mesh shown in Fig.4.9-I.

Finally, Fig.4.9-II shows a larger rectangular region (green points) refined with a smaller amorphous mesh (brown points). To make such mesh we start by making the rectangular mesh with a **Rectangular** pattern. See line 23 of Script 4.13, it

initializes a rectangular mesh (*mesh_ amorphous*). Then we need to refine it with the green points of Fig.4.9-II. First we define a numpy array with their coordinates, line 7 - *amorphous_ coordinates*. Then we define a region containing those points (*amorphous_ region*) - here we use the **Delaunay** shape, see Section 4.2. Lastly, we call the *mesh_ amorphous*.**refine()** method with *amorphous_ region* and *amorphous_ coordinates*. Notice there is no need to make a **Finite** pattern from *amorphous_ coordinates*.

```
1
2  import numpy as np
3  from pescado.mesher import shapes, patterns, mesh
4  from pescado.tools import meshing
5
6  # Define the coordinates of the mesh points
7  coordinates = 2 * np.pi / 3  * np.array([
8      [i + j, (i - j) *  np.sqrt(3)]
9      for i in range(-11, 11) for j in range(-11, 11)])
10
11 # Initialize a pattern and region defining the mesh
12 hex_finite = patterns.Finite.points(points=coordinates)
13 region_mesh = shapes.Box(
14   lower_left=np.array([-15, -15]), size=(30, 20))
15
16 mesh.Mesh(simulation_region=region_mesh, pattern=hex_finite)
```

Script 4.11: How to construct an hexagonal mesh from a set of points using **Finite**.

```
1
2  import numpy as np
3  from pescado.mesher import shapes, patterns, mesh
4  from pescado.tools import meshing
5
6  # Coordinates delimiting the boundary of the Finite pattern
7  # Required to have closed cells inside the pattern
8  boundary_coord = np.array([
9      [.2, 1.1], [.5, 1.1], [.2, -.1], [.5, -.1],
10     [-.1, .2], [-.1, .5], [1.1, .0],  [1.1, .2]])
11
12 # Coordinates inside the pattern
13 inside_coord = np.array([
14     [.0, .3], [.05, .5], [.1, .1], [.2, .05],
15     [.3, .0], [.3, .4], [.38, .6],
16     [.43, .0], [.43, .2], [.48, .3],
17     [.54, .5], [.54, .1], [.57, .3],
18     [.63, .6], [.63, .0], [.69, .2], [.75, .4],
19     [.9, .0], [.9, .2], [.9, .3], [.9, .5], [.9, .6]])
20
21 # Amorphous pattern
22 amorphous_finite = patterns.Finite.points(
23   points=np.concatenate((boundary_coord, inside_coord)))
24
25 region_mesh = shapes.Box(
26   lower_left=np.array([0.2, 0.1]), size=(0.55, 0.4))
27 mesh_amorphous = mesh.Mesh(
28   simulation_region=region_mesh, pattern=amorphous_finite)
```

Script 4.12: How to construct an amorphous mesh from a set of points using **Finite**.

```
1
2  import numpy as np
```

```
from pescado.mesher import shapes, patterns, mesh
from pescado.tools import meshing

# Coordinates of the amorphous mesh points
amorphous_coordinates = np.array([
    [.2, -.1], [.5, -.1], [.65, .23], [.6, .09],
    [-.1, .2], [-.1, .5], [.0, 1.1],  [.2, 1.1],
    [.3, .0], [.5, .05], [.1, .1], [.05, .2],
    [.0, .3], [.4, .3], [.6, .38], [-.14, .35],
    [.0, .43], [.2, .43], [.3, .48],
    [.5, .54], [.1, .54], [.3, .57],
    [.6, .63], [.0, .63], [.2, .69], [.4, .75],
    [.0, .9], [.2, .9], [.3, .9], [.5, .9], [.6, .9]])

amorphous_region = shapes.Delaunay(amorphous_coordinates)

region_mesh = shapes.Box(
  lower_left=np.array([-.35, -.35]), size=(1.2, 1.7))
rect_pat = patterns.Rectangular.constant(element_size=(0.1, 0.1))

mesh_amorphous = mesh.Mesh(
  simulation_region=region_mesh, pattern=rect_pat)
mesh_amorphous.refine(
  region=amorphous_region, coordinates=amorphous_coordinates)
```

Script 4.13: How to construct a regular mesh and then refine part of it with an amorphous mesh.

4.3.6 Algorithm for merging two meshes

The most important functionality of the *PESCADO* mesher is the **refine()** method (it takes a shape and a pattern as arguments). This method does in fact three things: first it removes the points of the mesh that belongs to the pattern; then it creates a new mesh by selecting the points of the pattern that belong to the shape; last it merges the old mesh with the new one. This last operation is somewhat delicate and in this section we describe the algorithm used to perform it efficiently.

4.3.6.1 Merging two meshes

The problem is defined as follows: given two meshes defined by their set of mesh points $\{\vec{r}_i\}$ and $\{\vec{r}_i{}'\}$ whose Voronoi cells $\mathcal{C}_i$ and $\mathcal{C}'_i$ have already been calculated, calculate the Voronoi diagram of $(\{\vec{r}_i\} \setminus A) \cup \{\vec{r}_i{}'\}$, where A is the list of the points of $\{\vec{r}_i\}$ that are to be removed.

The strategy used in *PESCADO* is to find out the list of points whose Voronoi cells will be affected by the new points and only recalculate the Voronoi cells for this (usually small) subset of points using Qhull [Barber *et al.* 1996]. A point $\vec{r}_i$ needs to have its Voronoi cell recalculated if and only if one of the two following things happen:

- this point has lost one of its neighbors during the first (extrusion) stage of the refinement. Indeed for these points the number of voronoi neighbors is smaller than the number of ridges in their voronoi cell. We call these points

'incomplete'. They are identified as the neighbors of the points of A that do not belong to A.

- this point has acquired one or more new neighbors, i.e. one of the new point $\{\vec{r_j}'\}$ is close enough to affect its Voronoi cell. Note that $\vec{r_j}' \in \mathcal{C}_i$ is a sufficient condition for these to happen but by no mean necessary. Sometimes $\vec{r_j}'$ being in a neighboring cell or even second nearest neighbor cell is sufficient to disturb the cell $\mathcal{C}_i$. Below we define the concept of safety region of point i: when $\vec{r_j}'$ is inside this safety region, the Voronoi cell $\mathcal{C}_i$ needs to be recalculated.

The above criteria are used on $\{\vec{r}_i\}$. Then the converse criteria are used on $\{\vec{r_i}'\}$ and one arrives at the complete list of points that need to be recalculated. To this list, one adds their neighbors to make these points 'complete' and the list is given to qhull. Last, one needs to perform the book-keeping operation of merging all the obtained Voronoi cells (existing with recalculated).

4.3.6.2 The voronoi safety region

The key part of the merging algorithm is the definition of the safety region for each point $\vec{r}_i$ (or $\vec{r_i}'$). The question is, for a given point $\vec{r}_i$, define a condition on any point $\vec{r}$ for this point to affect the Voronoi cell $\mathcal{C}_i$. More precisely, we will define a 'conservative' safety region for efficiency reasons, i.e. it is OK to recalculate a few extra points even if their Voronoi cell will not actually change.

The safety condition is best understood visually. Fig.4.10 shows a point $\vec{r}$ (the light blue cross) intruding in the Voronoi cell of $\vec{r}_i$ (the black circle at the center of the gray Voronoi cell). Also shown is the perpendicular bisector of $(\vec{r}, \vec{r}_i)$. When this perpendicular bisector has an intersection with the Voronoi cell (right panel of Fig.4.10), the Voronoi cell needs to be recalculated. Otherwise, as illustrated on the left panel of Fig.4.10, the Voronoi cell is unaffected. The boundary between the two cases corresponds to the perpendicular bisector touching the vertex point $\vec{v}$ (green point). This translates into the following condition,

$$\left[(\vec{v} - \vec{r}_i) - \frac{\vec{r} - \vec{r}_i}{2}\right].(\vec{r} - \vec{r}_i) \leq 0 \tag{4.6}$$

Eq.(4.6) can be written as,

$$\|\vec{r} - \vec{v}\|^2 \leq \|v - \vec{r}_i\|^2 \tag{4.7}$$

i.e. $\vec{r}$ is inside the disk (a sphere in 3D) of center $\vec{v}$ and radius $\|v - \vec{r}_i\|$. It follows that the overall safety region is given by the union of all the disks centered on all the vertices of $\mathcal{C}_i$ as shown on the right hand side of Fig.4.11(b) (small green circles). In practice keeping track of all the disks associated with the different vertices is somewhat cumbersome. Instead we consider a slightly larger 'conservative' safety region (hence we might unnecessarily recalculate a few extra points) by using the light gray large disk in Fig.4.11(b) (limited by a black dashed line) of equation

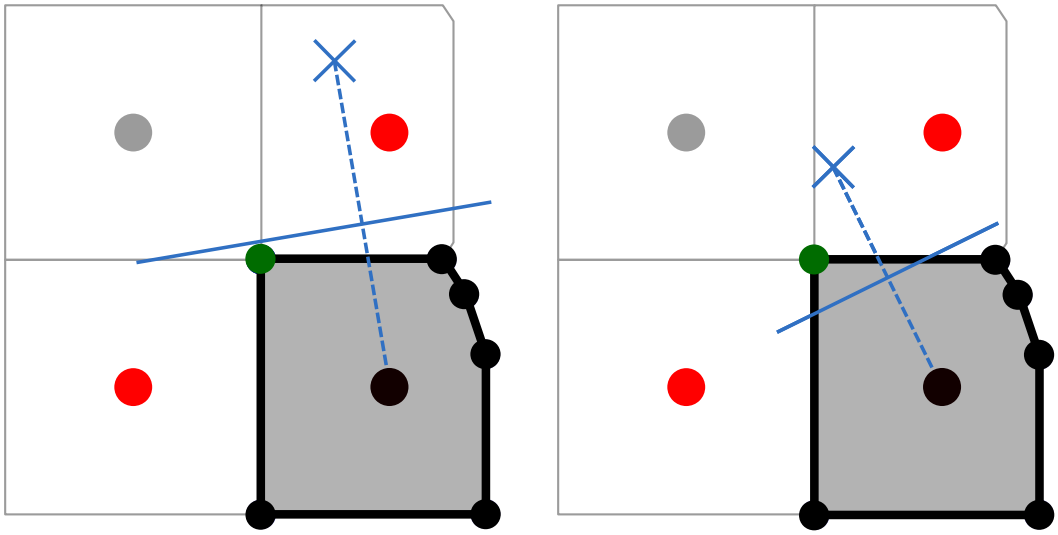


Figure 4.10: Schematic for the safety condition. Left: the point $\vec{r}$ (blue cross) does not affect the Voronoi cell in gray. Right: $\vec{r}$ affects the gray Voronoi cell that will need to be recalculated. The blue line is the perpendicular bisector of $\vec{r}$ and the center $\vec{r}_i$ of the gray Voronoi cell. See text for more details.

$$\|\vec{r} - \vec{r}_i\| \leq 2 \max_{\vec{v}} \|\vec{v} - \vec{r}_i\| \tag{4.8}$$

4.3.7 Defining a custom pattern in *PESCADO*

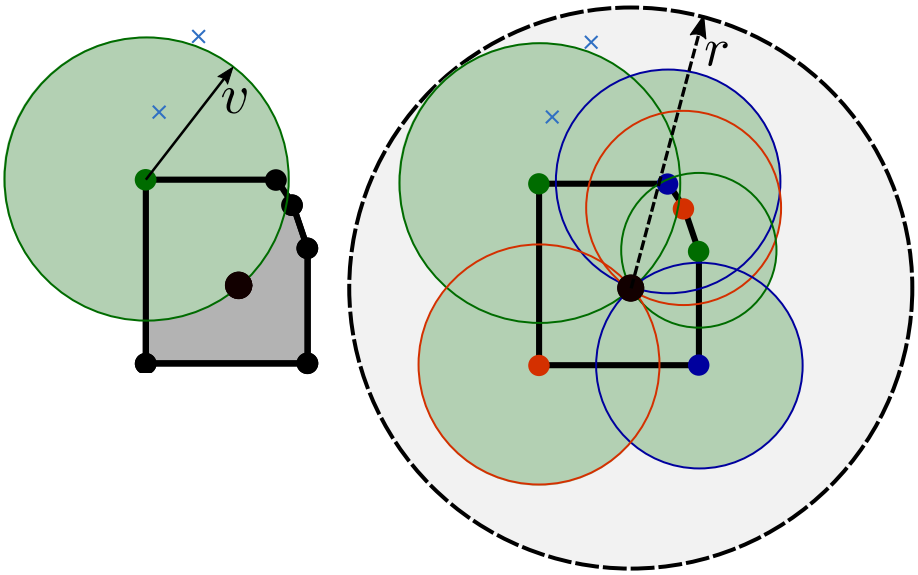
A pattern in *PESCADO* is an abstract class. *PESCADO* currently contains two instantiable patterns that derive from this abstract class (Rectangular and Finite) but users could define more if they want. Future releases of *PESCADO* might contain patterns for e.g. arbitrary lattices with complex unit cells (so that the center of the Voronoi cells actually match the positions of the atoms) or a discretization in terms of spherical coordinates.

To define a pattern a user must create a class (derived from the abstract class **Pattern**) that implements the following four methods:

- **__call__**(*tags*) : Returns the coordinates $\vec{r}$ for an array of points. In a **Pattern** each point is defined by its **tag** (an immutable python object). For instance, in a 2D **Rectangular** pattern the tag is a tuple (n, m). Therefore, **__call__**() has one argument, an array of **tags**, and returns one output, an array of point coordinates.

- **inside**(*shape*) : Returns the **tags** inside a **Shape**.

- **first_neighbours**(*points_tag*) : Returns the first neighbors for a given array of **tags**.



(a) (b)

Figure 4.11: (a) Safety condition associated with the green vertex. (b) Overall safety region for all vertices (green disks) and conservative (larger) safety region used in practice in *PESCADO* algorithm (light gray disk delimited by a dashed line).

- **ridges_and_vertices**(*points_tag*, *neig_tags*, *neighbors*, *point_neighbors*) : Returns the voronoi ridges and its vertices separating a set of points from their first neighbors. For a detailed description of each parameter, see the method description in the source code.

We refer the reader to the **Finite** class implementation as an example. Regarding testing the newly written class, we refer the reader to the script 'pescado.mesher.tests.tests_patterns.py' for inspiration.

With the four methods above we can define any type of pattern, let it be irregular or regular. In a irregular pattern there is no relationship between the position of its points, e.g. the amorphous pattern in Section 4.3.5. In contrast, in a regular pattern the points are distributed regularly in space. Hence from the point coordinate one can calculate the position of its nearest neighbors and its voronoi cell properties. Therefore, for a regular pattern one does not need to store in memory a list of all first neighbors and voronoi cell information. One can simply store an array of coordinates and their tags, the rest can be calculated as required by the *PESCADO* mesher. In *PESCADO* we have implemented a **Regular** class, that inherits from **Pattern**. It is an abstract class that defines the interface the *PESCADO* mesher requires to generate as it needs the mesh geometrical information (distances, surfaces, volumes and safety radius). Therefore, we strongly recommend the user to inherit from **Regular** when implementing a custom regular pattern. In addition

to the four methods mentioned above, a class inheriting from **Regular** must also define **_geometrical_properties()**. In essence, the purpose of this method is to return the distances, surfaces, volumes and safety radius for a given tag. The advantage is that the user can optimize the calculation of each one of these geometrical properties. This comes in hand when dealing with large 3D systems, and provides a consequent gain in both time and memory. The **Rectangular** pattern inherits from **Regular** and can be used as an example.

4.4 The electrostatics solver

The main solver of *PESCADO* solves the Poisson equation. Actually, it solves a generalization of the Poisson equation, a *generalized* Helmholtz equation. It reads,

$$\nabla \cdot (\varepsilon(\vec{r})\nabla U(\vec{r})) = -e[n(\vec{r}) + \rho(\vec{r})U(\vec{r})]. \tag{4.9}$$

Without the local density of states $\rho(\vec{r})$ term LDOS, Eq.(4.9) is simply the Poisson equation with $U(\vec{r})$ the electric potential, $n(\vec{r})$ the density of charges (positive for holes, negative for electrons) and $\varepsilon(\vec{r})$ the dielectric function. As we shall see, the LDOS term allows one to incorporate some quantum mechanical effects at a semi-classical level. Since the equation remains linear in presence of the LDOS term, it does not render the problem more complicated to solve. In practice the LDOS term is moved to the left hand side of the equation.

4.4.1 Finite volume discretization

To obtain a discrete problem from the continuous equation, we follow the usual route of finite *volumes* discretization. We start by integrating Eq.(4.9) over one Voronoi cell i and use the Green-Ostrogradsky theorem to arrive at

$$\sum_j \Phi_{ij} = eQ_i + e\tilde{Q}_i \tag{4.10}$$

where Q_i is the total charge inside the cell,

$$Q_i = \int_{\mathcal{C}_i} d\vec{r}\, n(\vec{r}), \tag{4.11}$$

$\tilde{Q}_i$ the total charge inside the cell induced by the LDOS term and Φ_{ij} the flux of the electric field through the planar surface S_{ij} that separates cell i from its neighbor j,

$$\Phi_{ij} = \int_{S_{ij}} \varepsilon(\vec{r})\vec{E}(\vec{r}) \cdot \vec{n}\, \mathrm{d}S \tag{4.12}$$

$\vec{E} = \vec{\nabla}U$ is the electric field and $\vec{n}$ is the unit vector perpendicular to S_{ij}. We define E_{ij} as the average of the normal part of $\vec{E}$ on the line that connects $\vec{r}_i$ to $\vec{r}_j$,

$$E_{ij} = \frac{1}{d_{ij}} \int_{\vec{r}_j}^{\vec{r}_i} \mathrm{d}\vec{r} \cdot \vec{E} = \frac{1}{d_{ij}} (U_j - U_i) \tag{4.13}$$

where d_{ij} is the distance $|\vec{r}_i - \vec{r}_j|$ and U_i the electric potential at the center of cell i. At this stage, the equations are exact and satisfy a discrete version of the Gauss theorem [Eq.(4.10)] as well as a discrete version of the circulation theorem, i.e. for any path that makes a loop $i_1 ... i_n$ with $i_{n+1} = i_1$ of straight lines between $\vec{r}_\alpha$ and $\vec{r}_{\alpha+1}$,

$$\oint d\vec{r} \cdot \vec{E} = \sum_{\alpha=1}^{n} E_{i_\alpha i_{\alpha+1}} d_{i_\alpha i_{\alpha+1}} = 0 \tag{4.14}$$

To close the system of equation, we suppose that the electric field varies sufficiently smoothly so that the average value of $\vec{E}.\vec{n}$ over the surface S_{ij} is equal to its average E_{ij} over the line that connects $\vec{r}_i$ to $\vec{r}_j$. We arrive at,

$$\Phi_{ij} \approx \frac{\varepsilon_{ij} S_{ij}}{d_{ij}} (U_j - U_i) \tag{4.15}$$

with the average dielectric constant ε_{ij} given by

$$\varepsilon_{ij} = \frac{2\varepsilon_i \varepsilon_j}{(\varepsilon_i + \varepsilon_j)} \tag{4.16}$$

where ε_i is the dielectric constant of cell i. Finally, defining

$$\rho_i = \int_{\mathcal{C}_i} d\vec{r}\, \rho(\vec{r}), \tag{4.17}$$

we obtain $\tilde{Q}_i \approx \rho_i U_i$ and finally arrive at,

$$Q_i = \sum_j (C_{ij} - \delta_{ij}\rho_j) U_j \tag{4.18}$$

with the capacitance matrix C_{ij} defined as,

$$C_{i \neq j} = -\frac{\varepsilon_{ij} S_{ij}}{e d_{ij}} \tag{4.19}$$

for neighboring cells,

$$C_{ii} = -\sum_{j(i)} C_{ij} \tag{4.20}$$

for the diagonal part ($j(i)$ stands for the neighbors of cell i) and $C_{ij} = 0$ otherwise.

An important property of the finite volume discrete problem is that it is a valid discrete electrostatic problem even if a very coarse discretization is used, in the sense that both the discrete Gauss theorems and circulation theorems are satisfied.

In particular $C_{ij} = C_{ji}$, $\sum_j C_{ij} = 0$ and $C_{i\neq j} \leq 0$ which implies that the matrix is positive: $\forall U_i$,

$$\sum_{ij} U_i C_{ij} U_j = -\frac{1}{2} \sum_{i \neq j} (U_i - U_j)^2 C_{ij} \geq 0 \tag{4.21}$$

We end this discussion with a note on dimensionality. The electrostatic problem solved in *PESCADO* always corresponds to the three dimensional (physical) space. However, in some cases the system is invariant along one or two directions and in that case the problem reduces *effectively* respectively to a two dimensional or a one dimensional problem. The derivation follows accordingly with one minor modification: the charge Q_i becomes a charge per unit length (effective 2D problem) or per unit surface (1D problem). All electric potentials in *PESCADO* are in Volts and all distance in nanometers. The units of the different quantities follow, as shown in Table.4.1.

	U_i	Q_i	ρ_i
1D	V	nm^{-2}	$nm^{-2}.V^{-1}$
2D	V	nm^{-1}	$nm^{-1}.V^{-1}$
3D	V	-	V^{-1}

Table 4.1: *PESCADO* default units for 1D, 2D and 3D systems. U_i is the voltage, Q_i the charge and ρ_i the local density of states.

4.4.2 Neumann, Dirichlet, Helmholtz (and Flexible) cells

When solving an electrostatic problem, the user can define three different sorts of cells, depending on the physics occurring in the corresponding region of space:

- The “Neumann" cells (N) are the cells for which the charge Q_i is known, $\rho_i = 0$ and one seeks to calculate the potential U_i. These are the “normal" cells of an electrostatic problem which include the regions with dielectrics (including vacuum) and regions with fixed densities of charge such as dopant regions.

- The “Dirichlet" cells (D) are the cells where the potential U_i is known, $\rho_i = 0$ and one seeks to calculate the total charge in the cell Q_i. Typically, the Dirichlet sites correspond to metallic region such as electrostatic gates (with infinite density of states) where the electric potential is imposed by e.g. a voltage source.

- The “Helmholtz" cells (H) are the cells for which $\rho_i \neq 0$ and one seeks to calculate the potential U_i. These are typically the active (quantum) part of the device where one has fixed the *electro-chemical* potential but not the electric potential.

Each cell belongs to one of the three types (and only one) as defined by the user (the default being Neumann). Note that, in principle, Helmholtz cells cover the three cases since a Dirichlet cell corresponds to $\rho_i = \infty$ and a Neumann cell to $\rho_i = 0$. However, for both numerical stability and efficiency reasons, it is better to sort the cells according to these three categories. Actually, *PESCADO* possesses a fourth category, the flexible cells (F). A flexible cell is a cell whose status (N,D or H) is defined at the latest possible moment, just before solving the linear problem. Flexible cells exist purely for efficiency reasons because, in *PESCADO* 's self-consistent algorithms, some cells need to change of status from one call to the solver to the next; flexible cells allows one to pre-calculate the associated matrix elements.

We now write the linear problem Eq.(4.18) according to its 3×3 block structure, denoting U_H, U_N and U_D (Q_H, Q_N and Q_D) the vectors of electric potentials (charges) in the set of H, N and D cells. ρ_H is the diagonal matrix defined as $(\rho_H)_{ij} = \delta_{ij}\rho_i$. We get,

$$\begin{bmatrix} C_{HH} - \rho_H & C_{HN} & C_{HD} \\ C_{NH} & C_{NN} & C_{ND} \\ C_{DH} & C_{DN} & C_{DD} \end{bmatrix} . \begin{bmatrix} U_H \\ U_N \\ U_D \end{bmatrix} = \begin{bmatrix} Q_H \\ Q_N \\ Q_D \end{bmatrix} \tag{4.22}$$

Moving the unknown variables of the equation (Q_D, U_N and U_H) to the left hand side and the known input (U_D, Q_N, Q_H) to the right hand side, we arrive at the linear problem actually solved by *PESCADO* electrostatic solver,

$$\begin{bmatrix} C_{HH} - \rho_H & C_{HN} & 0 \\ C_{NH} & C_{NN} & 0 \\ C_{DH} & C_{DN} & -\mathbb{I} \end{bmatrix} . \begin{bmatrix} U_H \\ U_N \\ Q_D \end{bmatrix} = \begin{bmatrix} \mathbb{I} & 0 & -C_{HD} \\ 0 & \mathbb{I} & -C_{ND} \\ 0 & 0 & -C_{DD} \end{bmatrix} . \begin{bmatrix} Q_H \\ Q_N \\ U_D \end{bmatrix} \tag{4.23}$$

which we call the "mixed Neumann-Dirichlet-Helmholtz Problem". Taking advantage of the sparsity of the capacitance matrix, solving Eq.(4.23) can be done with efficient linear problem packages such as the MUMPS package [Amestoy *et al.* 2001, Amestoy *et al.* 2006].

Solving an electrostatic problem in *PESCADO* is done in two steps. In the first, one defines the problem and its discretization in order to form the set of linear equations Eq.(4.23). This is the role of the **Poisson.ProblemBuilder** class as illustrated in Script 4.1. In the second, one actually solves the linear problem and extracts the physical quantities. This is the role of the **Poisson.Problem** class as illustrated in Script 4.2. In the rest of this section, we document the usage of these two classes.

4.4.3 Electrostatic solver API: The ProblemBuilder class

Let us follow the step by step construction of an electrostatic problem, following Script 4.14, in order to construct the system shown in the top panel of Fig.4.12. This 2D system consists of a two dimensional electron gas (red), some dielectric layers (salmon), part of which contain dopants (green), two metallic gates (blue)

and some air/vacuum (white). We need to define the mesh and name the different regions of interest in order to set their properties: the type of each site (N,D,H or F) and the value of the dielectric constant. The values of the inputs (potential or charge depending of the site type) will be set later, just before solving the linear system Eq.(4.23). Internally an instance of **ProblemBuilder** merely contains the mesh of the system (which can be accessed through the **mesh** attribute) and the different (named) **shapes** of the different regions. Its role is to eventually produce an instance of **Problem** which contains the capacitance matrix.

The first part of the script meshes the system. The method **initialize_mesh()** does exactly what its name suggests and takes the shape of the simulation box and the pattern to be used as arguments. Subsequent refine steps are identical to what was described in Section 4.3, the **refine_mesh()** method being a simple warper for a call to **Mesh.refine()**, c.f. Script 4.7. It is possible to retrieve the **Mesh** instance **ProblemBuilder** creates, c.f. line 23. In the second part of the script (after line 24), one defines various shapes for the regions of different colors of Fig.4.12 and then call various methods to assign these shapes to a cell type or to set the value of the electric permittivity. The methods to set the cell type are respectively **set_dirichlet**, **set_Helmholtz** and **set_neumann**. Besides the shape argument, they also take a name argument (**setup_name**) that will be used later to extract specific information about these regions. In addition to the three main methods above, *PESCADO* also implements **set_metal**. It not only sets the cell type to Dirichlet, but also mark it as a metal. This is important, because for metallic sites there is no point in defining the value for the electric permittivity. The **set_metal** method takes the same argument as the other three. By default, all cells are assigned to Neumann (N), so that **set_neumann** is actually only used to name a N region (here, the doped region).

In our example, we set the cells inside the electrostatic gates (blue) as metal type and name them *gate*, see Line 29 of Script 4.14. The sites inside the 2DEG (red) are helmholtz sites and we name them *2deg*, see Lines 31-32. Using the method **set_relative_permittivity** we can change the value of the relative dielectric permittivity, ε_r, from the default value of $\varepsilon_r = 1$. Hence by default the dielectric permittivity in a cell is $\varepsilon = \varepsilon_0$, the vacuum permittivity. In line 39 we set $\varepsilon_r = 12$ at the entire dielectric region, including the 2DEG.

```
1  from pescado.mesher import patterns, shapes
2  from pescado.poisson import ProblemBuilder
3
4  # Define the ProblemBuilder
5  pb = ProblemBuilder()
6
7  ### Define the Finite volume mesh
8
9  # Define the simulation region and mesh point spacing
10 system = shapes.Box(lower_left=[-1000, -300], size=[2000, 693])
11 rect_pattern = patterns.Rectangular.constant(element_size=(10, 10))
12
13 # Make the mesh
14 pb.initialize_mesh(
15   simulation_region=system, pattern=rect_pattern)
```

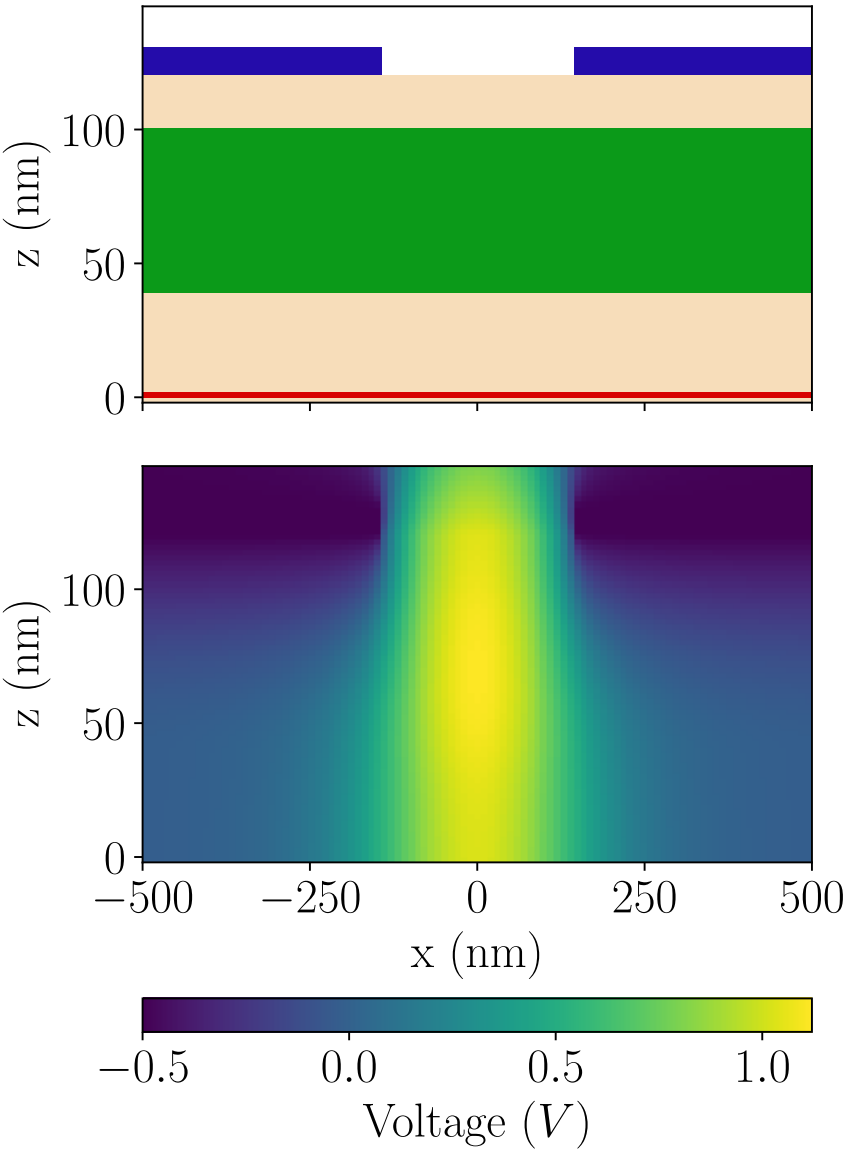


Figure 4.12: Top panel: schematics of the nanowire device constructed in Script 4.14. The system is invariant by translation along the y dimension so that the simulation reduces to a 2D one. The different colors stand for the different regions of interest: gates (blue), dopants (green), two-dimensional gaz (red), vacuum (white) and other dielectric (salmon). Bottom panel: electric potential profile calculated using Script.4.17

```
16
17 # Refine the mesh
18 fine_pattern = patterns.Rectangular.constant(element_size=(2, 2))
19 ref_region = shapes.Box(lower_left=[-1000, -3], size=[2000, 200])
20 pb.refine_mesh(region=ref_region, pattern=fine_pattern)
21
22 ### Access the Mesh instance
23 mesh_pb = pb.mesh
24
25 ### Split sites into N, D, H and F.
26 # Define local boundary conditions
27 gate = (shapes.Box(lower_left=[-1000, 120.1], size=[850, 10])
28         | shapes.Box(lower_left=[150, 120.1], size=[850, 10]))
29 pb.set_metal(region=gate, setup_name='gate')
30
31 twodeg = shapes.Box(lower_left=[-1000, -1], size=[2000, 2])
32 pb.set_helmholtz(region=twodeg, setup_name='2deg')
33
34 dopants = shapes.Box(lower_left=[-1000, 40], size=[2000, 60])
35 pb.set_neumann(region=dopants, setup_name='dopants')
36
37 ### Set relative dielectric permittivity
38 dielectric = shapes.Box(lower_left=[-1000, -100], size=[2e3, 220])
39 pb.set_relative_permittivity(val=12, region=dielectric)
40
41 ### Create the capacitance matrix
42 pd = pb.finalized()
43 # Save the obtained Problem (optional)
44 pd.save('elec_problem_tutorial')
```

Script 4.14: Tutorial on making a finite volume mesh and discretizing the device on Fig.4.12. It explains how to use **pescado.poisson.ProblemBuilder**.

The last two lines of Script 4.14, respectively, generate and save on disk the **Poisson.Problem** that will be subsequently solved. The saving step is optional but can be handy when the problem needs to be solved many times (e.g. different machines in a parallel calculation), as it avoids calculating the capacitance matrix more than necessary. In particular, **ProblemBuilder** can have a significant memory footprint (as it stores the whole mesh) so it is a good practice to delete it when the **Problem** instance has been initialized. In contrast, **Problem** only stores the capacitance matrix, the coordinates of the mesh points and their volume, hence is a much smaller object.

Before we finish on the **ProblemBuilder** class, let us discuss the functions used to visualize the partition of the system in terms of D,N,H and F cells (**plot_boundary_condition()**) and in terms of the various named regions (**plot_system_regions()**). These tools help, upon building a system, to make sure that one has defined the various regions properly. They work for 1D, 2D or 3D systems.

For 1D and 2D systems, **plot_boundary_condition()** takes two parameters, *pattern* (a **Pattern**) and *region* (a **Shape**). The function first generates the points of the *pattern* that are inside the *region*. For each of these points $\vec{u}$, **ProblemBuilder** finds the point $\vec{r}$ in the mesh which is the closest and attributes the type (N,D,H) of $\vec{r}$ to $\vec{u}$ with one color per type. The advantage of this approach is that one

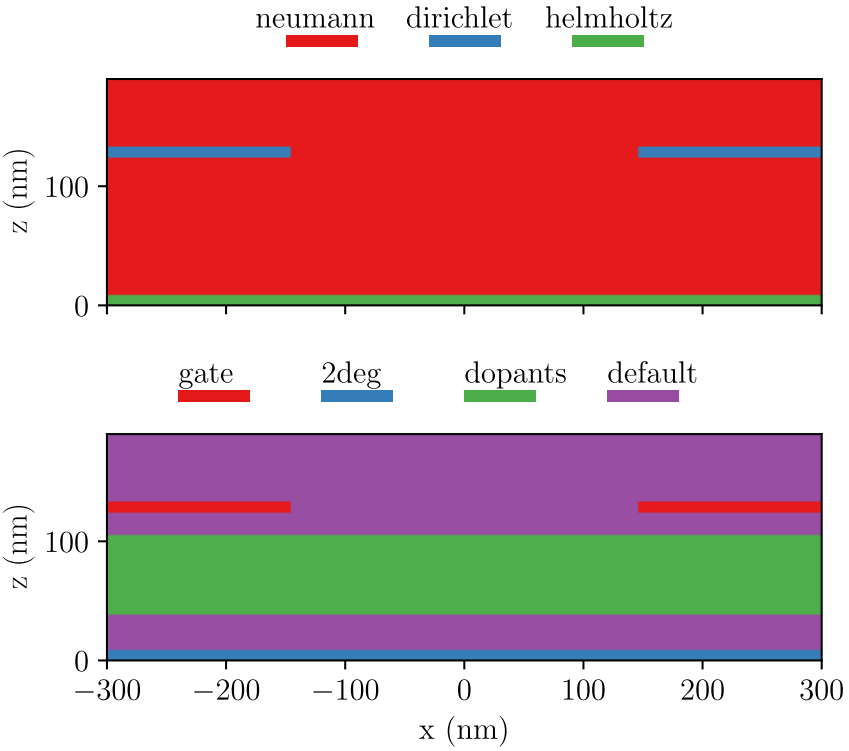


Figure 4.13: Top colorplot shows the regions according to their boundary conditions. Bottom colorplot shows the regions according to their **setup_name**. Obtained with Script 4.15.

obtains a regular grid of points $\vec{u}$, easy to plot and manipulate while the original mesh $\vec{r}$ may be highly unstructured. Calling **plot_boundary_condition()** does not generate the plot immediately. It returns a function. Calling this function with a matplotlib **axis** generates the actual plot. **plot_system_regions()** works very similarly but plot the different tags instead of the types of sites. For a tutorial on **plot_boundary_condition()** and **plot_system_regions()** for 3D systems we refer to Section 4.4.5.

Lines 3-11 in Script 4.15 generates the top figure in Fig.4.13 for the device on Fig.4.12. Line 14-15 makes the bottom colorplot in Fig.4.13 for the device on Fig.4.12. The function can be called with optional arguments. The ax argument defines the **matplotlib.pyplot.gca()** instance containing the plot. The remaining arguments are passed down to **matplotlib.imshow()**.

```
1 import matplotlib.pyplot as plt
2
3 fig, ax = plt.subplots(2, 1, sharex=True, figsize=(5, 15))
4
5 # Define the plot region and the plot pattern
6 plot_region = shapes.Box(lower_left=[-300, -3], size=[600, 200])
7
8 # Plot the boundary condition
9 bc_profile = pb.plot_boundary_condition(
```

```
  region=plot_region, grid_step=10)
bc_profile(ax=ax[0])

# Plot the region
region_profile = pb.plot_system_regions(
  region=plot_region, grid_step=10)
region_profile(ax=ax[1])
```

Script 4.15: Tutorial on plotting the regions according to their boundary condition and **setup_name**.

4.4.4 Electrostatic solver API: The Problem class

We are now in possession of a **Problem** instance for the device shown in Fig.4.12. There are two remaining steps to solve the electrostatic problem: provide the missing necessary input (charge in the Neumann regions Q_N, voltages at the gates U_D and density of states ρ_H and charges Q_H in the active Helmholtz regions) and solve the corresponding linear problem.

Each mesh point in a **Problem** is identified by an index that ranges from 0 to $N-1$ where, N is the total number of mesh points in the problem. An instance of **Problem** contains several attributes and methods that facilitate the handling of the indices (used internally for solving) and the real coordinates.

- **npoints** is the number of mesh points N.
- **coordinates** is a 2D numpy array of coordinates where the first dimension corresponds to the site index and the second to the x, y and z axis (respectively for values 0, 1 and 2).
- **volumes** is a 1D numpy array that contains the volume Ω_i of each cell.
- The **points_inside(*shape*)** method returns a 1D numpy array that contains the indices of the mesh points inside *shape*.
- The **points(*name*=*None*, *coordinates*=*None*)** method retrieves the indices for some mesh points. To indicate which points we are looking for, we can either give the *name* or *coordinates* argument. If *name* is given (string), then the method returns the indices inside a region tagged *name*. Otherwise if *coordinates* (a 2D numpy array whose second dimension labels the x, y and z axis) is given, then it returns the indices of the mesh points with such coordinates.

4.4.4.1 Using SparseVector to represent the inputs and outputs

The input (U_D, Q_N, ρ_H and Q_H) and output (Q_D,U_H,U_N) of the electrostatic problem are vectors that are often defined on small regions of the device and have value zero everywhere else. To manipulate these values, we use the concept of **SparseVector**, a simple class that implements a sparse 1D array.

A **SparseVector** contains a list of index and a list of associated values (both lists have the same sizes). For instance the vector $(0, 1.1, 0, 0, 5.2, 0, 0, 0, 0, 0)$ can be encoded into a **SparseVector** that stores the list of non zero indices $(1, 4)$ and the list of associated values $(1.1, 5.2)$. Lines 5-7 in Script 4.16 show the construction of this **SparseVector** with a default value of zero for all elements that are not explicitly set. A **SparseVector** can also have an undefined default value. In that case unset indices have an undefined value. This will be used, for instance, for defining U_D since the input potential is undefined on Neumann sites. A **SparseVector** can be accessed and sliced like a regular normal 1D numpy array. Line 10 shows an example and returns a numpy array with the values of the first 6 elements (including several zeros). Accessed elements can be modified (see Line 18) and the whole **SparseVector** can also be multiplied by a constant (see Line 21). It is also possible to add two **SparseVector** together (Line 27). A **SparseVector** can also be extended with new non-zero elements (Line 30). In that case only the *new* indices are included in the **SparseVector**, the preexisting ones are not modified. Last, the **extract()** method of **SparseVector** produces another **SparseVector** restricted to the list of sites that are given to the *indices* argument (see Line 35).

```
1  from pescado.tools import SparseVector
2  import numpy as np
3
4  # Defining a SparseVector
5  sv_1 = SparseVector(
6    values=np.array([1.1, 5.2]),
7    indices=np.array([1, 4]), default=0)
8
9  # Access the elements 0 to 5
10 sv_1_elements = sv_1[np.arange(0, 6)]
11
12 # Define the U_D input
13 gate_idx = pd.points(name='gate')
14 voltage_in_sv = SparseVector(
15     indices=gate_idx, values=np.ones(len(gate_idx)) * -0.5)
16
17 # Multiply the values of the gate_idx elements by -1
18 voltage_in_sv[gate_idx] = voltage_in_sv[gate_idx] * -1
19
20 # Or multiply all values by -1
21 voltage_in_sv *= -1
22
23 # Add two sparse vectors
24 voltage_in_sv_II = SparseVector(
25     indices=gate_idx, values=np.ones(len(gate_idx)) * -1)
26
27 voltage_in_sv_II = voltage_in_sv + voltage_in_sv_II
28
29 # Define the voltage somewhere else
30 voltage_in_sv_II.extend(SparseVector(
31     indices=np.array([0, 1, 10, 20]),
32     values=np.array([1, 10, 15, 20])))
33
34 # Extract the voltage at the gates
35 voltage_in_sv_II = voltage_in_sv_II.extract(indices=gate_idx)
```

Script 4.16: Tutorial on **SparseVector**. The *pd* variable refers to the **Problem**

instance from line 41 Script 4.14

Defining an input such as U_D is done using a combination of the properties of **SparseVector** and of the **Problem** considered. Lines 13-15 of Script 4.16 define a U_D **SparseVector** for the system on Fig.4.12. There, the Dirichlet sites are located at the *gate*, and we set U_D to be equal to $-0.5V$ on the gate sites and undefined on other sites. Line 13 recovers the indices of the mesh points inside the gate region (blue region of Fig.4.12). Then, lines 14 and 15 initialize the **SparseVector** defining U_D.

4.4.4.2 Setting the inputs and solving the electrostatic problem

Such a manual construction of the input vectors is somewhat cumbersome and error prone. **Problem** contains a **sparse_vector()** method that facilitates the creation of these inputs. **sparse_vector()** defines the input value for an entire region of the system. For example, line 4 of Script 4.17 generates the same U_D **SparseVector** as lines 13-15 of Script 4.16. The *val* parameter sets the values of the **SparseVector** elements (a float or a function of the coordinates). The *name* parameter describes the region (a name of the region or a **Shape**). Lines 13 to 16 illustrate how to define a **SparseVector** using a function as the *val* parameter. The function should take a numpy array of real space coordinates and return numpy array of the values to be set at the input coordinates. Line 19 defines Q_N in a similar manner to U_D, using a float as *val*.

To finish setting the inputs and solve the electrostatic problem, we need to define both Q_N and ρ_H. The former is defined line 7 of Script 4.17. Note that, for convenience, there are two different ways to set the charge: (i) one can either define the total charge in the cell (the default; in 3D this quantity is dimensionless. It has dimension nm^{-1} in 2D and nm^{-2} in 1D) using the **charge** argument to the **solve** function. Or (ii), one can define the value of the **charge_density** in the cell and the charge is computed by *PESCADO* by multiplying the value of the charge density by the volume of the cell. The charge density always has units of nm^{-3}. In our case, we want to set an density of electrons for the 2DEG of $2.10^{15}m^{-2}$. Since the size of the cell along the z axis is $60nm$ (see Line 37 of Script 4.14) , this translates into a charge density of $2.10^{15}/60/10^{18} = 0.33\ 10^{-4}nm^{-3}$. Similarly, we want to set the density of states to the 2D bulk value of 2DEG given by the standard formula $\rho_{2deg} = 2m_e^*/(\pi\hbar^2)$ ($m_e^* = 0.067m_e$: effective electron mass in GaAs). Since ρ_{2deg} has units of electrons $m^{-2}J^{-1}$, one needs first to convert to $nm^{-2}V^{-1}$ by multiplying it by $10^{-18}e$, then by -1 because electrons are negatively charged, then divide by a factor $2nm$ (height of the 2DEG) because ρ_H stands for the total charge in the cell per volt, not the density.

The last three lines of Script 4.17 finally call the **solve()** method of the problem, c.f. Eq.(4.23), with the inputs that we have prepared. It returns the solution (voltages and charges) of the electrostatic problem for the device on Fig.4.12. The values of ρ_H, U_D are set respectively by the parameter **voltage** and **helmholtz_density**. Q_N and Q_H are set either by the (optional) parameters **charge** (to set the total

charge in the cell) or **charge_density** (in that case *PESCADO* multiplies the entry by the volume of the cell). If these parameters are not given, the default density is zero. Both **charge** and **charge_density** can be used simultaneously as long as the same site is not set twice.

The **solve()** method returns two outputs, *voltage_res* and *charge_res*. They are both instances of **SparseVectors**. The first is the voltage value for all mesh points - including dirichlet points. The second is the charge value for all mesh points - including Neumann and Helmholtz points. Their units follow Table.4.1. In Section 4.4.5 we explain how to further use and plot the output data.

```
1
2  from scipy import constants
3  from pescado.poisson import Problem
4
5  # Load the \PESCADO Problem
6  pd = Problem.load('elec_problem_tutorial')
7
8  # Define the U_d sparse vector
9  u_d = pd.sparse_vector(val=-.5, name='gate')
10
11 ## Define the Q_N sparse vector
12 # Using a function
13 def dop(r):
14     return (0.33e-4
15             + 0.33e-4 * np.exp(-(np.abs(r[:, 0]) / 400) ** 2))
16 q_n = pd.sparse_vector(val=dop, name='dopants')
17
18 # As a constant concentration
19 q_n = pd.sparse_vector(val=0.33e-4, name='dopants')
20
21 # Define the helmholtz density sparse vector
22 gas_idx = pd.points(name='2deg')
23 dens_si = (2*0.067 * constants.m_e) / (np.pi * constants.hbar ** 2)
24 dens_helm = (
25     -dens_si * 1e-18 * constants.elementary_charge
26     * pd.volume[gas_idx] / 2)
27
28 helmholtz_dens = SparseVector(indices=gas_idx, values=dens_helm)
29
30 # Solve
31 voltage_res, charge_res = pd.solve(
32     voltage=u_d, charge_density=q_n,
33     helmholtz_density=helmholtz_dens)
```

Script 4.17: This script starts from a **Problem** that has been saved into a file. Here we load it and solve the electrostatic problem for the device shown in Fig.4.12. The **Problem** was created in Script 4.14.

4.4.5 Reading and Plotting

The **solve()** method in **Problem** returns the voltage and the charge in the whole system (two **SparseVector**). In this section we discuss how to manipulate these vectors to extract the information. In particular, we show how to recover the voltage and charge for specific regions of the system, manipulate the corresponding values and plot them.

The most straightforward way to access the value of a **SparseVector** is to directly request them using the *vector[index]* syntax (similar to e.g. a numpy array). For instance if we want to extract the charge in the 2DEG region, we first obtain the list of indices using the method **Problem.points_inside()** or **Problem.points()**. In a second step we get the values of the charge (a numpy array) using *charge[indices]*. See Script 4.16 and the associated description of sparse vectors.

PESCADO implements a convenient function to do the above in a more transparent way. The **read_sparse_vector()** method of **Problem** takes a **SparseVector** and returns the values of its elements located inside a given region of the **Problem** (in the form of a numpy array). There are two ways to define the region, first (*region* parameter) using a **Shape** or second (*name* parameter) using the name it was attributed when **Problem** as made, e.g. at line 32 of Script 4.14. Setting the *return_indices parameter* to True will also return the associated list of indices. See Script 4.18 for an example. Script 4.18 also shows how one can use **SparseVector** to manipulate the data (here summing sparsevectors and multiplying by a float). In this example, we calculate the total electronic density in the cells in S.I. units $n = (\rho V + Q)/\Omega$ where Ω_i is the volume of the cell.

```
# Extract the values and indices inside a region
gas_charge, gas_idx = pd.read_sparse_vector(
    charge_res, name='2deg', return_indices=True)

# Define the total charge sparse vector
total_charge = SparseVector(
    indices=charge_res.indices,
    values=charge_res.values, default=0)

total_charge[gas_idx] += (
    helmholtz_dens[gas_idx] * voltage_res[gas_idx])

total_charge[total_charge.indices] = (
    total_charge[total_charge.indices]
    / pd.volume[total_charge.indices])
total_charge *= 1e18
```

Script 4.18: Working with *PESCADO* data output.

Let us now discuss how to plot the objects calculated in *PESCADO* . The difficulty lies in that *PESCADO* uses an unstructured mesh. To circumvent this, *PESCADO* implements an interpolation scheme that allows one to evaluate the charge/voltages on a uniform grid. For each point $\vec{u}$ on the uniform grid, *PESCADO* looks for the closest point $\vec{r}$ in the **SparseVector** and attributes the corresponding value to $\vec{u}$ (in practice this is implemented internally using an efficient KDTree - from scipy.spatial.KDTree).

The primary method for plotting is **Problem.plot_sparse_vector()**. It is designed to produce 1D or 2D interpolated slices of the data along the x, y or z direction (1D) or in the xy, xz, yz plane (2D). This method takes four parameters as input: *sparse_vector* (the data, an instance of **SparseVector**), *region* (the region to plot, an instance of **Shape**), *grid_step* (distance between points in the uniform

grid where the interpolation will be performed, a float) and optionally *direction* (the dimension ignored when the plot has a smaller dimension than the initial problem, it is an integer or sequence of integers). The interpolation data generated is either 1D or 2D while the electrostatic problem is typically of higher dimension. The *direction* parameter tells which dimension (or dimensions) are frozen when taking the slice. It takes the value 0, 1 and 2 respectively for the x, y and z axis. The *cut_position* variable (see below) then sets the value where the cut is performed. For instance, suppose that one has a 3D simulation and wants to plot the potential in the xz plane (2D slice) for a fixed value $y = 3.5$. Then we have direction=1 and cut_position=3.5. For a 1D plot along y at fixed value $x = 0, z = 1.3$, one would set direction=(0,2) and cut_position=(0,1.3). The **Problem.plot_sparse_vector** does not plot the figure immediately, it returns instead a function. Calling this function actually makes the plot. The first three input parameters are *cut_position* (floats or sequence of floats), *ax* (instance of **matplotlib.pyplot.gca()**) and *cmap* (string). One can also add more parameters. They will be sent directly to either **matplotlib.imshow()** (if 2D section) or **ax.plot()** (if 1D cut).

Script 4.19 and Script 4.20 illustrate how to use **plot_sparse_vector()** in concrete examples. The Script 4.19 plots the results *total_charge* and *voltage_res* from respectively Script 4.18 and Script 4.17. Line 7 defines the 2D *region* parameter, lines 9 to 11 generates the plotting function and line 14 plots the 2D voltage profile shown on the bottom of Fig. 4.12. Line 18 defines the 1D *region* parameters, lines 20 to 22 generates the plotting function and line 29 plots the 1D charge cut along $\vec{x}$ for $y = 0$ shown on the left of Fig. 4.14. The line 31 plots the 1D voltage cut along $\vec{x}$ for $y = 0$ shown on the right of Fig. 4.14. The Script 4.20 plots the *qcharge_btf* results obtained from the bulk thomas fermi calculations done with Script 4.4. Lines 7 to 22 plots a 2D $x - y$ slice for $z = 0$ and lines 24 to 31 a 1D cut along $\vec{x}$ for $z = 0$ and $y = 0$. Line 12 defines the 2D *region*, line 15 to 17 make the plotting function and line 19 actually plots the 2D section of the charge shown on the top right of Fig. 4.3. Lines 24 defines the 1D *region*, lines 25 to 27 make the plotting function and finally line 29 plots the 1D cut along $\vec{z}$ of the charge shown on the bottom right of Fig. 4.3.

The function **ProblemBuilder.plot_boundary_condition()** (resp. **ProblemBuilder.plot_system_regions()**) that was used in Section 4.4.3 to plot the Dirichlet / Neumann / Helmholtz/Flexible partitioning of the system (resp. the different regions) uses the same interpolation scheme as **plot_sparse_vector()**.

```
1
2  from pescado.mesher import patterns
3  from pescado.mesher import shapes
4  import numpy as np
5  import matplotlib.pyplot as plt
6
7  plot_region = shapes.Box(lower_left=[-300, -3], size=[600, 293])
8
9  voltage_section = pd.plot_sparse_vector(
10     sparse_vector=voltage_res,
11     region=plot_region, grid_step=5)
12
```

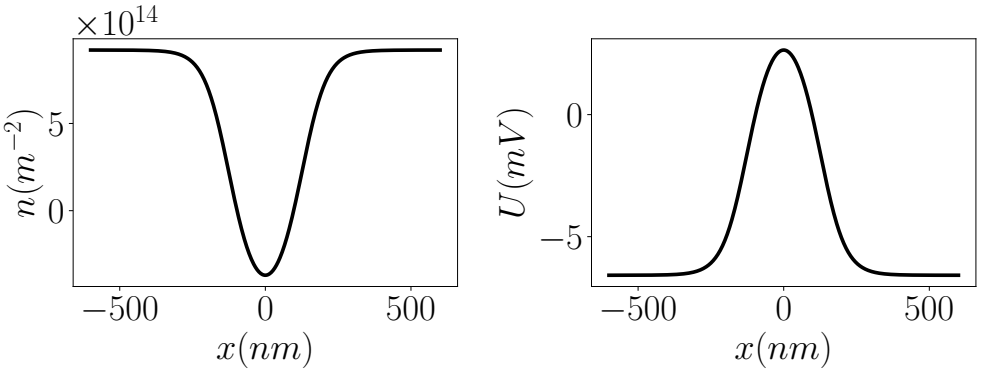


Figure 4.14: Charge (left) and potential (right) profile along $\vec{x}$ at the 2DEG for Fig.4.12. Obtained with Script 4.19.

```
13 fig, ax = plt.subplots()
14 ax, img = voltage_section(ax=ax)
15 fig.colorbar(img, ax=ax)
16 plt.show()
17
18 plot_region = shapes.Box(lower_left=[-600, ], size=[1200, ])
19
20 total_charge_cut = pd.plot_sparse_vector(
21     sparse_vector=total_charge,
22     region=plot_region, grid_step=2, direction=1)
23
24 voltage_cut = pd.plot_sparse_vector(
25     sparse_vector=voltage_res * 1e3, # to mV
26     region=plot_region, grid_step=2, direction=1)
27
28 fig, ax = plt.subplots(1, 2, figsize=(12, 5))
29 total_charge_cut(cut_position=0., ax=ax[0], c='k', linewidth=3)
30 voltage_cut(cut_position=0, ax=ax[1], c='k', linewidth=3)
31 plt.show()
```

Script 4.19: How to use **Problem.plot_sparse_vector()** to reproduce the voltage 2D profile on the bottom of Fig. 4.12 (Lines 7 to 16) and the charge and voltage 1D cut in Fig. 4.14 (Lines 18 to 31). The charge and voltage it plots are obtained from respectivelly Script 4.18 and Script 4.17.

```
1
2 from pescado.mesher import patterns
3 from pescado.mesher import shapes
4 import numpy as np
5 import matplotlib.pyplot as plt
6
7 qcharge_btf = qcharge_btf * (1e18 / 100) # to SI
8 qcharge_btf.default = np.nan
9
10 fig, ax = plt.subplots()
11
12 plot_region = shapes.Box(
13   lower_left=[-500, -500], size=[1000, 1000])
```

```
14
15 total_charge_section = pd.plot_sparse_vector(
16   qcharge_btf, region=plot_region,
17   grid_step=10, direction=2)
18
19 ax, img = total_charge_section(cut_position=0, ax=ax)
20 fig.colorbar(img, ax=ax)
21
22 plt.show()
23
24 plot_region = shapes.Box(lower_left=[-150, ], size=[450, ])
25 total_charge_cut = pd.plot_sparse_vector(
26   qcharge_btf, region=plot_region,
27   grid_step=10, direction=(1, 2))
28
29 total_charge_cut(cut_position=(0, 0), c='k', linewidth=3)
30
31 plt.show()
```

Script 4.20: How to use **Problem.plot_sparse_vector()** to reproduce the charge 2D profile on the top right of Fig. 4.3 (Lines 7 to 22) and the charge 1D cut on the bottom right of Fig. 4.3 (Lines 24 to 32). The charge it plots are obtained with Script 4.4.

4.4.6 Solving a discrete electrostatic problem with flexible sites

From a physics point of view, *PESCADO* implements only three types of sites: $\mathcal{N}$, $\mathcal{D}$ and $\mathcal{H}$. Nothing else is needed. However, *in practice* there are many iterative algorithms where one site is, say, $\mathcal{H}$ in one iteration and becomes $\mathcal{N}$ in the next. For instance in a self-consistent quantum-electrostatic calculation, the sites of the "active" region (e.g. the 2DEG) are generally $\mathcal{H}$. If during the calculation one finds that some of these sites are depleted then they become $\mathcal{N}$ (the inverse is also possible, the iterative algorithm may have wrongly depleted a site). To deal with this situation in a way that introduces the minimum computational overhead, *PESCADO* implement a fourth site type, flexible sites $\mathcal{F}$. When a site is set as $\mathcal{F}$ during the construction of the **Problem**, *PESCADO* pre-calculates the capacitance matrix elements in such a way that one can easily switch these sites between $\mathcal{N}$, $\mathcal{D}$ and $\mathcal{H}$ at the solving stage. Hence a $\mathcal{F}$ site is simply one whose status is undecided at the construction stage. It must be set before solving. Flexible sites are just an optimization. It is particularly convenient in the common situation where there is only a small fraction of $\mathcal{F}$ sites (e.g. in a 2DEG where the electron gas is only a small fraction of the entire stack).

To set a site as flexible during the construction of the problem, one must use the **set_flexible()** method of **ProblemBuilder**. Then, to partition the flexible sites into the $\mathcal{N}$, $\mathcal{D}$ and $\mathcal{H}$ sets, one must use the **assign_flexible()** method of **Problem**. Once all flexible sites have been assigned, one uses the **freeze()** method to update Eq.(4.23). One then proceeds with solving the problem as explained in Section 4.4.4.

Script 4.21 shows how to modify Script 4.14 such that the sites in the 2DEG region become flexible sites. First, one must replace line 32 of Script 4.14 with

line 4 of Script 4.21. Second, one must set the flexible sites before calling the **Problem.solve()** method. This is done by adding the lines 9-14 of Script 4.21. In this example the first 50 sites of the 2DEG will be $\mathcal{N}$ and the remaining sites will be $\mathcal{H}$.

```
1  [...]
2
3  # Replace line 38 of Script 10
4  pb.set_flexible(region=twodeg, setup_name='2deg')
5
6  [...]
7
8  # Add before calling pd.solve()
9  gas_idx = pd.points(name='2deg')
10 pd.assign_flexible(
11   neumann_index=gas_idx[:50],
12   helmholtz_index=gas_idx[50:])
13
14 pd.freeze()
15
16 [...]
```

Script 4.21: Modifications of Script 4.14 and Script 4.17 to introduce flexible indices.

4.5 The Non-Linear Helmholtz solver

We have covered how *PESCADO* solves linear electrostatic problems. We now turn to the *PESCADO* NLH solver. The NLH solver can be used by itself (in that case it yields a solution of the generalized Thomas-Fermi approximation problem) or in conjunction with a quantum solver to perform fully self-consistent quantum-electrostatic calculations, see Chapter 3. *PESCADO* NLH solver is itself constructed on top of the LH equation solver we have seen previously.

The NLH problem is defined by the following equations:

$$\nabla \cdot (\varepsilon(\vec{r})\nabla U(\vec{r})) = \mp e \int_{-\infty}^{\mu(\vec{r})} dE \rho(\vec{r}, E) \tag{4.24}$$

$$\mu(\vec{r}) \pm eU(\vec{r}) = E_F. \tag{4.25}$$

It describes one (or more) quantum region with electro-chemical potential E_F (Fermi level) containing electrons (lower sign) or holes (upper sign). The LDOS per volume and per energy, $\rho(\vec{r}, E)$, is considered to be known. The right hand side of Eq.(4.24) is the number of charges per unit volume and is known as the ILDOS $n(\vec{r}, \mu)$. It reads,

$$n(\vec{r}, \mu) = \int_{-\infty}^{\mu(\vec{r})} dE \rho(\vec{r}, E) \tag{4.26}$$

These definitions are valid at zero temperature. At finite temperature or out-of-equilibrium, one simply needs to adapt the definition of $n(\vec{r}, \mu)$ and its derivative. As before $\varepsilon(\vec{r})$ is the dielectric constant and $U(\vec{r})$ is the electric potential. Eq.(4.25) decomposes the electro-chemical potential as the sum of its electric $\pm eU(\vec{r})$ and

chemical $\mu(\vec{r})$ contributions. After discretization of the problem one arrives at the site- ILDOS and site-LDOS,

$$Q_i(\mu) = \int_{\mathcal{C}_i} d\vec{r}\; n(\vec{r},\mu) \tag{4.27}$$

$$\rho_i(\mu) = \int_{\mathcal{C}_i} d\vec{r}\; \rho(\vec{r},\mu) \tag{4.28}$$

The input of the NLH problem is the (site variant of the) ILDOS $Q_i(\mu)$ and its derivative the LDOS $\rho_i(\mu) > 0$ (which importantly is always positive). There are different ways to arrive at a NLH problem. One could want to solve the electrostatic problem self-consistently with quantum mechanics. The standard approach for this is to calculate the density of electrons for a given potential using quantum mechanics. Then one solves the electrostatic problem to obtain the potential from this density and iterates this loop until convergence to the self-consistent equation. In this approach the self-consistency is obtained on the electronic density itself. The *PESCADO* approach to the self-consistent problem is different: one computes the full LDOS from quantum mechanics not just the density, then one solves the NLH problem and obtains the potential. The LDOS is recalculated and one continues until self-consistency is achieved. We see that much more information is exchanged between the quantum and electrostatic solver: the electrostatic solver knows not only what is the density but also the density *of states*, i.e. how the density changes when the potential is modified (screening). The corresponding self-consistent loop converges in a handful of iterations, very robustly. Very often a full self-consistent quantum-electrostatic solution is not needed and one can obtain accurate results at a cruder approximation level. A common one is the generalized Thomas-Fermi approximation: instead of calculating the LDOS, one uses the bulk density of states $\rho(E)$ (DOS) which is known for each material: $\rho(\vec{r},E) \approx \rho(E)$. Note that we use the same letter ρ for the LDOS, the DOS (its bulk value) and the density of states $\rho(\vec{r})$ of the LH solver [implicitly $\rho(\vec{r})$ is the LDOS at a given energy, for instance $\rho(\vec{r}) = \rho(\vec{r},E_F)$]. We use the arguments to determine which object is used. When $\rho(\vec{r},E)$ does not depend on energy, the NLH problem reduces to the linear one.

PESCADO has two different algorithms for solving the NLH problem: the piecewise Newton-Raphson algorithm and the piecewise dichotomy algorithm. The piecewise Newton-Raphson algorithm is fast and robust, it should be preferred as a default. The piecewise dichotomy algorithm is a bit slower but more robust, it should be preferred in the rare cases where piecewise Newton-Raphson algorithm fails to converge. Both explicitly handle the discontinuities of the LDOS. These discontinuities are always present at the band edges and are the main source of non-linearity in the problem (in particular when one partially depletes the system with electrostatic gates). A detailed description of these algorithms is given in Chapter 3.

PESCADO has two possible representations of the ILDOS: the general **self_consistent.ildos.PContinuousIldos** class implements arbitrary functions $Q_i(\mu)$. The second representation is the **self_consistent.ildos.PLinearIldos** class

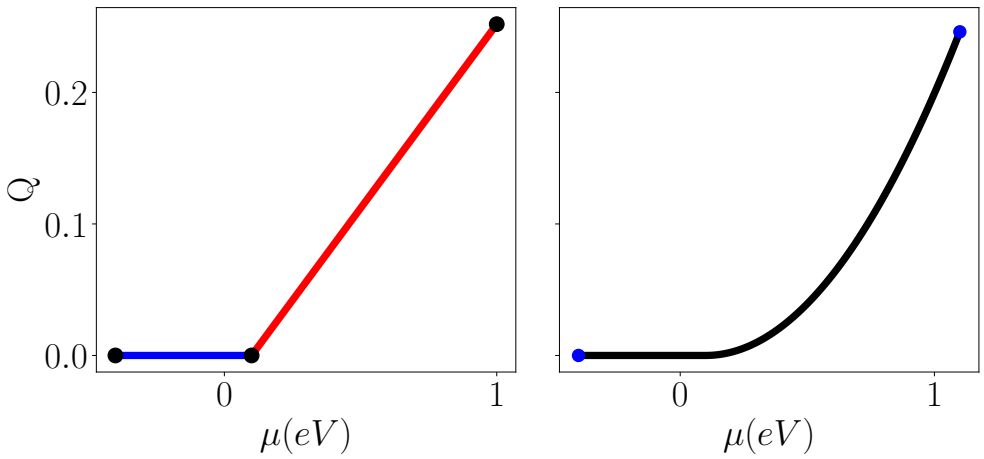


Figure 4.15: On the left a piecewise linear Ildos, defined with **PLinearIldos**. In this simple case, one uses just two intervals, blue and red. The black dots correspond to the intervals bounds. The figure on the right shows a piecewise continuous ILDOS $n(\mu) \propto \theta(\mu)\mu^2$, defined with **PContinuousIldos**.

which is restricted to ILDOS that are piecewise linear (see Fig. 4.15 for examples). This latter class can be used directly in simple cases where the ILDOS is actually piecewise linear. It is also generated internally by the piecewise dichotomy algorithm.

The two solvers are implemented in the **self_consistent.solver.PieceWiseDichotomy** and **self_consistent.solver.PieceWiseNewtonRaphson** class. The **thomas_fermi** constructor is a thin wrapper that returns an instance of one of the two classes. In the next subsections we first explain how to define the problem, i.e. set up an instance of **PContinuousIldos** or **PLinearIldos**. Then we proceed with solving Eq.(4.24). Finally, we show how to customize the self consistent algorithm implemented in *PESCADO* .

4.5.1 Defining an ILDOS

Let us illustrate how to define the two ILDOS functions shown in Fig.4.15. For the function on the left we shall use a **PLinearIldos** instance. Regarding the one on the right, we shall use an instance of **PContinuousIldos**. Finally, we will also show how to discretize the function on the right using a series of linear intervals.

4.5.1.1 API of PLinearIldos

We start with the ILDOS on the left side of Fig.4.15. It corresponds to the bulk DOS of a 2DEG : $\rho(E) = 0.067 m_e/\pi\hbar^2$ (in SI units) for $E > E_b$ where $E_b = 0.1eV$ is the band edge (red interval) and $\rho(E) = 0$ for $E \leq E_b$ (blue interval). To define

the corresponding **PLinearIldos** instance, one needs to provide a series of values (E_α, Q_α) and *PESCADO* interpolates linearly between them. By convention, the first segment is continued al the way to $-\infty$ and the last segment all the way to $+\infty$ so that the energy value of the first (last) point is irrelevant as long as it is chosen to be smaller (larger) than E_b (here we chose $E_0 = -0.4eV$, $E_1 = E_b = 0.1eV$ and $E_2 = 1eV$).

Script 4.22 shows the construction of the ILDOS. The DOS is defined in Si units on line 6 ($m^{-2}.J^{-1}$). We transform it to *PESCADO* units on line 7 ($nm^{-2}.eV^{-1}$). The next step (lines 9-12) is to define a numpy array *bounds* that contains the values (E_α, Q_α) (bounds$[\alpha, 0] = E_\alpha$ and bounds$[\alpha, 1] = Q_\alpha$). Last, in line 20 we initialize the ILDOS using the **PLinearIldos.from_array** method that takes *bounds* as the coordinates parameter.

There is also a second method to initialize an instance of **PLinearIldos**. One simply calls the class directly with three parameters *coordinates*, *dos* and *origin* (this bypasses the **from_array()** method). The *coordinates* argument is the same as that for **from_array()**. The *dos* and *origin* are the, respectively, DOS and origin for each interval defined by *coordinates*. Lines 15 to 17 define the *ildos_pl* instance.

Once the ILDOS has been constructed, it provides methods to easily use it. Calling it directly with a numpy array (a list of energies E) returns a tuple of numpy arrays with the DOS and the ILDOS (line 23). Calling it with a float E returns a tuple (DOS,ILDOS) together with the index of the interval to which E belongs if *return_intervals* is set to True (line 24). The *interval(E)* just returns this index (line 25). Last, one may also directly access the list of dos and origin in the different intervals (line 26). In each interval a, one has $Q(\mu)$ =dos[a]*μ + origin[a].

```
1  import numpy as np
2  from scipy import constants
3  from pescado.self_consistent import ildos
4
5  # Define the ILDOS intervals bounds
6  dos = (0.067 * constants.m_e) / (np.pi * constants.hbar ** 2)
7  dens_helm = dos * 1e-18 * constants.elementary_charge
8
9  E_b = 0.1
10 bounds = np.empty((3, 2), dtype=float)
11 bounds[:, 0] = np.array([-0.4, E_b, 1])
12 bounds[:, 1] = np.array([0, 0, dens_helm * (1 - E_b)])
13
14 # Define the piecewise linear ILDOS
15 ildos_pl = ildos.PLinearIldos(
16     coordinates=bounds, dos=np.array([0, dens_helm]),
17     origin=np.array([0, - E_b * dens_helm]))
18
19 # Define a piecewise linear Ildos using from_array
20 ildos_plfa = ildos.PLinearIldos.from_array(coordinates=bounds)
21
22 # Recover the dos, q for a given set of u
23 dos, q = ildos_plfa(np.linspace(-0.2, 0.5, 50))
24 dos, q, interval = ildos_plfa(u=0.2, return_intervals=True)
25 interval = ildos_plfa.interval(u=0.2)
```

```
26 dos, origin = ildos_plfa.dos, ildos_plfa.origin
```

Script 4.22: Define the piecewise continuous ILDOS shown in Fig.4.15

4.5.1.2 API of PContinuousIldos

We now consider the continuous ILDOS shown on the right side of Fig.4.15 (in black). To define it in *PESCADO* , we must use an instance of **PContinuousIldos**. The two arguments used to initialize an instance of **PContinuousIldos** are *coordinates* and *functions*. The *functions* argument is a list of python functions. Each function takes as input either a float or array of chemical potential and return their density of states and ILDOS value (tuple of two floats or two arrays). There is no limitation to the number of elements in the *functions* list. Lines 10 to 25 of Script 4.23 defines the *ildos_ quadratic* function. It defines the black ILDOS on the right side of Fig.4.15. The *coordinates* parameter behaves in the same way as for **PLinearIldos**. It must have n+1 elements, with n the number of elements in *functions*. Lines 28 to 30 define the bounds for *ildos_ quadratic*. Line 32 of Script 4.23 initializes an instance of **PContinuousIldos**, *ildos_ cont*. It is possible to recover the density of states, ILDOS and interval value for a given chemical potential. To do so we refer to lines 23 to 25 of Script 4.22.

Finally, it is possible to discretize an instance of **PContinuousIldos** into a set of linear intervals. Each linear interval is defined around a chemical potential value, which we refer to as functional point. The linear interval is the tangent touching the ILDOS at its functional point. The bounds of the interval is defined by the intersection between the current interval tangent and that of its neighboring intervals. In case no intersection is found within its neighboring functional points, *PESCADO* adds either a vertical or horizontal interval in between them. This ensures the discretized ILDOS is monotonically continuous.

In practice, to discretize a **PContinuousIldos** instance we call the **linearize()** method. It discretizes the piecewise continuous ILDOS around a set of functional points. Calling this method returns an instance of **PLinearIldos**. Line 36 and 37 of Script 4.23 call **linearize()** with five functional points. The discretized ILDOS is shown by the blue dotted line on the right of Fig.4.15. It is also possible to iteratively add intervals to the discretized ILDOS even after it has been initialized. To do so use the **add_interval** method. It takes as parameter μ, *dos*, *origin* and q. The q parameter is optional, it is only required if *dos* is infinity (vertical interval) or None. Lines 40 to 43 update *ildos_ discretized* for four different μ values. The resulting linear approximation is shown as the red dotted line on the right side of Fig.4.15. The green dots are the functional points. The left ILDOS of Fig.4.15 has only seven functional points (green), instead of the nine added in lines 35 and 39-42 of Script 4.23. This is because the discretization algorithm we implemented in *PESCADO* removes functional points that are redundant. Redundant points are those whose LDOS and origin are the same as those of at least one of its neighboring functional points.

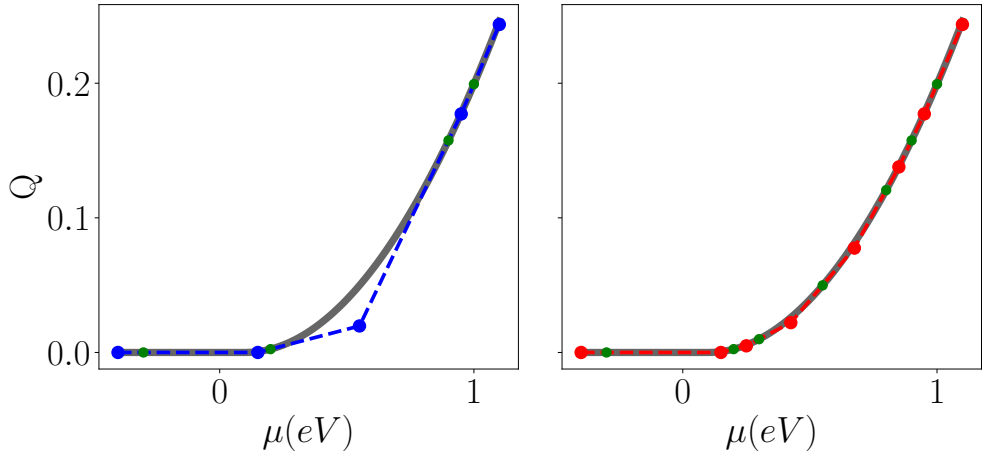


Figure 4.16: Quadratic ILDOS (faded black) of Fig.4.15 after discretization into linear intervals. On the left the ILDOS is discretized around four functional points (green dots). The four linear intervals are marked as the dotted blue lines, the blue points are their bounds. On the right the discretization on the left is refined with four additional functional points. The linear intervals and their bounds are shown in red.

```

from collections.abc import Sequence
import numpy as np
from scipy import constants
from pescado.self_consistent import ildos

# Quadratic ildos
E_b = 0.1
def ildos_quadratic(mu):

    t = constants.m_e * 0.067 * 1e-18 / constants.hbar ** 2
    dens = t ** 2 * (constants.elementary_charge ** 2) / np.pi

    if isinstance(mu, (Sequence, np.ndarray)):
        ildos_ = dens * ((mu - E_b) ** 2)
        dens_ = 2 * np.ones(len(mu)) * dens * (mu - E_b)
        ildos_[mu <= E_b], dens_[mu <= E_b] = 0, 0
    else:
        dens_, ildos_ =  0, 0
        if mu > E_b:
            dens_ = 2 * dens * (mu - E_b)
            ildos_ = dens * ((mu - E_b) ** 2)

    return dens_, ildos_

# Define the two intervals
bounds = np.zeros((2, 2))
bounds[[0,1], 0] = np.array([-0.4, 1.1])
bounds[[0,1], 1] = ildos_quadratic(mu=bounds[[0,1], 0])[1]

ildos_cont = ildos.PContinuousIldos(
```

```
32      coordinates=bounds, functions=[ildos_quadratic, ])
33
34  # First linearize around fpoints
35  fpoints = np.array([-0.3, 0.06, .2,.9, 1.0])
36  ildos_discretized = ildos_cont.linearize(fpoints=fpoints)
37
38  # Add additional function points
39  for fp in [-0.05, .3, .55, .8]:
40      dos, q = ildos_cont(fp)
41      ildos_discretized.add_interval(
42          u=fp, dos=dos, origin=q - dos * fp)
```

Script 4.23: Define the piecewise continuous ILDOS shown in Fig.4.15

4.5.2 Solving the NLH equation

Once we have defined the ILDOS for our problem, we must construct the electrostatic problem as before. A selection of the sites - hereafter referred to as the *quantum sites* will be attributed an ILDOS. All the quantum sites must be flexible sites, the other sites can be of any type. We can then proceed with calling the solver. The **thomas_fermi** constructor returns an instance of one of the two solvers **self_consistent.solver.PieceWiseNewthonRaphson** or **self_consistent.solver.PieceWiseDichotomy** (by default the first one). The function takes five parameters (the last three being optional). The first parameter *poisson_problem* is the Poisson problem, the construction of which has been discussed in the preceding sections. The second *ildos* is the ILDOS. If a single instance of **PContinuousIldos** or **PLinearIldos** is given, then the same ILDOS is used on all the quantum sites. *ildos* can also be a list of ILDOS when the ILDOS is *e.g.* spatially dependent. In that case, one needs to set the input *sites_ildos*, an instance of **SparseVector**. For each site present in *sites_ildos*, the corresponding value of the sparsevector is the index of the *ildos* (*i.e.* the ILDOS of site i is *ildos[sites_ildos[i]]*). The last two parameters are the Fermi level E_F (*e_f*) which is 0.0 by default and *method* that decides on the solver (a string, either **PieceWiseNewthonRaphson** or **PieceWiseDichotomy**). Let us go through a practical calculation for the 2D model shown in the top sketch of Fig.4.1. We will treat the 2DEG at the Thomas-Fermi level, a much better approximation than what was done in Script 4.1 (where the 2DEG was described by an equipotential). First, let us construct the Poisson problem. We will use the simple Script 4.1 and only need to modify two lines:

- in Line 29, we replace *pb.set_dirichlet(region=twodeg, setup_name='2deg')* with *pb.set_flexible(region=twodeg, setup_name='2deg')* since the quantum sites will correspond to the sites of the 2DEG.
- change the last line to a different name *pd.save('Problem_tf_example_2D')* since it is a different electrostatic problem.

Script 4.24 shows the API of the solver. In Line 7, one loads the **Problem**. Line 10 constructs the solver. Here we use the piecewise linear ILDOS shown on

the left of Fig.4.15 and constructed at the beginning of this section (see Script 4.22). The same ILDOS is used in all quantum sites (a single ILDOS is given, not a list). Since this ILDOS is the one of a bulk 2DEG, this is effectively the Thomas-Fermi approximation. The actual calculation is done when one calls the **solve()** method. It takes two input parameters, *poisson_input* and *initial_guess*. The *poisson_input* is a dictionary defining the usual inputs that need to be given to the poisson problem for all the sites that are not a quantum site. For this specific example, it defines the values of the gate voltage and the doping charge density. The keys of this dictionary are the names of the input parameters passed down to the **Problem.solve()** method. The *initial_guess* is a **SparseVector**. It plays a double role. First, the sites that are present in this sparsevector will be defined as the quantum sites. The self-consistency only applies to these sites. It may change from one call to **solve()** to another. For instance if some sites are depleted for a given gate voltage, one may set them to Neumann and remove them from the quantum sites to save computing time. Second, it sets the initial guess for the chemical potential on the quantum sites. It is not important to have a good initial guess but it can speed up the convergence (using e.g. the result of a previous calculation at a slightly different gate voltage). Line 17 to 19 define the *initial_guess* setting $\mu = 0$ at all quantum sites. It happens that for this example all flexible sites are quantum sites, hence line 19.

Line 21 to 23 call *tf_plinear.solve()*. To recover the physical results there are five available methods. They all take as input *iteration*, an integer specifying the iteration number in the self-consistent calculation (iteration=-1 for the last one). They all return an instance of **SparseVector**.

- **charge(*iteration*)** returns the charge Q_i for all sites in the poisson problem. Note that for the quantum sites, this charge does not include the contribution from the chemical potential (see below for the correct method);
- **potential(*iteration*)** returns the potential U_i for all sites in the poisson problem (including the quantum sites);
- **quantum_charge(*iteration*)** returns the full charge Q_i for the quantum sites only;
- **chemical_potential(*iteration*)** returns the chemical potential μ_i for the quantum sites only;
- **intervals(*iteration*)** returns the index of the ILDOS interval to which the solution belongs, for the quantum sites.

Lines 25 and 26 of Script 4.24 recovers respectively the charge and chemical potential at the quantum sites at convergence. Figure 4.17 shows the charge (black) and chemical potential (green) recovered in lines 25 and 26 respectively.

```
1
2 from pescado.poisson import Problem
```

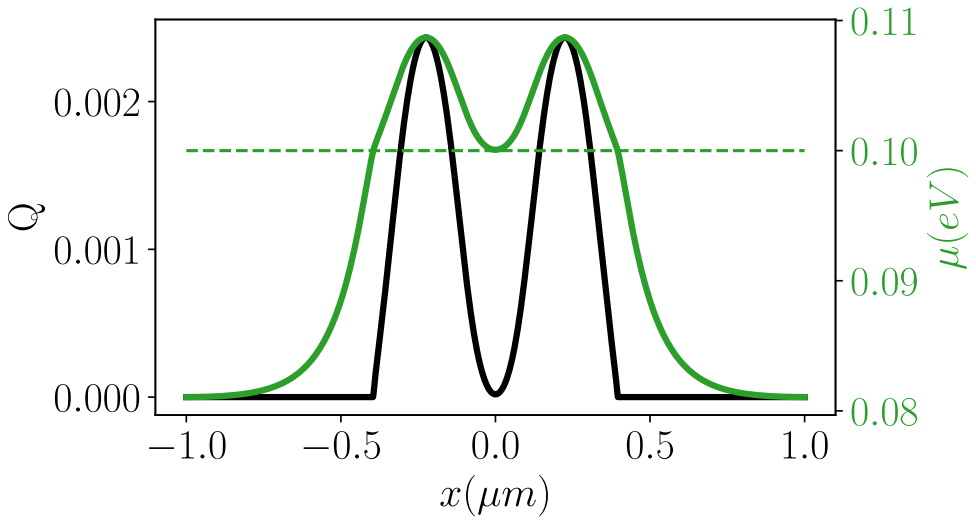


Figure 4.17: In black and green respectively the charge and chemical potential thomas fermi solution. Obtained with the piecewise Newton Raphson algorithm with Script 4.24. The dashed green line is at the fermi energy $E_F = 0.1$

```
3  from pescado.self_consistent import solver
4  from pescado.self_consistent import ildos
5  from pescado.tools import SparseVector
6
7  pd = Problem.load('Problem_tf_example_2D')
8
9  e_f = 0.1
10 tf_plinear = solver.thomas_fermi(
11     ildos=ildos_plfa, poisson_problem=pd, e_f=e_f)
12
13 sv_gate = pd.sparse_vector(val=-0.2, name='gate')
14 sv_dop = pd.sparse_vector(val=1e-4, name='dopants')
15 poisson_input = {'voltage':sv_gate, 'charge_density':sv_dop}
16
17 initial_guess = SparseVector(
18     values=np.zeros(len(pd.flexible_indices)),
19     indices=pd.flexible_indices, dtype=float)
20
21 tf_plinear.solve(
22     poisson_input=poisson_input,
23     initial_guess=initial_guess)
24
25 charge_tf_plinear = tf_plinear.quantum_charge(iteration=-1)
26 cp_tf_plinear = tf_plinear.chemical_potential(iteration=-1)
```

Script 4.24: Define the piecewise continuous ILDOS shown in Fig.4.15

Let us now consider a slightly different calculation: we replace the ILDOS with the continuous ILDOS shown on the right of Fig.4.15. Script 4.25 shows the code for this calculation. We also use a different solver, the **PieceWiseDichotomy**, as shown in Line 13. Line 15 calls the **solve()** method of *tf_pd*. In the piece-

wise dichotomy method, the continuum ILDOS is approximated by a piecewise linear ILDOS that matches the continuum one (both for the ILDOS and its derivative the LDOS) on certain points (the so-called functional points or fpoints). The list of fpoints is constantly updated with new points (close to the solution) until convergence. The piecewise dichotomy method, requires an additional input - *initial_fpoints*. It is a numpy array of floats defining the initial values of chemical potential used in the discretization of the continuous ILDOS. For each quantum site this solver creates an instance of **PLinearIldos** that matches the input ILDOS on the *initial_fpoints*. More points are added to this **PLinearIldos** until convergence (see Figure 3.6). Figure 4.18 shows the charge (black) and chemical potential (green) recovered in lines 20 and 21 respectively.

```
1
2  from pescado.poisson import Problem
3  from pescado.self_consistent import solver
4  from pescado.self_consistent import ildos
5  from pescado.tools import SparseVector
6
7  pd = Problem.load('Problem_tf_example_2D')
8
9  e_f = 0.1
10
11 tf_pd = solver.thomas_fermi(
12     ildos=ildos_cont, poisson_problem=pd,
13     e_f=e_f, method='PieceWiseDichotomy')
14
15 tf_pd.solve(
16     poisson_input=poisson_input,
17     initial_guess=initial_guess,
18     initial_fpoints=np.array([-1e-4, 0.3]))
19
20 charge_tf_pd = tf_pd.quantum_charge(-1)
21 cp_tf_pd = tf_pd.chemical_potential(-1)
```

Script 4.25: Define the piecewise continuous ILDOS shown in Fig.4.15

Script 4.26 shows a third variation on the same problem. This time, we want to use different ILDOS on different sites. This illustrates a situation where *e.g.* one would like to use a gross approximation for the ILDOS in a not so interesting part of the sample (here the discrete bulk ILDOS for $|x| \geq 700nm$) and a more accurate one for the interesting region (for instance recalculated using a quantum solver), here we use the continuum ILDOS that we have defined for $300nm \leq |x| \geq 700nm$. We assume that all other sites are depleted and set them to Neumann condition.

Line 11, 14 and 16 construct the corresponding list of sites using standard numpy tools. In line 20, we construct the list of ildos, here there are only 2 of them. There can be up to one ILDOS per quantum site in the case where one is interested in coupling this solver with a quantum solver that calculates the ildos. Line 21 constructs the *sites_ildos* which takes value 0 for the sites associated to the discrete ILDOS and one for the sites associated with the continuum one. Line 29 constructs the initial guess and sets which sites are the quantum sites (here not all the flexible sites will belong to this set since we're setting some to be Neumann). Last, in line 33, we need to add the information about the flexible sites that will be set to Neumann

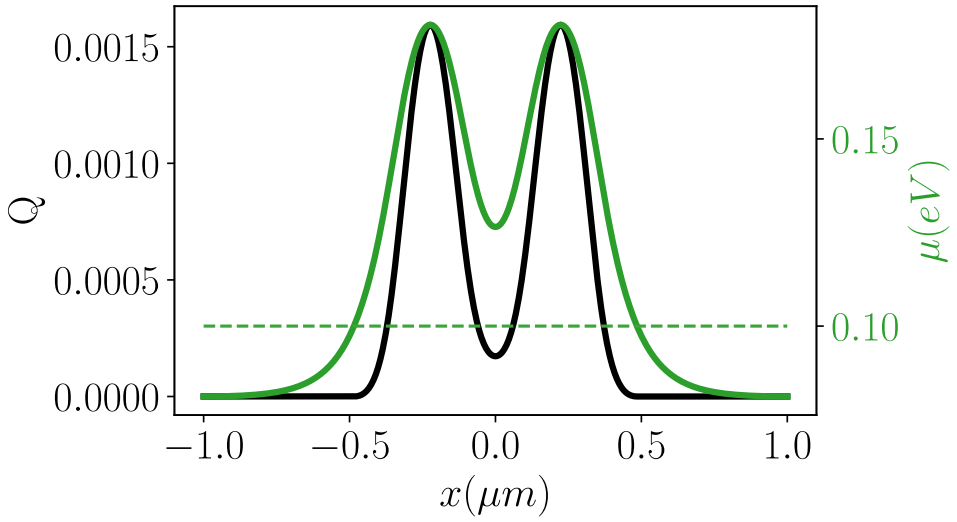


Figure 4.18: In black and green respectively the charge and chemical potential thomas fermi solution. Obtained with the piecewise Dichotomy algorithm with Script 4.25. The dashed green line is at the fermi energy $E_F = 0.1$

using the *flexible_input* entry. The rest of this example is identical to the previous two cases.

```
1
2  from pescado.poisson import Problem
3  from pescado.self_consistent import solver
4  from pescado.self_consistent import ildos
5  from pescado.tools import SparseVector
6
7  pd = Problem.load('Problem_tf_example_2D')
8
9  e_f = 0.1
10
11 quantum_sites = pd.flexible_indices[
12     np.abs(pd.coordinates[pd.flexible_indices][:, 0]) < 700]
13
14 quantum_cont_sites = pd.flexible_indices[
15     np.abs(pd.coordinates[pd.flexible_indices][:, 0]) < 300]
16 quantum_lin_sites = np.setdiff1d(quantum_sites, quantum_lin_sites)
17
18 empty_sites = np.setdiff1d(pd.flexible_indices, quantum_sites)
19
20 ildos_list = [ildos_plfa, ildos_cont]
21 sites_ildos = SparseVector(
22     values=np.zeros(len(quantum_sites)), indices=quantum_sites)
23 sites_ildos[quantum_cont_sites] = np.ones(len(quantum_cont_sites))
24
25 tf_mixed = solver.thomas_fermi(
26     ildos=ildos_list, sites_ildos=sites_ildos,
27     poisson_problem=pd, e_f=e_f)
28
29 initial_guess = SparseVector(
```

```
    values=np.zeros(len(quantum_sites)),
    indices=quantum_sites, dtype=float)

poisson_input_partial = {
    'voltage':sv_gate, 'charge_density':sv_dop,
    'flexible_input':{
        'charge':SparseVector(
            values=np.zeros(len(empty_sites)),
            indices=empty_sites)}}

tf_mixed.solve(
    poisson_input=poisson_input_partial,
    initial_guess=initial_guess)

charge_mixed = tf_mixed.quantum_charge(-1)
cp_mixed = tf_mixed.chemical_potential(-1)
```

Script 4.26: Define the piecewise continuous ILDOS shown in Fig.4.15

4.6 Conclusion

We have presented the python library *PESCADO* , a high level, lightweight, NLH solver package. While *PESCADO* is geared towards electrostatics, more precisely to the SCQE problem, it is constructed from various abstraction layers that may be used for other projects. Its geometrical engine, based on the concept of shapes, is very lightweight. We have found that it is sufficiently expressive to cover all the practical use cases that we have encountered so far. For instance, using shapes, it was relatively straightforward to construct a *PESCADO* simulation from the GDS files that are used as an input to the e-beam lithography machines, hence providing a direct tool to simulate a device prior to fabrication [Fouad Kalo 2023]. The finite volume mesher has been constructed in the same spirit. It provides enough control to adapt the grid to *e.g.* another python package in charge of a different part of the calculation. The next layer solves the electrostatic problem (LH) and the last layer the NLH problem. Future work will present an API that fully integrates this last layer with a quantum solver such as Kwant. At the moment this last step is left to the user but is relatively straightforward to do.

Part II

Applications

Chapter 5

Positioning of edge states in quantum hall graphene PN junction

Electronic interferometers probe the phase of quantum states. For instance, they can be used to read the phase of a quantum bit, e.g. flying quantum bits [Bäuerle *et al.* 2018], or observe fractional statistics, e.g. anyonic statistics of fractional quantum hall states [Nakamura *et al.* 2020]. In this Chapter we shall simulate a MZI. It is a two-path interferometer. In a MZI an initial pulse is split into two independent paths, the two arms of the MZI. When propagating along two different paths, the two pulses can gain a relative phase to one another. At the end of the MZI the two paths interfere. The interference pattern gives us information about the phase difference gained during propagation.

Although the MZI is a theoretically simple device, only recently has it been realized experimentally [Ji *et al.* 2003]. Double slit [Schuster *et al.* 1997, Ji *et al.* 2000] or Fabry-Pérot interferometers [de C. Chamon *et al.* 1997, Deviatov & Lorke 2008] are simpler to fabricate. However due to their multiple interference paths and open geometry (double slit), extracting information from the measured pattern is considerably harder. Since the MZI has only two paths interfering at a single spot, simulating their behavior is much simpler. However, it also means during experiments we must have a precise control of the path taken by the propagating pulse. The first MZI has been achieved using the quantum hall chiral edge states of the 2DEG in GaAs/AlGaAs heterojunctions [Ji *et al.* 2003]. First, the states are localized at the edge of the sample. Second, their chiral nature drastically reduces the probability for elastic scattering, thus avoiding counter-propagating states, and hence unexpected interference events. Often modern MZIs uses the chiral edge states, hence are devices that function under high magnetic fields. This is not a necessity though. Recently an MZI has been fabricated by drawing electronic highways in the underling 2DEG through precise gate placement [Yamamoto *et al.* 2012]. This implies a huge nanofabrication effort, notably on the lithography side. Hence MZIs using large magnetic fields still dominate the experiments.

Recently graphene has emerged as a another promising material for "electron quantum optics" experiments. A more general term encompassing experiments seeking to generate, manipulate and measure coherent quantum states. First, the use of hexagonal boron nitrate (h-BN) to encapsulate graphene instead of depositing the graphene layer directly on SiO_2 or other oxides has drastically increased car-

rier mobility [Dean *et al.* 2010]. Second, nearby graphite electrostatic gates screen the electrostatic interaction as well as the charges trapped in the substrate at the Si/SiO2 interface. MZIs based on graphene PN junctions now report record performances with interference visibilities nearing 100% [Wei *et al.* 2017, Jo *et al.* 2021].

Most features of these graphene Mach-Zehnder experiments could be understood within a LB picture, c.f. section 1.1.1.1. One observation, however, remained puzzling: the large separation W between the two interfering channels that form the two paths of the Mach-Zehnder, ranging between $W = 50$ and 200 nm, depending on the experimental setup [Wei *et al.* 2017, Jo *et al.* 2021]. In a naive LB picture, such a large separation should be associated with an abnormally high value of the exchange interaction, almost in the 100 meV range. In this Chapter we will show that this paradox is due to a breakdown of the LB picture in the QHE regime.

The LB approach does not account for the dominating electrostatic energy [Armagnat & Waintal 2020] and must be replaced by the more elaborate CSG model, see [Chklovskii *et al.* 1992a] and section 1.1.1.2. Performing the CSG construction of the compressible and incompressible stripes in a graphene pn junctions, our main result is a simple explanation to the above mentioned paradox. Even though the splitting between the interface states is indeed due to the presence of an exchange interaction, we find, in contrast to previous claims [Wei *et al.* 2017], that the actual value of the distance between the edge states is entirely controlled by the geometry of the device (via its electrostatic properties) and is essentially independent of the value of the exchange splitting.

This chapter is organized as follows. In Section 5.1 we introduce the electronic MZI. In Section 5.2 we describe the graphene MZI, notably the experimental device and main results of [Wei *et al.* 2017, Jo *et al.* 2021]. Then, in Section 5.3 we study qualitatively the role of exchange interaction on the edge state separation W. In Section 5.4 we follow CSG and perform the compressible / incompressible stripes reconstruction. Finally, in Section 5.5 we perform SCQE calculations of the compressible / incompressible stripes for the devices in [Wei *et al.* 2017, Jo *et al.* 2021]. It confirms the predominant role of electrostatics on the chiral edge state separation at the PN junction. This work has been published as:

> I.M. Flor, A. Lacerda-Santos, G.Fleury, P.Roulleau and Xavier Waintal. **Positioning of edge states in a quantum Hall graphene pn junction** Physical Review B 105, L241409 (2022)

5.1 Mach-Zehnder interferometer

The Mach-Zehnder is a two path interferometer. Figure 5.1 (a) shows an optical MZI. First a light beam is emitted by the source S. Then BS1, a beam splitter, splits the incoming light into two paths. The mirrors M1 and M2 redirect the two incoming beams s.t. they recombine and interfere at BS2. Afterwards the two outgoing beams reach the detectors D1 and D2. As the phase along one of the paths oscillates, the signals recovered from D1 and D2 oscillate out of phase.

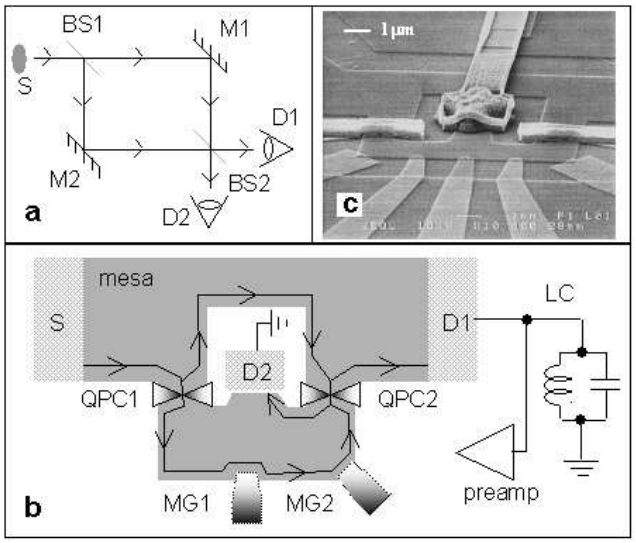


Figure 5.1: Taken from [Ji *et al.* 2003]. (a) Schematics of an optical MZI. (b) Schematics of the electronic MZI of [Ji *et al.* 2003] (c) Scanning electron microscopy image of the experimental MZI. For a detailed description of the schematics we refer to Section 5.1.

Figure 5.1 (b)-(c) shows the first realization of an electronic MZI by [Ji *et al.* 2003]. The MZI is defined on the 2DEG of an AlGaAs/GaAs heterostructure placed in the quantum hall regime. The light beam is replaced by a current carried by the chiral edge states of the 2DEG. The beam splitters BS1 and BS2 are replaced by respectively QPC1 and QPC2. The QPC1 splits current incoming from the source. Part of it is transmitted into the device "inner path" and the rest reflected into the device "outer path". They recombine at QPC2, interfere and are collected by D1 and D2. MG1 and MG2 are two gates used to define the edge of the (populated) 2DEG. Hence, they can change the inner path of the interferometer. The area A enclosed by the inner and outer paths can be tuned by V_{MG}, the voltage applied at the MG1 and MG2 gates.

Let B be the magnetic field perpendicular to the 2DEG plane. Then the Aharonov-Bohm effect implies that at QPC2 the phase of the electrons flowing through each path will differ by

$$\Phi = \frac{eBA}{\hbar} \tag{5.1}$$

Therefore the interference current measured oscillates as a function of B and A. Figure 5.2, adapted from Figure 2 of [Ji *et al.* 2003], shows how the current measured at D1 oscillates as a function of B and V_{MG}. The amplitude of oscillation is proportional to $cos(\Phi)$.

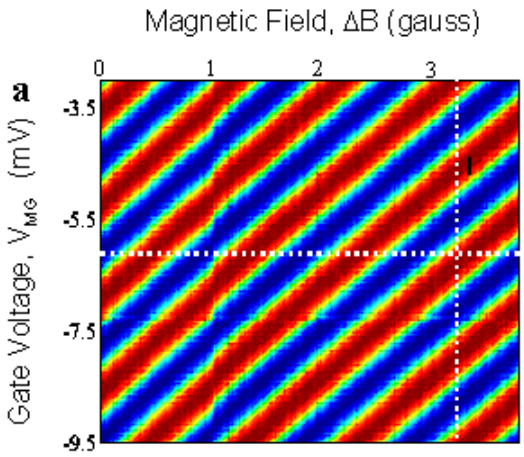


Figure 5.2: Taken from [Ji *et al.* 2003]. Interference pattern of the device in 5.1. The colormap shows the current measured at D1 as a function of gate voltage and magnetic field.

5.2 Graphene PN junction Mach-Zehnder interferometer

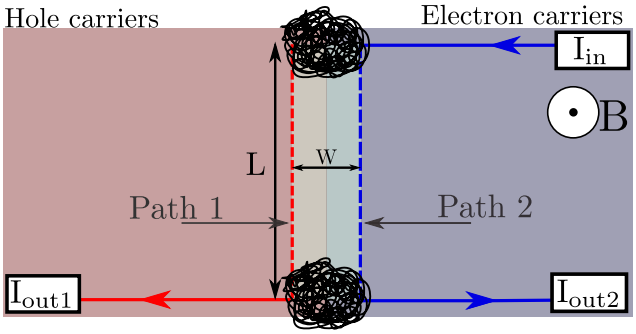


Figure 5.3: Schematics of a Mach Zehnder interferometer made using a graphene PN junction in the quantum hall regime. The P side is shown in red and the N side in blue. The current is injected from the right side. As it reaches the junction it splits into two, part of it is carried by the holes of the P side and the rest by electrons of the N side. Then they propagate separately along the junction, as shown by the dotted lines. As they reach the end of the junction, the two paths mix and are scattered to the right and left side of the device, where two detectors measure the current.

In this chapter we study the MZI measured by [Wei *et al.* 2017] and [Jo *et al.* 2021]. They are both fabricated using Graphene PN junctions. Similarly to the electronic MZI explained in Section 5.1, the device is placed under a perpendicular magnetic field B. This places the PN junction into the quantum hall regime, with counter-

propagating edge channels at each side of the junction and a insulating $\nu = 0$ state at the graphene's charge neutrality point.

Figure 5.3 shows a minimalist sketch of the device. The blue and red regions correspond to the respectively N and P type graphene. The counter-propagating edge channels are shown in the blue and red lines. The beam splitters are formed at the two opposing edges of the PN junction due to interchannel scattering, see hatched black circles. In the middle of the junction interchannel scattering is suppressed, hence the electrons propagate coherently on each side of the junction. This forms the two arms of the interferometer, see the dotted lines in Figure 5.3. The surface englobed by the two paths, WL, is the surface of the MZI capturing the magnetic flux. Figure 5.6 (c) shows a typical measurement of the MZI transmission as a function of magnetic field. From the period of oscillations, ΔB^{MZ}, and Eq.(5.1) we can extract the surface :

$$WL = \frac{h}{e}\frac{1}{\Delta B^{MZ}} \tag{5.2}$$

The question we seek to answer in this chapter is : what controls the separation W between the edge channels ? To answer this question we have developed the two models shown in Figure 5.4. In Section 5.2.1 we describe them in detail. To verify the W we extract from the models in Figure 5.4, we use as reference the experiments of [Wei *et al.* 2017] and [Jo *et al.* 2021], see Section 5.2.2.

5.2.1 Device geometry

We consider the two geometries (I) and (II) displayed in Figure 5.4. They closely mimic the experimental setups used in [Wei *et al.* 2017] and [Jo *et al.* 2021] respectively. We also consider a third geometry (III) studied in the supplementary material of [Jo *et al.* 2021]. It is essentially identical to (I) but with a different value of the distance between the graphene layer and the top gate, noted d_2.

A hBN encapsulated graphene monolayer is sandwiched by two gates. A bottom gate (at voltage V_b) spans the full graphene flake while a top gate (at voltage V_t) is only present on half of the flake, $x < 0$. By setting different values of the voltages V_b and V_t, one may form a pn junction with e.g. electrons accumulated under the top gate and holes in the other part of the sample. A magnetic field $\mathbf{B} = +B\hat{z}$ is applied perpendicular to the graphene flake to bring the graphene layer into the QHE regime. Here, we focus on the situation with $V_b < 0$ and $V_t > 0$ such that the filling factor is $\nu = 2$ at the n side and $\nu = -1$ at the p side, where $\nu = n_s h/(eB)$ (n_s: electron surface density). In the n region, two channels circulate counter-clockwise while in the p region one propagates in the clockwise direction. In setup (II), the two additional side gates (at voltage V_s) allow one to tune the transmission between the edge channels along the graphene boundaries and the interface states along the pn junction.

For $-2 \leq \nu \leq 2$, only a single Landau level is filled. This peculiar Landau level is pinned at the Dirac point (our energy reference) and is a specificity of the Dirac

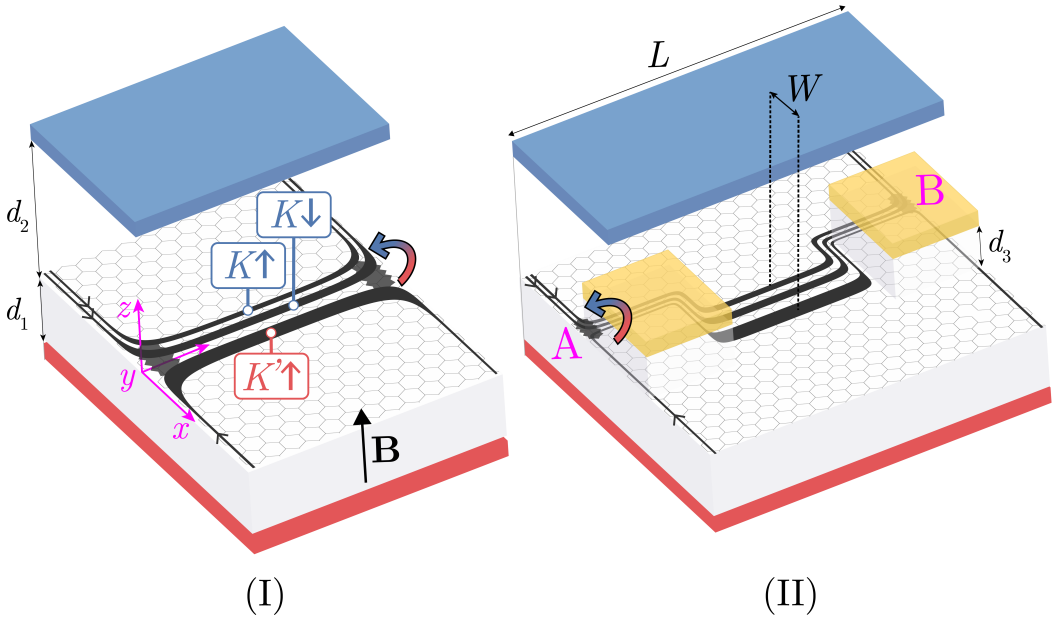


Figure 5.4: Two graphene pn junctions (I) and (II) respectively without [Wei *et al.* 2017] and with [Jo *et al.* 2021] side gates (in yellow). The top (blue) and bottom (red) gates create the pn interface. Insulating hBN fill the empty space between gates and graphene. A magnetic field $B = 9$ T is applied in the $\hat{z}$ direction. Two spin-split quantum Hall channels with valley isospin K propagate at the n side ($x < 0$). One spin-polarized channel with opposite valley isospin K' propagates at the p side ($x > 0$).

dispersion relation of graphene. In a non-interacting theory, this Landau level is degenerate in both spin $(\uparrow,\downarrow)$ and valley (K,K'). The exchange interaction E_{ex}, however, lifts this degeneracy [Werner & Oswald 2020]. The existence of interference in these experiments relies on the intervalley scattering at the intersection between the physical edges of the sample and the pn interface [Tworzydło *et al.* 2007]. In the LB picture, an incoming state — say $K\uparrow$ coming from the n side at $y=0$ — is scattered into a state $K'\uparrow$ (respectively $K\uparrow$) at point (A) in the pn junction with amplitude $S^A_{K'K}$ (respectively S^A_{KK}). We write the scattering eigenstate at point A as :

$$|\Psi^A\rangle = S^A_{K'K}|\uparrow,K'\rangle + S^A_{KK}|\uparrow,K\rangle \tag{5.3}$$

The state then propagates along the pn junction (near $x=0$ between $y=0$ and $y=L$) as a superposition of $K\uparrow$ and $K'\uparrow$. It does so coherently due to valley conservation along the interface [Tworzydło *et al.* 2007, Trifunovic & Brouwer 2019]. As it propagates along the junction, state $|\uparrow,K\rangle$ gains a phase difference $\Phi_{AB}=eBWL/\hbar$, s.t :

$$|\Psi(x=0,y=L)\rangle = S^A_{K'K}|\uparrow,K'\rangle + S^A_{KK}e^{i\Phi_{AB}}|\uparrow,K\rangle \tag{5.4}$$

Then, it is scattered again at the (B) corner with an amplitude $S^B_{KK'}$ (respectively S^B_{KK}) into an outgoing channel, say $K\uparrow$ towards the n side where it further propagates towards an Ohmic contact situated at $x=-\infty$, $y=L$. Note that the valley index K, K' is not necessarily well defined on the edges of the sample where intervalley scattering can occur (depending on the microscopic structure, say armchair versus zigzag) but we keep the same letter for labeling these states for convenience. The transmission probability of $|\Psi(x=0,y=L)\rangle$ onto, say state $|\uparrow,K\rangle$ propagating along the n side, is :

$$T_{MZI} = \left|\langle\Psi^B\,|\,\Psi(x=0,y=L)\rangle\right|^2 \tag{5.5}$$

with $|\Psi^B\rangle$ the scattering eigenstate at point B :

$$|\Psi^B\rangle = S^B_{KK'}|\uparrow,K'\rangle + S^B_{KK}|\uparrow,K\rangle \tag{5.6}$$

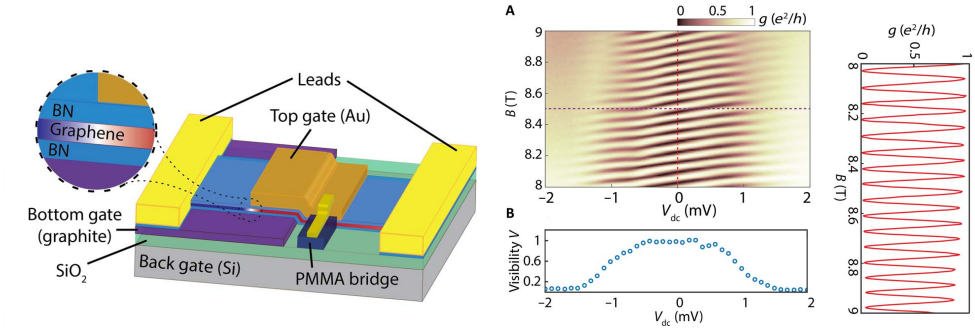
s.t.

$$T_{MZI} = \left|S^B_{KK'}S^A_{K'K} + S^B_{KK}S^A_{KK}e^{i\Phi_{AB}}\right|^2 \tag{5.7}$$

The resulting differential conductance obtained from the Landauer formula is

$$g = \frac{e^2}{h}T_{MZI} = \frac{e^2}{h}\left|S^B_{KK'}S^A_{K'K} + e^{i\Phi}S^B_{KK}S^A_{KK}\right|^2 \tag{5.8}$$

Note that in this picture, the $K\downarrow$ state has a different spin from the other two channels and is simply a spectator. Indeed, in the absence of magnetic impurities



(a) (b)

Figure 5.5: (a) Schematics of the graphene PN junction fabricated by [Wei *et al.* 2017] (adapted from Figure 1). The position where the junction is formed is shown by the zoom, with p type in red and n type in blue. (b) On the top left the conductance as a function of magnetic field and bias voltage V_{DC} for a filling factor of 1 in the n region and -2 in the p region. On the bottom left the visibility of the interferometer as a function of V_{DC}. On the right a 1D cut of the conductance as a function of magnetic field for a constant V_{DC} (red line on the top left figure). Adapted from Figure 3 of [Wei *et al.* 2017].

or spin-orbit coupling, spin is conserved along the edge states. Eq.(5.8) predicts that the conductance oscillates with magnetic field. The period of these oscillations directly provides the separation W between the edge states $K\uparrow$ and $K'\uparrow$ in the pn junction, the length L of the junction being defined by the sample geometry, c.f. Eq.(5.2).

5.2.2 Main experimental observations

Figure 5.5 (a) shows the device studied by [Wei *et al.* 2017], c.f. model (I) of Figure 5.4. Figure 5.5 (b) shows their main results. From Eq.(5.2) and the frequency of the conductance oscillations of Figure 5.5(b), we can extract $W = 52nm$. The left bottom figure shows the visibility of their device, defined as $V = (g_{max} - g_{min}/g_{max} + g_{min})$. The visibility reaches values as high as 98%.

Among the findings of [Wei *et al.* 2017], the most relevant to this chapter are: i) the edge channels belonging to the zeroth Landau Levels (LL) are well isolated from higher energy LL; ii) interchannel scattering does not occur at electrostatically defined edges, only at the physical edges and iii) the high visibility of the MZI indicates near perfect phase coherence at the PN interface.

Figure 5.6 shows the device of [Jo *et al.* 2021], c.f. model (II) of Figure 5.4. It differs from the device of [Wei *et al.* 2017] by the two side gates at the edges of the PN junction. By tunning the side gate voltages V_s, they move the edge channel away from the physical edge of the sample. When the edge of the PN junction is far enough from the physical edge s.t. no interchannel scattering is allowed, the beam splitter is in total reflection mode. They found that for $V_s > -0.3$ they obtain total

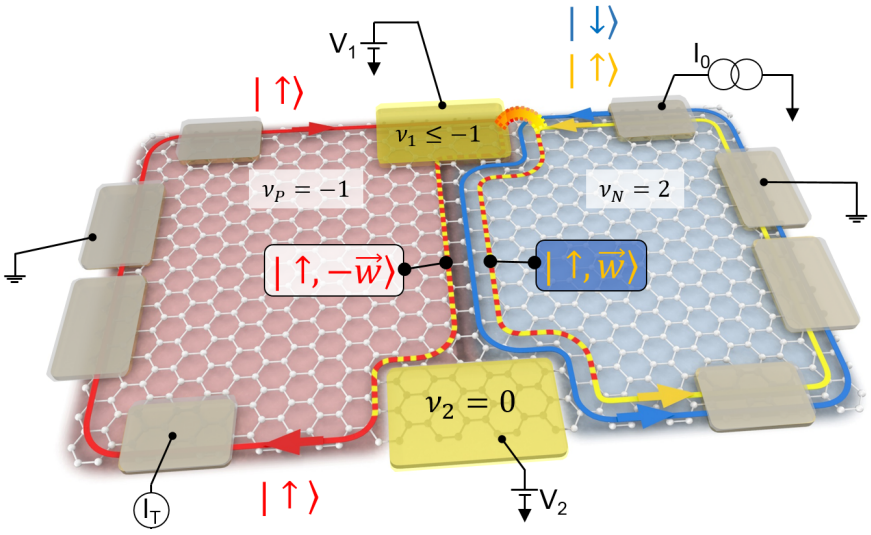


Figure 5.6: Schematics of the graphene PN junction fabricated by [Jo *et al.* 2021].

reflection at the beam splitters, c.f. [Jo *et al.* 2021] Figure 2(a).

Figure 5.7 is the main result of [Jo *et al.* 2021]. Figure 5.7 (a) shows the device transmission oscillations when the transmission at the second beam splitter is set to zero ($T_2 = 0$). Figure 5.7 (b) shows the device transmission oscillations for when $T_2 \neq 0$. They observed small scale oscillations superimposed to the large scale oscillations of Figure 5.7 (a). The small scale oscillations are those due to the magnetic flux captured by the MZI. From their periodicity ($\Delta B^{MZ} \approx 25mT$), they extracted $W = 110nm$. Using the periodicity on V_1 ($\Delta V_1^{MZ} = 50mV$), they also extracted how the area WL changes with V_1. They estimate a shift of the PN junction bellow the top gate of $\delta W \approx 1nm$ for a $\Delta V_1 = 100mV$.

5.3 Qualitative role of the exchange interaction in the value of the edge state separation W.

Lets go back to the model of Figure 5.4. We now focus on the pn junction and ignore the boundaries at $y = 0$ and $y = L$. In the Landau gauge, a Landau level with momentum k along the y-direction is centered along $x = k\ell_B^2$ where $\ell_B = \sqrt{\hbar/eB}$ is the magnetic length. In the presence of an electrostatic potential $U(x)$ that varies smoothly on the scale of ℓ_B, the dispersion relation of the propagating channels (within the Dirac point Landau level) takes the form :

$$E_p(k) = -eU(k\ell_B^2) + \frac{pE_{\text{ex}}}{4} \tag{5.9}$$

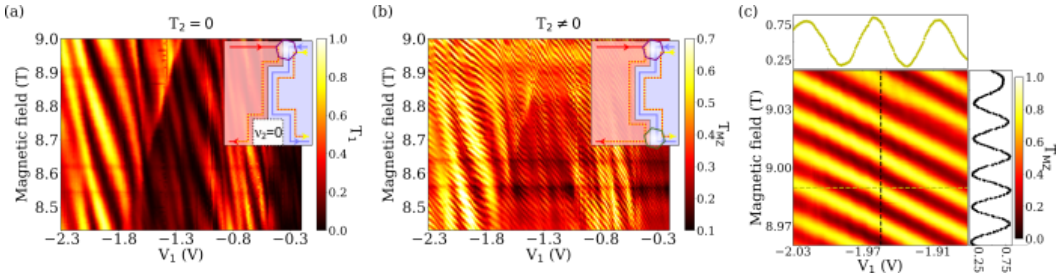


Figure 5.7: (a) Measurement of the top beam splitter transmission ($T_1 = I_T/(I_0/2)$) as a function of top side gate voltage V_1 and magnetic field. The filling factor of the graphene bellow the bottom side gate is set to zero (total reflection hence $T_2 = 0$). (b) Measurement of the Mach-Zehnder transmission (T_{MZ}) for a filling factor bellow the bottom side gate set to -1 (hence $T_2 \neq 0$). (c) T_{MZ} as a function of magnetic field and V_1 when the transmission at both beam splitters is the same and equal to 1/2. Taken from [Jo *et al.* 2021]

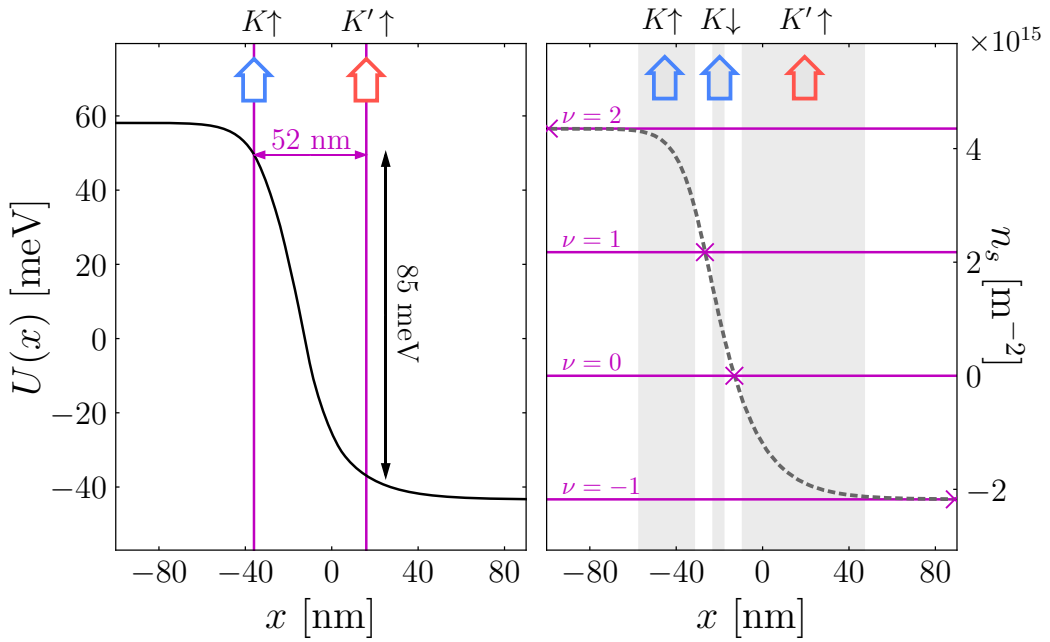


Figure 5.8: Calculations of a pn junction in the absence of magnetic field for the geometry (I) with $d_1 = d_2 = 20$ nm as in [Wei *et al.* 2017] and $V_b = -0.2$ V, $V_t = 0.6$ V. (Left) TF potential $U(x)$. Also shown is the value of $E_{\text{ex}} \approx 85$ meV needed to account for the experimentally observed $W = 52$ nm in [Wei *et al.* 2017] according to the LB picture. (Right) electronic density $n_s(x)$ in the pure electrostatic approximation. Also shown are the positions in x where the filling ν would take integer values at $B = 9$ T. These locations will become incompressible stripes (in white) separating the conducting compressible stripes (in gray) following the CSG picture.

where the integer $p \in \{\pm 1, \pm 3\}$ labels the 4 different channels with different valleys K, K' and spins $\uparrow, \downarrow$. Here, the exchange interaction energy takes the value E_{ex} for valleys and $E_{ex}/2$ for spins. As we shall see, this choice of values will be mostly irrelevant in what follows. It follows from the dispersion relation that, for two channels at the Fermi energy E_F, the exchange energy must exactly compensate the change of electrostatic energy due to the spatial separation. Hence, for a constant gradient of potential :

$$e\frac{\partial U}{\partial x}W = E_{\text{ex}}. \tag{5.10}$$

In Equation (S1.6) of [Wei *et al.* 2017], Eq.(5.10) was used to determine the width $W \approx E_{\text{ex}}/(e\partial U/\partial x)$. To calculate W they assumed the electric potential under magnetic field to be the same as the one in the absence of magnetic field. However this assumption is false and using it to solve Eq.(5.10) leads one to set unreasonably high values for E_{ex} if the experimental values of W [Wei *et al.* 2017] are to be recovered, e.g. left panel Fig. 5.8. It is instead the charge density that only slightly changes when the magnetic field is added, c.f. CSG picture in Section 1.1.1.2 (the potential can change quite a lot). Therefore, in the section bellow we argue that while Eq.(5.10) is strictly speaking correct, it cannot be used to determine W. In contrast, Eq.(5.10) defines the value the potential gradient takes while W is essentially determined by the geometry of the system (in the (I) geometry, $W \propto d_2$ the distance to the top gate).

5.4 Construction of the Chklovskiii-Shklovskii-Glazman (CSG) compressible and incompressible stripes.

Following CSG, see [Chklovskii *et al.* 1992a] and Section 1.1.1.2, we start by calculating the potential profile $U(x)$ and density profile $n_s(x)$ in the junction *in the absence* of magnetic field. We do so within the TF approximation. In our 2D geometry (infinite pn junction along y), there is no bulk electronic density so that Poisson equation reads:

$$\Delta U(\mathbf{r}) = 0 \tag{5.11}$$

with Dirichlet boundary conditions at the electrostatic gates. The graphene 2D electronic density $n_s(x)$ gives rise to a discontinuity of the electric field given by:

$$\frac{\partial U}{\partial z}(x, z = 0^+) - \frac{\partial U}{\partial z}(x, z = 0^-) = -\frac{e}{\varepsilon}n_s(x) \tag{5.12}$$

where $\varepsilon = 4\varepsilon_0$ corresponds to the hBN dielectric constant. In the TF approximation, the density is controlled by the bulk graphene density of state $\rho(E)$ which reads at zero temperature:

$$n_s(x) = \int_0^{\mu = eU(x)} dE\ \rho(E) \tag{5.13}$$

assuming that the Fermi energy $E_F = 0 = \mu - eU(x)$ (μ: chemical potential) is constant across the graphene sheet. In the absence of magnetic field, graphene has a linear density of states :

$$\rho(E) = \frac{2|E|}{\pi\hbar^2 e^2 v_F^2} \tag{5.14}$$

with $v_F \approx 10^6$ m/s the Fermi velocity in graphene. In the calculation shown in Figure 5.8, we adjust the top and bottom gate voltages in order for the electronic density in the bulk n and p regions to correspond respectively to that of $\nu = 2$ and $\nu = -1$ at $B = 9$ T. Note, however, that the magnetic field remains zero in the calculation at this stage.

The numerical calculation is performed using a generalization of the approach described in [Armagnat *et al.* 2019]. In the left panel of Figure 5.8, the electrostatic potential was calculated for the setup in experiment (I) where the distance between the top gate and the graphene layer is $d_2 \approx 20$ nm [Wei *et al.* 2017]. Assuming that this potential profile would be weakly affected by the magnetic field, one finds [using Eq.(5.10)] that the large value $W = 52$ nm observed experimentally requires an exceedingly large exchange energy of $E_{\text{ex}} \approx 85$ meV. As a reference, this value is almost as large as the distance to the next Landau level $\hbar v_F \sqrt{2}/\ell_B \approx 100$ meV. Such a large exchange energy would imply a deep reconstruction of the Landau levels that is not observed experimentally. The assumption that the electrostatic potential is unaffected by the magnetic field is in fact not valid [Armagnat & Waintal 2020]. In contrast, it is the electronic density $n_s(x)$ shown on the right panel of Figure 5.8 that is *almost unaffected* by the presence of a magnetic field. Indeed, modifying the electronic density can provide a gain in energy of the order of the exchange energy E_{ex} or the cyclotron frequency $\hbar\omega_c$ at a great loss in electrostatic energy. This is favourable only when the density is close to an integer filling factor. To understand this statement more quantitatively, suppose that the exchange energy or the cyclotron energy is at the origin of a change of density δn that is transferred at a distance x. The associated increase of electrostatic energy E_U due to the creation of this dipole is :

$$E_U \approx \frac{e^2 \delta n x}{c\varepsilon} \tag{5.15}$$

with $c = 1$ the geometrical constant for a simple planar capacitor geometry [Armagnat *et al.* 2019]. This translates into :

$$E_U \approx \frac{e^3 \delta\nu x B}{ch\varepsilon} \tag{5.16}$$

in terms of the variation of filling factor $\delta\nu$. For $\delta\nu \approx 1$, $x = 100$ nm and a magnetic field of $B = 10$ T, one arrives at $E_U \approx 400$ meV, an energy scale which is much larger than any other energy scale present in the problem. It follows that such an important variation of the density is impossible; such a large increase of electrostatic energy cannot be compensated by a gain in cyclotron or exchange

energy. Hence, the onset of quantum Hall effect can only change the density *a little and locally*.

Hence we consider the $B = 0$ density and, in the spirit of the CSG approach, we identify the positions in the right panel of Figure 5.8 that correspond to integer values of ν. Upon switching the magnetic field, a small region around these points will become incompressible stripes with a flat density $n_s(x) = \nu h/eB$. Away from these points, $n_s(x)$ is not constant which means that there must be one partially filled Landau level pinned at the Fermi level. These regions are the compressible stripes, where propagation is allowed. In these regions, the electrostatic potential $U(x)$ remains constant. We refer to [Chklovskii *et al.* 1992a, Chklovskii *et al.* 1993] for the details of the original construction and to [Armagnat & Waintal 2020, Armagnat *et al.* 2019] for a more recent version compatible with numerical calculations. Note that in the numerics, in contrast to the analytical construction, we do not assume that the change of density due to B and E_{xc} is small. In the CSG picture, the size of each incompressible stripe is proportional to $\sqrt{E_{\text{ex}}}$ [Armagnat *et al.* 2019]. Their positions, however, are entirely determined by the electrostatic potential at $B = 0$ hence by the geometry of the problem. In particular the width W, that corresponds to the distance between the centers of the two outer compressible stripes, is entirely determined by the electrostatics (hence independent of E_{ex}). Here we estimate $W \approx 62$ nm, without adjustable parameter, which is in good agreement with the experimentally found value $W = 52$ nm in experiment (I) for the same geometry. For experiment (III) with $d_2 = 50$ nm, we find $W = 90$ nm, also in good agreement with the value $W \approx 83$ nm found experimentally (see Figure S3 in the supplementary of [Jo *et al.* 2021]).

5.5 Numerical calculations of the compressible/incompressible stripe structure.

To actually calculate the stripes, we now use the finite B density of states. It is a sum of Dirac peaks at the positions of the Landau sublevels [Castro Neto *et al.* 2009]:

$$\rho(E) = \frac{1}{2\pi\ell_B^2} \sum_{n\in\mathbb{Z}} \sum_{\substack{p=\pm1\\,\pm3}} \delta\left(E - E_n - \frac{p}{4}E_{\text{ex}}\right) \tag{5.17}$$

where $E_n = \hbar v_F \text{sgn}(n)\sqrt{2|n|}/\ell_B$ are the Landau levels of degenerate graphene. This Generalized Thomas-Fermi (GTF) approximation includes the effect of the (Fock) exchange interaction phenomenologically. Indeed, it is only the existence of a splitting and not its exact value that affects the results presented here. At $B = E_{\text{ex}} = 0$, one recovers the TF approximation above. Considering only the $n = 0$ Landau level in the limit $E_{\text{ex}} = 0$, we obtain the Pure Electrostatic (PE) approximation, i.e. the graphene is subject to a Dirichlet condition with an equipotential $U(x, z = 0) = E_F = 0$.

The right panel of Figure 5.9 illustrates the three cases considered by showing

the integrated density of state Eq.(5.13) in bulk graphene. The results of the self-consistent calculation are shown in the upper (density) and lower (potential) left panel of Figure 5.9. They are fully consistent with the picture described in the above paragraph. We have also verified (not shown) that the width of an incompressible strip is indeed proportional to $\sqrt{E_{\rm ex}}$ and that the value of W does not depend on it. Hence the value of $E_{\rm ex}$ used in the calculations can be chosen arbitrarily. The energy ordering of the states with different spins and valleys depends on the competing interactions at the junction, and is here chosen to match the experimental observations. We note that the PE calculation approximates the quantum Hall graphene better than the $B = 0$ TF one. This is unsurprising since the PE approximation naturally captures the position of the $n = 0$ Landau level.

5.5.1 Effect of a side gate.

We now turn to the experimental setup (II) with additional side gates at voltage V_s. The main usage of the side gates is to control the scattering amplitudes of Eq.(5.8) in order to maximize the visibility of the interference pattern [Jo *et al.* 2021]. We retain two experimental findings associated with this side gate: (i) The interference is only present for negative values of $V_s < -0.3$ V, (ii) the period of the oscillations is equal to 25 mT, c.f. 5.2.2. The period of oscillations is roughly constant except close to $V_s = -0.3V$ where it is about 45% smaller, around 14 mT. The values are extracted from an analysis of the data of Figure 5.7, the qualitative period change is visible with the bare eye.

In Figure 5.10 (left), we distinguish four density profiles for different values of the potential V_s. When $V_s < 0$, all compressible stripes are situated at the left ($x < 125$ nm) part of the side gate. However, when $V_s > 0$, an incompressible region necessarily finds itself extended over the entire width of the side gate in between two Landau sublevels. The inter-channel separation in this case is increased dramatically. Consistent with observation (i), inter-channel scattering at the graphene edges A and B is expected to be fully suppressed. As for observation (ii), W steeply decreases as V_s tends away from zero, resembling the experiment. Quantitatively, the experimental results correspond to an average shift $dW/dV_s = 10$ nm/V; in our calculations, the center of the pn interface shifts by 17 nm over 2 V, which results $dW/dV_s = 8.5$ nm/V in close agreement. As a final quantitative comparison, we calculate the average edge-channel separation $W_{\rm avg}$ along the entire interface of setup (II). For this, we approximate the interferometer area of the more complex geometry and get :

$$W_{\rm avg} = (W_0(L - 2L_{sy}) + W_s(L_{sx} + 2L_{sy}))/(L + L_{sx}), \tag{5.18}$$

with : $W_0 = 195$ nm and $W_s = 40$ nm the calculated inter-channel separation without and with side gates respectively; $L_{sx} = 500$ nm and $L_{sy} = 200$ nm the side gate lengths in the x and y-directions respectively. This yields $W_{\rm avg}$ =102 nm. The same estimate in the experiments (effective area divided by $L + L_{sx} = 1.5\mu$m) gives $W_{\rm avg} = 110$ nm. We find again a very good agreement.

	Sample I	Sample II	Sample III
Exp W	52 nm	110 nm	83 nm
Num W	62 nm	102 nm	90 nm
Exp dW/dV_s		10 nm/V	
Num dW/dV_s		8.5 nm/V	

Table 5.1: Numerical vs experimental data. Sample I and II see Figure 5.4. Sample III has the same geometry as I, but with a different d_2 (distance between the top gate and graphene sheet). Sample I has $d_2 = 20$ nm and sample II $d_2 = 50$ nm

5.6 Conclusion

The results of this chapter, see Table 5.1, show that the edge states structure in a graphene pn junction can be understood quantitatively from the sole knowledge of the device electrostatics. Hence, the properties of these interferometers can be engineered by clever design of the device geometry. Compared to conventional semiconductors, it opens up new research avenues in electron quantum optics where interaction between propagating edge states can be precisely tuned. This should lead to the demonstration, in future experiments, of more complex quantum operations in graphene such as entanglement [Ionicioiu *et al.* 2001].

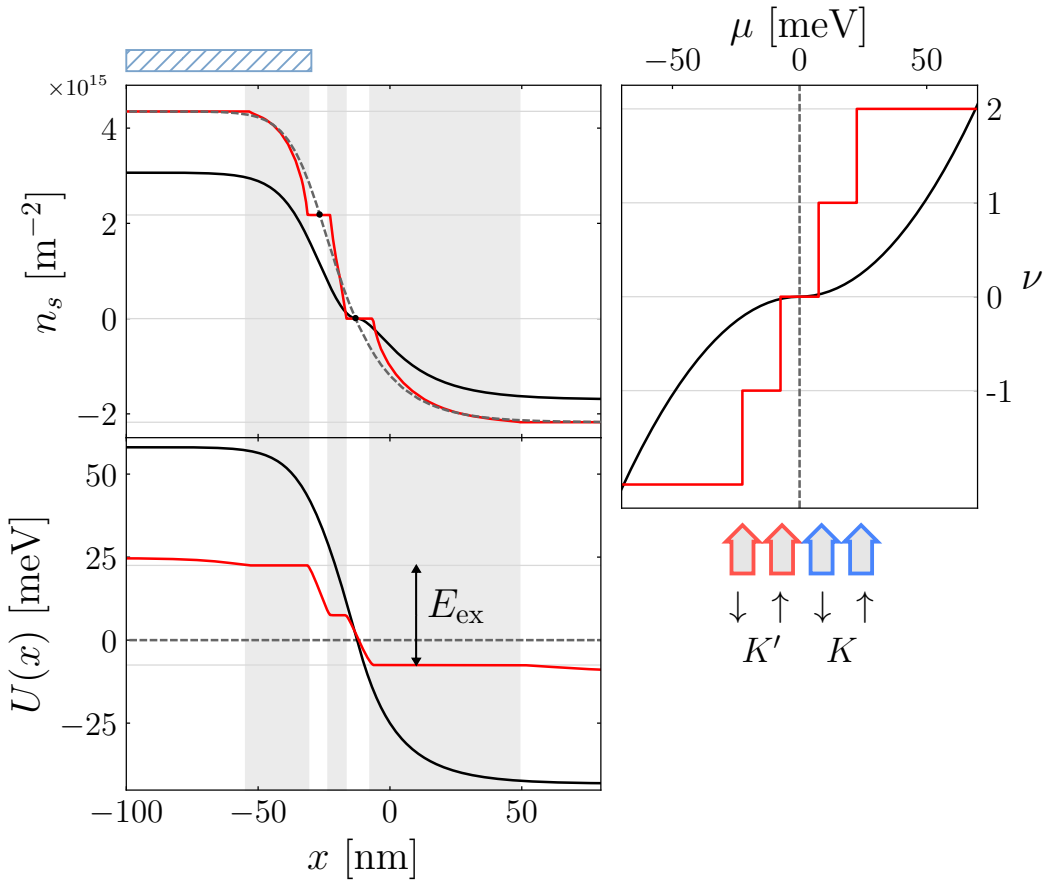


Figure 5.9: Reconstruction of the edge channels in geometry (I) with three approximations: pure electrostatic (PE) (dashed gray) and (generalized) TF with $B = 0$ (solid black) and $B = 9$ T (solid red) respectively. For each case, the density profile (top left), the carrier density in the zeroth Landau level (top right) and the potential (bottom left) are shown. Here $d_1 = d_2 = 20$ nm. The chosen value $E_{\text{ex}} = 30$ meV only affects the width of the incompressible region. The integer filling factors are shown with gray horizontal lines in both top plots. (In)compressible regions are shown in (white) gray patches and the horizontal position of the top gate is indicated by the blue hatched region.

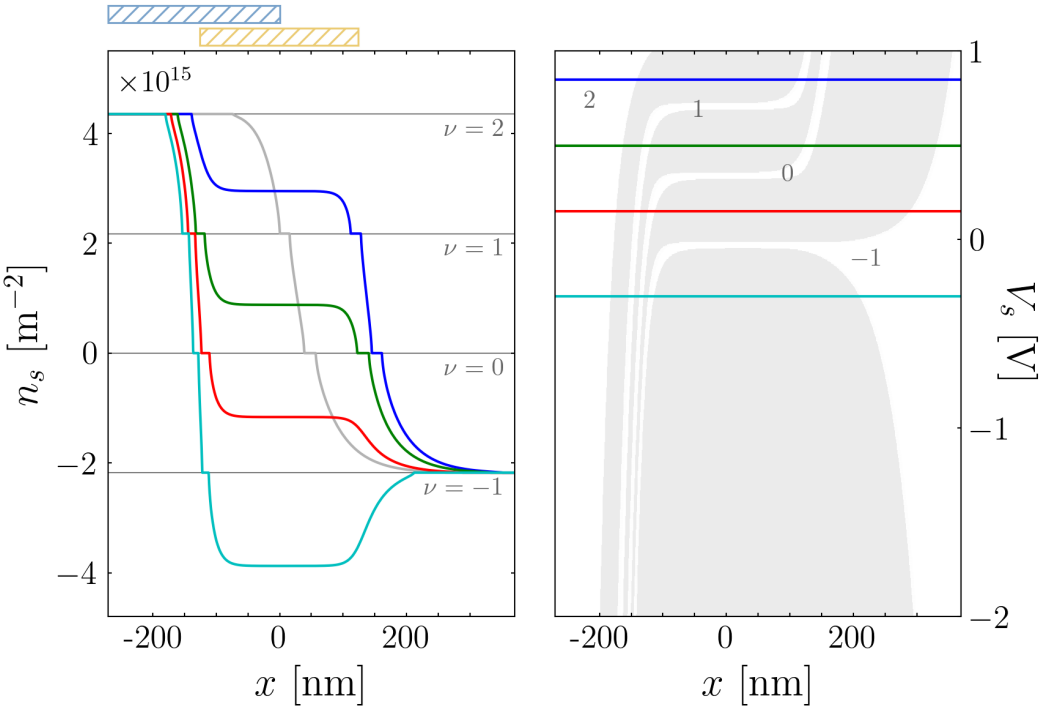


Figure 5.10: (Left) Density profiles calculated in the generalized TF approximation for $B = 9$ T and different values of V_s (color) and for an identical device without side gate (gray). The top and side gate horizontal position are indicated by the hatched regions. (Right) (In)compressible stripes as a function of V_s with lines drawn at the values for V_s depicted in (left). $d_1 = d_3 = 30$ nm and $d_2 = 60$ nm. Below $V_s = 0$, the inter-channel separation decreases from 48 nm to 35 nm.

Chapter 6

Predictive Modeling of GaAs-based nanoelectronic devices

A predictive model of nanoelectronic devices must be simple enough such that all of its parameters are extracted experimentally. At the same time, it must be general enough to predict the behavior of devices of different design. For instance, a single model must predict the behavior of devices with different gate geometry or dopant concentration. In this chapter we present the work published in [Chatzikyriakou *et al.* 2022]. We develop a model for nanoelectronic devices built on GaAs/AlGaAS heterostructures. We validate our model by assessing its capacity to predict a large experimental dataset of conductance measurements made on 110 devices with 48 different geometries. Indeed, due to the complexity of the semiconductor physics taking place in nanoelectronic devices, a match between the experimental data from a single device and quantum transport simulations is insufficient. It does not guarantee we have properly captured the device electrostatics. By making a comparative study between a single model and a large experimental dataset we put significant constraints on the modeling and asses its level of predictability. Furthermore, we have designed the experiments on well known systems such that their sole purpose is to validate our numerical model.

The work presented in this chapter is divided into two parts. First an experimental part where we generate the data-set used to develop our numerical model. Second a simulations part where we asses our model predictive power.

The experimental part provides the extensive data set we use to calibrate the modeling and assess its predictive power. Indeed, as pointed out recently by [Ahn *et al.* 2021], there is a lack of extensive experimental measurements of nanoelectronic, quantum devices in the literature. Our objective is to assess how well we can predict quantitatively the behavior of devices whose physics is supposed to be already well understood. Therefore, we worked in close collaboration with Chris Bauerle's experimental group at Néel Institute (hereafter referred to as CBG). CBG fabricated a large set of QPC on the 2DEG formed in a GaAs/AlGaAs heterostructure. They have measured the low temperature differential conductance of a total of 110 different QPCs with 48 different geometries of various shapes, widths and lengths. The full set of experimental data is published in [dat 2022].

The simulation part predicts the different values of the gate voltages where the QPC conductance vanishes, the "pinch-off" voltages. The pinch-off values are

unaffected by the low energy physics affecting the quantum behavior of the device. Understanding them amounts to understanding the charge distribution in the device. That is, the physics in the meV–eV range. Only when one is confident that this physics is taken care correctly, it makes sense to try to predict the physics taking place at lower energies. That is, only when one can correctly predict the pinch-off voltage for any QPC among the 110 devices fabricated to generate the data set, one can hope to develop a model precise enough such that it can correctly capture the relevant quantum physics. Hence, the simulations we perform aim at giving a quantitative answer to the question: "where are the charges in the device?"

This chapter is organized as follows: Section 6.1 summarizes our main findings: We show that the experimental pinch-off voltages match the predictions of the simulations within a $\pm 5\%$ accuracy. Section 6.2 describes our experimental protocol. Section 6.3 explains the model used in the simulations. In section 6.4 we present the comparison between the experimental data and the simulations. We end this article with section 6.5, which contains a critical discussion of the modeling.

The experiments were performed by C.Baurle's group at Institut Néel. The initial model development and simulations was performed by me. Most of the simulations generating the results presented in this chapter were done by Eleni Chatzikyriakou. The work in this Chapter is published as :

> Eleni Chatzikyriakou, Junliang Wang, Lucas Mazzella, Antonio Lacerda-Santos, Maria Cecilia da Silva Figueira, Alex Trellakis, Stefan Birner, Thomas Grange, Christopher Bäuerle and Xavier Waintal. **Unveiling the charge distribution of a GaAs-based nanoelectronic device: A large experimental dataset approach.** Physical Review Research 4, 043163, 2022.

6.1 Summary of the approach and model predictions

CBG fabricated and measured a large set of QPCs of various shapes and sizes. Quantum points contacts are one of the simplest devices used in quantum nanoelectronics [van Wees *et al.* 1988, Wharam *et al.* 1988]. Despite their simplicity, there remains open questions about their behavior, notably in the regime called 0.7 anomaly [Thomas *et al.* 1996]. In the work presented in this Chapter we do not focus on the 0.7 anomaly nor on the conductance quantization, but we rather establish, on firm grounds, the electrostatic potential seen by the conducting electrons. This amounts to understanding the charge distribution in the device. To reach this goal, we perform a systematic comparison between the simulated and measured "pinch-off" voltages.

Experimentally, the pinch-off voltage is the voltage value applied to the electrostatic gates such that the conductance vanishes or presents a cusp — an indication that the 2DEG gets fully depleted in some part of the system.

Figure 6.1c shows a schematic of a typical device (see Fig. 6.5 for a SEM picture with the scales). The device (zoomed-in inset) has a transistor-like geometry with

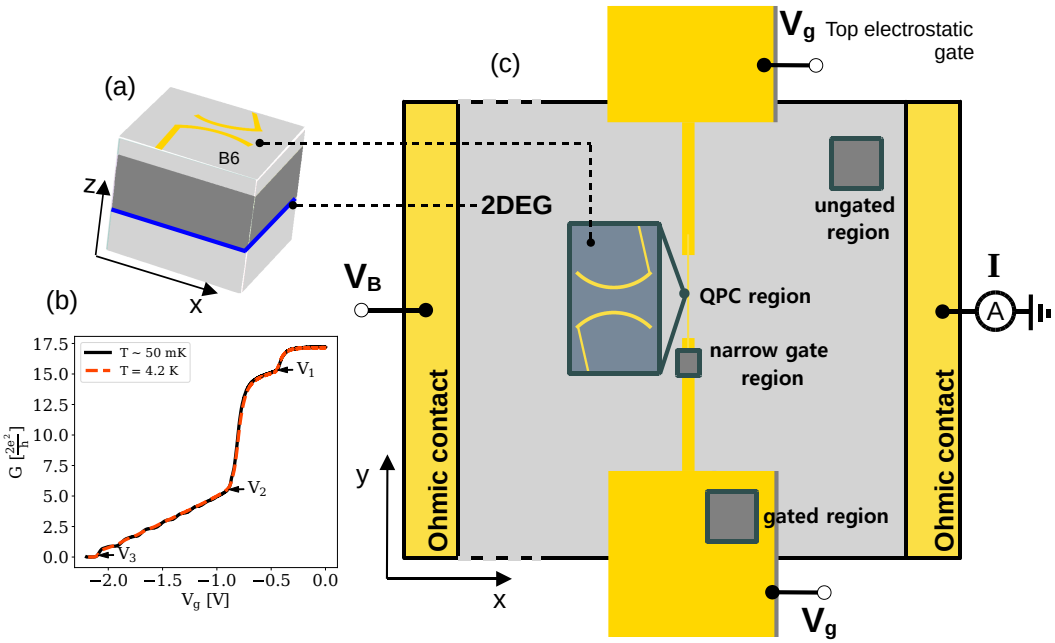


Figure 6.1: a) Schematic of the 3D stack with GaAs (dark gray), AlGaAs (light gray), top QPC gate (yellow) and the 2DEG region (blue). b) Typical experimental curve for gate voltage measurements. The three different points, V_1, V_2 and V_3, correspond to values of the gate voltage where the gas is depleted underneath the different gate regions. V_1 depletes the gas in the *gated* region, V_2 in the *narrow gate* region and V_3 in the *QPC* region. c) Simplified top view of a device with a transistor-like geometry. The Ohmic contacts (source and drain) and the electrostatic gates (situated $\approx$ 110 nm above the 2DEG) are indicated in yellow. For the simulations, the system is broken into 4 different subregions tagged *ungated*, *gated*, *narrow gate* and *QPC* region, see text.

source and drain Ohmic contacts and electrostatic split gates. Applying a negative voltage V_g on the gates depletes the 2DEG underneath. As we indicate in Fig. 6.1c, each gate is further divided into three regions of different width. The region closest to the border of the 2DEG is very wide (several μm) and is called the "gated region". A second region of intermediary width (50 nm) is noted "narrow gate" region. Finally, the "QPC" region is located at the middle of the device where the gates split. A sketch of the full stack, a standard high mobility GaAs/AlGaAs heterostructure, is shown in Fig. 6.1a.

The current I versus gate voltage V_g characteristics for each device were measured; see Fig. 6.1b for a typical experimental trace. As one decreases V_g from zero towards negative values, one first depletes the 2DEG underneath the "gated" region. Indeed, the large width of the gates (several μm) on this region compared to the rest of the split gate means the 2DEG will first be depleted underneath it. The value for which the 2DEG is depleted underneath the "gated" region is denoted V_1. There, one observes a cusp in the current–gate voltage curve, as indicated on Fig. 6.1b.

In the simplest model for V_1, accurate within a few percent (see the discussion in section 6.5), the 2DEG and the electrostatic gate form a simple plane capacitor. The electron density in the gated region is given by:

$$n(V_g) = n_g - \frac{\varepsilon V_g}{ed} \tag{6.1}$$

with n_g the electronic density in the gated region with zero volts applied to the gate, $\varepsilon \approx 12\varepsilon_0$ the dielectric constant, $d = 110nm$ the total distance between the 2DEG and the gate.

It follows that V_1 is an almost direct measure of the electronic density in the gated region:

$$n_g \approx \frac{\varepsilon V_1}{ed}. \tag{6.2}$$

As one further decreases the gate voltage, one eventually depletes the gas below the "narrow gate" region. This region is tens of micron long along the y direction, but only 50 nm wide. A second cusp in the conductance versus V_g curve is observed at the voltage V_2 where this region is fully depleted. Finally, as one continues to decrease V_g towards strongly negative values, the gas is depleted in the central QPC region. At that moment the conductance between the left and right Ohmic contacts vanishes entirely. We denote the gate voltage at which this depletion is observed as V_3. The set of voltages V_1, V_2 and V_3 reflect the initial density at various parts of the sample and the interplay between the field effect of the gate and the screening of the 2DEG. This is the main data we use to asses the predictive power of our model. The full set of current-gate voltage characteristics is provided as a zenodo archive [dat 2022]. They could be further used to study, e.g. conductance quantization.

In order to predict the different values V_1, V_2 and V_3, we perform a different type of calculation for each of the three gate regions. The dimensions of the "gated" and

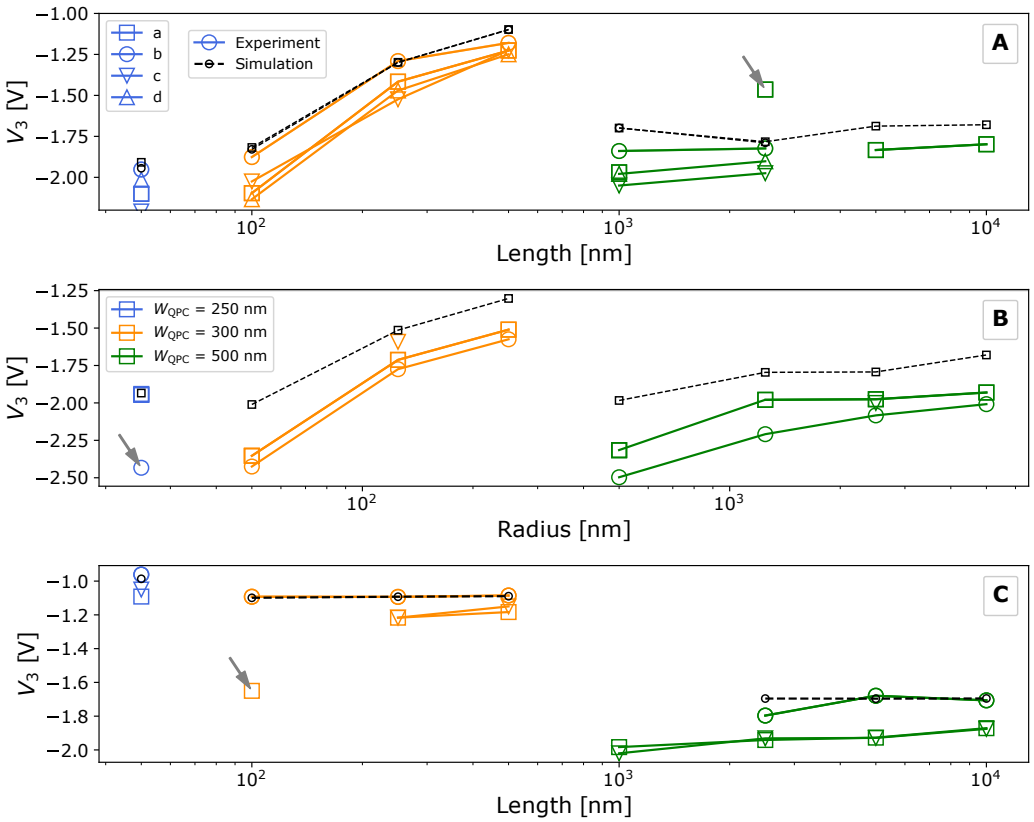


Figure 6.2: Comparison between simulation (small black symbols and dashed line) and experiment (color symbols and solid line). The QPCs are grouped according to design: A (top), B (middle), C (bottom). The results have been color-coded according to the width of the QPC, W_{QPC} (Figure 6.4): blue for 250 nm, orange for 300 nm and green for 500 nm. The different symbols correspond to different devices with identical nominal characteristics but at different locations in the wafer, c.f. Figure 6.6. A letter is attributed to each symbol: “a" (rectangles), “b" (circles), “c" (up triangles) and “d" (down triangles) so that a given QPC is uniquely identified by its geometry (A, B, C or equivalently upper, middle and down panel), its rank (1-8 from left to right in the figure) and the letter a,b,c,d. For instance QPC “A5b" corresponds to the fifth circle in the upper panel. Arrows point to outliers that we attribute to lithography problems or structural damage during cooldown or initial measurements, see text.

"narrow gate" regions have been kept constant for all QPCs. Hence we expect very little sample to sample variation of the experimental value for V_1 and V_2. The value of V_3, however, corresponds to the "QPC" region that has been varied in different devices.

- To calculate V_1, one simulates the "gated" region. It can be approximated as infinite along x and y directions due to the large dimensions of the gates. Therefore one only needs to perform 1D simulations along the z direction. Additional 1D simulations were performed for the "ungated" region, *i.e.* without top gate. It allows one to calculate the 2DEG bulk density n_s far away from the gates. Such value can be compared to the experimental bulk density $n_{bulk} = 2.8 \cdot 10^{15}$ m^{-2} obtained by Hall measurements.

- To calculate the value of V_2, we simulate the narrow gate region. The latter is very long along the y direction (up to 50 μm), but very narrow (50 nm in most samples) along x. Hence we consider a system infinite along y and need only to perform 2D simulations in the (x, z) plane. We decrease V_g until the density vanishes underneath the middle of the *narrow* gate. Then we record the associated value of V_g as V_2.

- To calculate the value V_3 we perform a full 3D simulation of the "QPC" region. The V_3 value is then extracted by decreasing V_g until the density vanishes underneath the middle of the gap between the two gates. At $V_g \leq V_3$, the 2DEG is split into two disconnected left and right parts.

The model we used to simulate the devices has two *a priori* independent input parameters: the dopant density n_d and surface charges density n_{sc} (see Sec.6.3). With this model we do not attempt to predict the experimental bulk 2DEG density in the ungated (n_{bulk}) or gated ($\propto V_1$) regions. Instead, we calibrate the model values of n_d and n_{sc} by fitting the model to the experimental values of V_1 and n_{bulk}. This calibration sets the value of the electronic density in the model in the ungated (n_s) and in the gated (n_g) regions. While $n_s = n_{bulk}$ after calibration, we keep two different letters for the model and experimental values, respectively, for clarity. Predicting n_s and n_g would imply having a precise knowledge of many microscopic parameters. Accurate values of the dopant ionization energy, dopant concentration, surface states energy, band alignment, dielectric layers thickness etc. would have to be obtained either from theoretical arguments or from experiments. This is a hard task, and also not necessary for the physics we seek to understand, the transport properties - see Section 1.3.1. In Section 6.5 we argue that n_d and n_{sc} are in fact *not* independent and that a single effective input parameter may be used (Fermi level pinning). This further increases the predictive power of our model. However, the relation between n_d and n_{sc} has not been assumed in the simulations and is considered here as a prediction of the modeling.

Once n_d and n_{sc} have been calibrated, we then proceed to predict the V_2 and V_3 pinch-off voltages. In Fig. 6.2 we compare the experimental (color symbols) to the

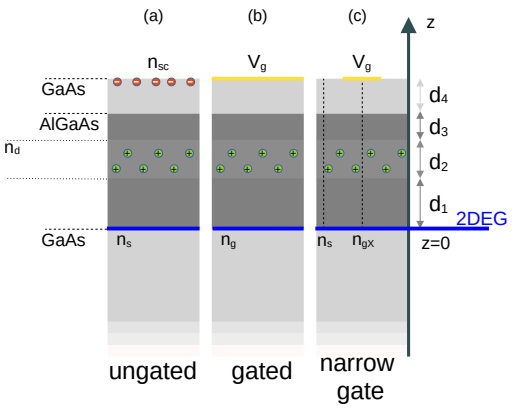


Figure 6.3: Panel (a): Side view of the experimental heterostructure stack. The widths of the different layers are respectively $d_1 = 25$ nm, $d_2 = 65$ nm, $d_3 = 10$ nm and $d_4 = 10$ nm. The central AlGaAs layer of width d_2 is doped. Panel (a), (b) and (c) correspond, respectively, to the ungated, gated and narrow gate regions as indicated in Fig. 6.1c. In the simulations, (a) and (b) correspond to 1D models without and with a top gate, respectively, while (c) corresponds to a 2D model with a gate of finite width (50 nm) at its surface.

simulated (black symbols) values of V_3 for the different QPC designs; see Fig.6.4 for the latter. It shows a systematic agreement of the theoretical prediction for V_3 with that obtained experimentally within a precision of 10% or better. Figure 6.2 implies that we can reliably predict the spatial variations of the electronic density in devices of arbitrary geometries. This opens the path for making quantitative calculations at smaller energy scales and predict genuine quantum effects quantitatively and without fitting parameters.

Beyond the overall agreement between experiments and simulations, Fig. 6.2 further shows significant sample to sample variations for nominally identical samples as well as systematic deviations (the simulation curves being systematically above the experimental ones). These features, which we attribute to disorder, will be discussed later in this chapter.

6.2 Experiments: details of the set of quantum point contact devices

The experimental samples were fabricated on a Si-modulation-doped $GaAs/Al_{0.34}Ga_{0.66}As$ heterostructure grown by molecular beam epitaxy (MBE). The high mobility 2DEG lies at the GaAs/AlGaAs interface, located 110 nm below the surface. Performing Hall measurements at 4.2 K under dark conditions, CBG found a bulk 2DEG density of $n_{\rm bulk} \approx 2.79 \times 10^{15}$ m^{-2} and a mobility of $\mu \approx 9.1 \times 10^5$ cm^2/Vs. The corresponding Fermi wave-length is $\lambda_{\rm F} = \sqrt{2\pi/n_s} \approx 47$ nm. The surface electrodes that define the QPCs are made out of a metal stack of

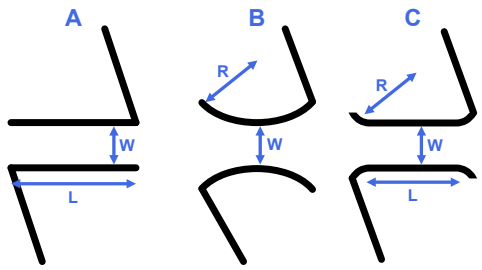


Figure 6.4: Schematic of QPC designs: Rectangular (A), Round (B) and Smooth (C). The characteristic geometrical parameters L (length), W (width) and R (radius) are indicated by arrows.

4 nm titanium and 13 nm gold, deposited by successive thin-film evaporation. The composition of the stack of the heterostructure is shown in Fig. 6.3a together with the widths of the different layers.

In order to investigate the geometrical influence of QPCs, we designed three kinds of shapes: Rectangular (A), Round (B) and Smooth (C) (see Fig. 6.4). Rectangular (A) designs correspond to a wire of length L defined by two parallel gates separated by width W. Round (B) designs consist on two semi-circular gates with radius R that define the point contact. At last, Smooth (C) designs belong to an intermediate design between A and B, combining the linear constriction with adiabatic entrances.

For each design (A,B,C), 16 different combinations of geometrical parameters L, R and W are investigated, from the smallest (A1, B1, C1) to the largest (A16, B16, C16) sizes.

Figure 6.5 shows Scanning Electron Microscopy (SEM) images of various fabricated designs; see Appendix A for exact parameters. To account for statistical variability, devices with the exact same design are repeated across the chip. We label them with an additional Latin letter ("a" to "d") in the device name. For example, A2a and A2b are different QPCs with identical nominal characteristics.

In order to maximize the number of measured devices in a same cooldown, a set of 8 QPCs is placed in series sharing a common pair of Ohmic contacts (see top panel in Fig. 6.5). With a separation more than 40 μm, we ensure that no mutual effect occurs between the neighboring QPCs. We call such a set of 8 QPCs, a sample. We draw attention to the fact that we follow this notation throughout this chapter, as different such sets of QPCs (samples) present larger deviation in their measured characteristics than QPCs within the same sample.

CBG fabricated and measured a total of 110 QPCs with 48 unique designs that are distributed in 16 sets on a chip of 10 mm $\times$ 8 mm. A schematic layout is shown in Figure 6.6. The sample that contains a given QPC can be identified by the a column index X and a row index Y. For example, the device A2a is located in the set X=1 and Y=2.

The conductance characterization was performed at two temperatures T $\approx$ 4.2 K and T $\approx$ 50 mK. Unless stated explicitly, all the data shown below have been

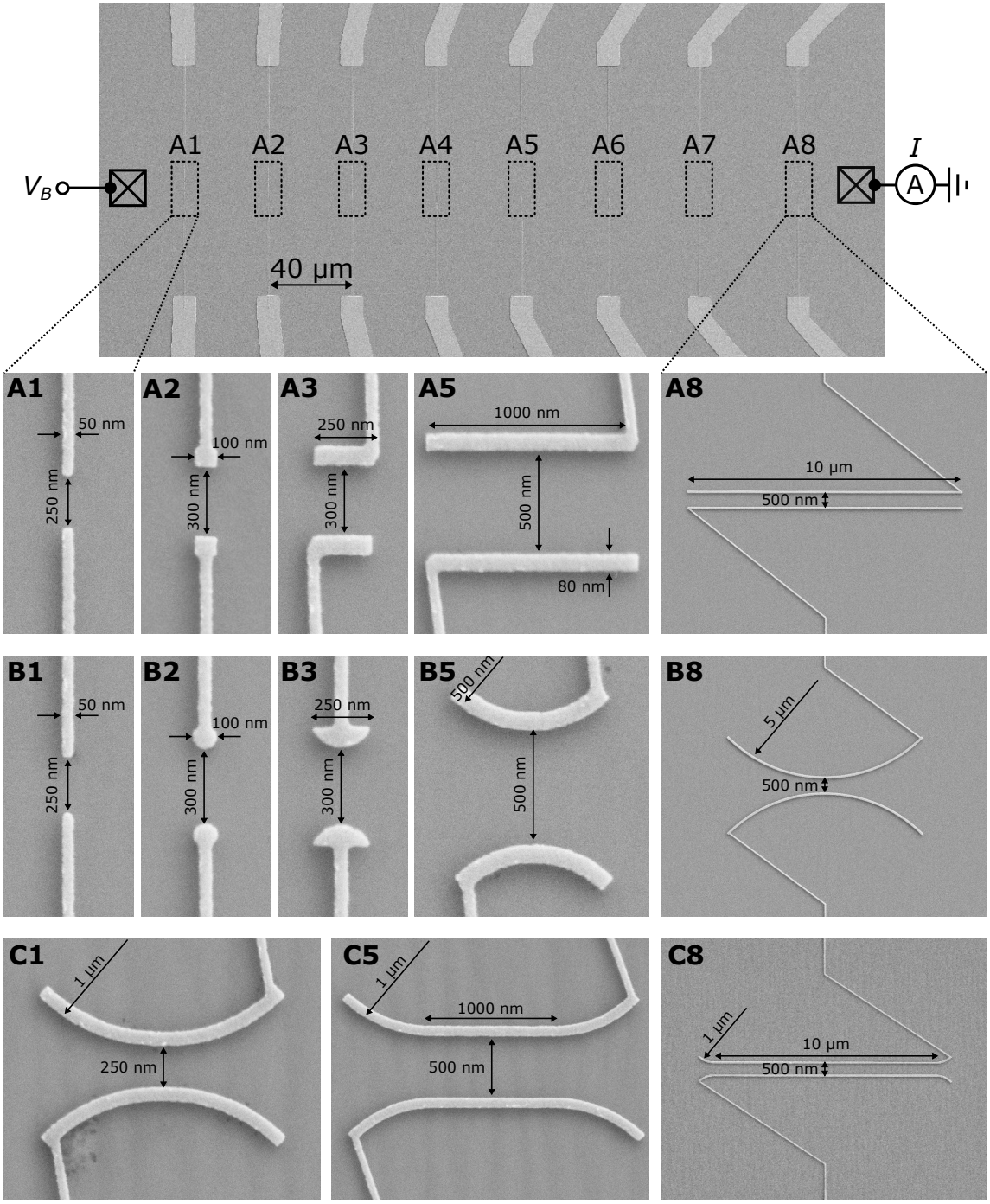


Figure 6.5: Scanning electron microscopy (SEM) images of QPCs. (Top) Overview of a set of 8 QPCs in series sharing a pair of Ohmic contacts (a sample). (Bottom) Examples of investigated shapes (A, B, C).

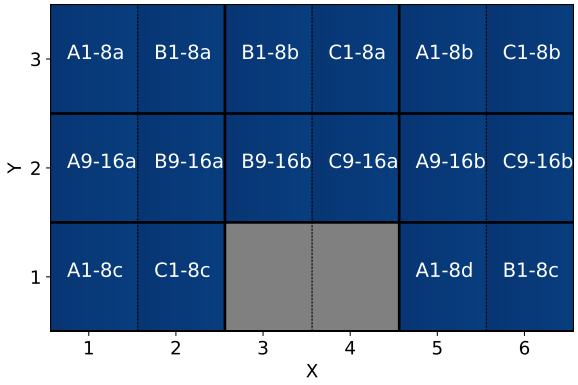


Figure 6.6: Repartition of the QPCs in the GaAs dice. A set of 8 QPCs (a sample) is represented as a rectangular box identified by its X and Y indices. Gray areas have not been used. The dimensions of one sample are $\sim 1.6 \times 2.3$ mm^2.

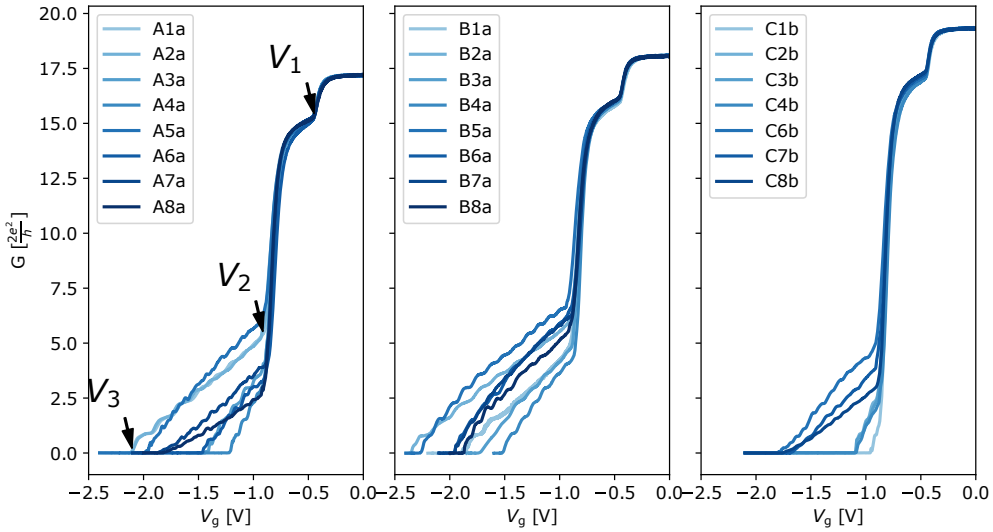


Figure 6.7: G = I/V_b versus V_g measurements for 3 sets of QPCs (A1-8a, B1-8a, C1-8b) with an Ohmic bias $V_b = 500\mu$V at T $\approx$ 50 mK. The arrows indicate the characteristic voltage drops V_1, V_2 and V_3. Note that the current $I_0(V_g = 0)$ is the same for all QPCs in a given set. This is because of the common contribution from the Ohmic contact.

taken at 4.2 K as only a limited number of samples have been measured at 50 mK. While the temperature strongly affects features like conductance quantization, the temperature variations of the pinch-off voltages can be ignored, as one can observe in Fig. 6.1b. We note, however, that there is a small decrease of $\leq$ 25 mV of the V_3 pinch-off voltages between 4.2 K and 50 mK. This small variation is irrelevant here considering the level of accuracy of the simulations and the sample to sample experimental variations. CBG apply a bias voltage $V_B = 500\,\mu$V between the Ohmic contact to induce the current I. To characterize the transport properties, CBG measured the current I as a function of surface-gate voltage V_g for each device. The full data set of these transport measurements can be found in [dat 2022].

Figure 6.7 shows conductance versus V_g measurements for various QPCs at 50 mK temperatures, which have more pronounced quantization features than those at 4.2 K. Three distinct regions can be identified separated by the pinch-off voltages V_1, V_2 and V_3. In the first two regions ($V_g \geq V_2 \approx$ -0.75 V), different devices share the same conductance behavior. This is expected as in this regime the current is dominated by the electron flow in the large "gated" or the "narrow gate" regions, which is identical for all QPCs (see Fig. 6.1). In the third region ($V_g \leq V_2$), the transport properties are only affected by the narrow constriction formed between the gates. Clear conductance quantization steps are observed for numerous QPCs with wide-ranging pinch-off voltages V_3. Note that the pinch-off voltages V_1 and V_2 are also visible when one biases only one of the two gates (e.g. top or bottom). Also note that we show the raw data without subtraction of the series resistance due to the Ohmic contacts and measuring apparatus.

A few samples deviated significantly from the theoretical predictions, as indicated by the grey arrows in Fig. 6.2. CBG have performed a visual inspection of the SEM image of some of these samples which did not reveal any particular problem. We attribute these outliers to fluctuations of the density in the QPC region due to *e.g.* a fluctuation of the concentration of dopants above.

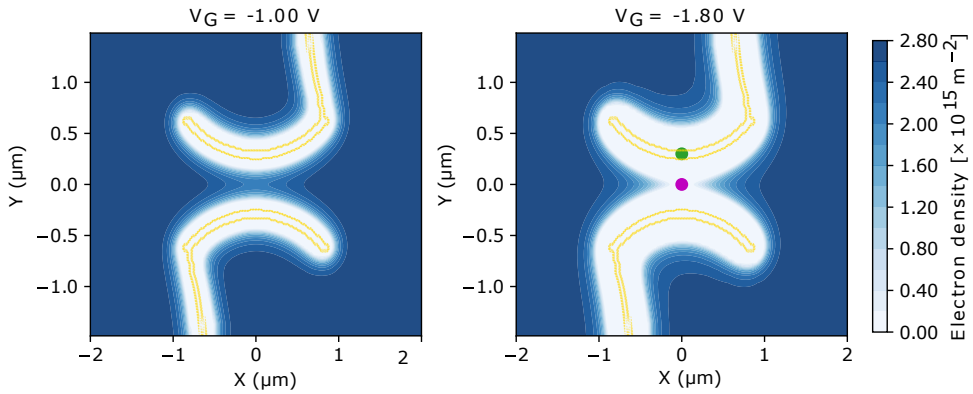
As a general trend, we find the conductance plateaus to get quickly washed out upon increasing the temperature to 4.2 K or making the sample too long (only the ones with $L \leq 250$ nm showed clear plateaus). This is commonly observed by other experimental groups.

6.3 Simulations: details of the modeling

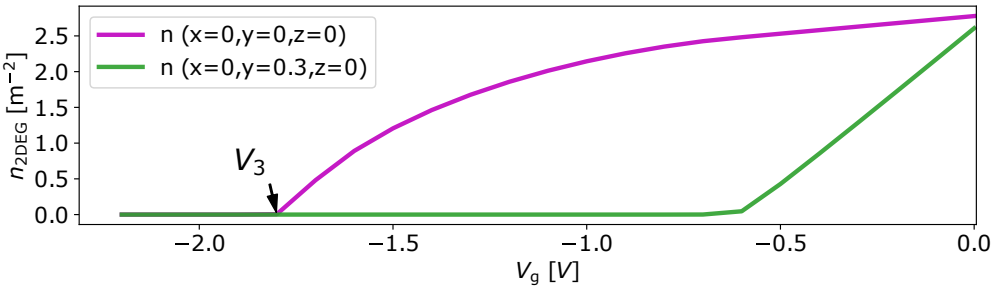
The simulations predicting the the pinch-off values of the CBG QPCs were done within the Thomas-Fermi approximation at zero temperature using the commercial software nextnano++ [Birner *et al.* 2007, Trellakis *et al.* 2007]. At the time *PESCADO* was still in early development and was not ready to be used on the scale required by this project. This is why nextnano++ was used.

We model the QPC using the self-consistent Poisson equation,

$$\vec{\nabla}.\left[\varepsilon(\vec{r})\vec{\nabla}U(\vec{r})\right] = eN[\mu = E_F + eU(\vec{r})] - eN_d(\vec{r}) + eN_{sc}(\vec{r}) \tag{6.3}$$



(a)



(b)

Figure 6.8: (a) Simulation of the electron density distribution in the 2DEG for QPC B6 at different voltages. Left: $V_g = -1.0$ V. Right: V_g = -1.8 V. (b) Density versus V_g at the two different points indicated in the right (a) panel. V_3 is identified as the value for which n($x = 0,\ y = 0,\ V_g = V_3$) vanished.

where $U(\vec{r})$ is the electrostatic potential, $\varepsilon(\vec{r})$ the dielectric constant, $N_{\rm d}(\vec{r})$ is the ionized dopant density in the doped layer, $E_F = 0$ the Fermi level (electro-chemical potential) of the 2DEG and $N_{\rm sc}(\vec{r})$ the frozen surface charge density at the surfaces not covered by metallic gates and μ is the chemical potential of the 2DEG. We use the uppercase letter N to indicate volume densities (*e.g.* density of dopants $N_{\rm d}$) in m^{-3} and the lowercase letter n to indicate surface densities in m^{-2} (*e.g.* $n_{\rm s}$ the bulk 2DEG electronic density). Whenever possible, we will convert the volume densities to effective surface densities. For instance the dopant density $N_{\rm d}$ over a layer of thickness d_2 is equivalent to an effective 2D dopant density of $n_{\rm d} = N_{\rm d} d_2$. We use Dirichlet conditions $U(\vec{r}) = V_g - V_w$ at the gate-semiconductor interface. V_g is the applied voltage with respect to the grounded 2DEG. V_w is the work function of the gold/GaAs interface which for definiteness we take as $V_w \approx 0.75$ V. However, the actual value of V_w is irrelevant since any change of V_w will be compensated by a change in n_d to keep V_1 calibrated to the experiments.

To complete the theoretical model we must provide the relation between the density N of the 2DEG and the chemical potential μ.

This relation is defined by the integral up to μ of the system local density of states. Here, we approximate the local density of states to be equal to the bulk density of states of GaAs, ignoring the quantum fluctuations (Thomas-Fermi approximation). The integrated DOS equation for N thus reads:

$$\begin{aligned} N(\mu) &= \frac{(2m^*)^{3/2}}{3\pi^2\hbar^3}(\mu - E_b)^{3/2} \text{ for } \mu > E_b \\ N(\mu) &= 0 \text{ for } \mu \leq E_b \end{aligned} \tag{6.4}$$

where E_b is the position of the bottom of the conduction band in GaAs and m^* its effective mass. As discussed for V_w above, the actual value of E_b is irrelevant to the predictions of our model. Note that for the purpose of pinch-off voltage calculations, we could have used the constant density of state of a 2DEG, $n = m^*\mu/(\pi\hbar^2)$, instead of the three dimensional Eq.(6.4) and obtained the same results within our accuracy.

Figures 6.3a and 6.3b show a side view of the geometry used in the simulations of the “ungated" and “gated" QPC regions respectively (see Fig. 6.1c). We define: n_s as the 2DEG density underneath the ungated region, and n_g the 2DEG density underneath the gated region for $V_g = 0$. The stack is made of several layers of widths $\mathrm{d_i}$. The models for the gated and ungated regions are translationally invariant along the (x, y) plane, hence the problem reduces to a 1D simulation along the z direction. Figure 6.3c shows a side view of the geometry of the “narrow gate" region. Since it is invariant only along y, the problem reduces to a 2D simulation of the (x, z) plane. Finally, Fig.6.1a shows the geometry used for a QPC region. The simulations of the QPC regions are performed in 3D. A single set of parameters n_d, $n_{\rm sc}$ and the thicknesses $\mathrm{d_1} = 25$ nm, $\mathrm{d_2} = 65$ nm, $\mathrm{d_3} = 10$ nm and $\mathrm{d_4} = 10$ nm is used in the simulations of all the different regions.

The values of n_s and n_g at $V_g = 0$ is a complex function of the model parameters n_d, $n_{\rm sc}$, V_w, E_b and the $\mathrm{d_i}$. However, once these parameters are set (in our case, calibrated to the experiments), the density profile and the electric potential in the

2DEG are simply a function of n_s, n_g, V_g and the total distance $\mathrm{d} = \sum_{\mathrm{i}=1}^{4} \mathrm{d_i} = 110$ nm between the 2DEG and the gates. The approach we take in our model is to use n_s and n_g as effective parameters and ignore the large set of microscopic parameter. Note that in a typical 2DEG, n_d is roughly equal to 10 times $\mathrm{n_s}$, *i.e.* 90% of the dopant electrons go to the top surface and only 10% to the 2DEG [Buks *et al.* 1994b]. Furthermore, not all dopants necessarily get ionized. Hence a precise calculation of n_s (idem for n_g) requires a very precise knowledge of the dopant density and of the various energies level of the dopants and at the surface, c.f. Section 1.3.

In the simulations, we used a mesh with a discretization step smaller than 1 nm. We explicitly checked that the results are unaffected by the discretization within a precision better than 10 mV by performing several simulations with higher accuracy.

Figure 6.8(a) shows a typical 3D simulation of a QPC region (here device B6) at different gate voltages. The color map shows the electronic density around the central part of the device. At $V_g \gg V_3$, the density is only slightly decreased below the gates. At $V_g = -1.8V < V_3$, the region in between the two gates is fully depleted. Figure 6.8(b), shows the density versus V_g at two different points of interest. As expected, we find that the pinch-off, *i.e.* cutting the system into disconnected left and right parts, occurs when the central point $x = y = 0$ is depleted. Hence we take the corresponding V_g value as our calculated V_3. The typical potential profile observed in the simulations is almost flat in the 2DEG and abruptly rises in regions where the 2DEG has been depleted and cannot screen the gates. Plots of the behavior of the potential (at zero field but also in the quantum Hall regime) can be found in [Armagnat *et al.* 2019].

6.4 Comparison between Experimental and Simulation Pinch-off Voltages

6.4.1 Model Calibration using the $\mathrm{V_1}$ pinch-off of the gated regions

Our model has two free parameters. One is the dopant density n_d. It sets the 2DEG charge density underneath the "gated" region at $V_g = 0$ equal to n_g. The second is the surface charge density n_{sc}. It sets the 2DEG density underneath the "ungated" region, n_s, for a given $\mathrm{n_g}$. Our model allows for a spatially varying density even in the absence of applied voltage, *i.e.* $n_s \neq n_g$. As we shall see in section 6.5, there are multiple experimental evidences that point towards the fact that these two densities are in fact equal ($n_s = n_g$) due to "Fermi level pinning" and the model could be further simplified. Our calibration always leads to $n_s \approx n_g$ within 10% which is consistent with Fermi level pinning.

To calibrate our model, we use a two step process and two experimental values, V_1 and n_{bulk}. First, we vary n_d and calculate the pinch-off voltage V_1 in the "gated" region. We set n_d so that the simulated V_1 matches the experimental value. This sets n_g. In the second step, we vary n_{sc} and calculate the density n_s in the ungated region. We set n_{sc} so that n_s matches the experimental 2DEG bulk charge density

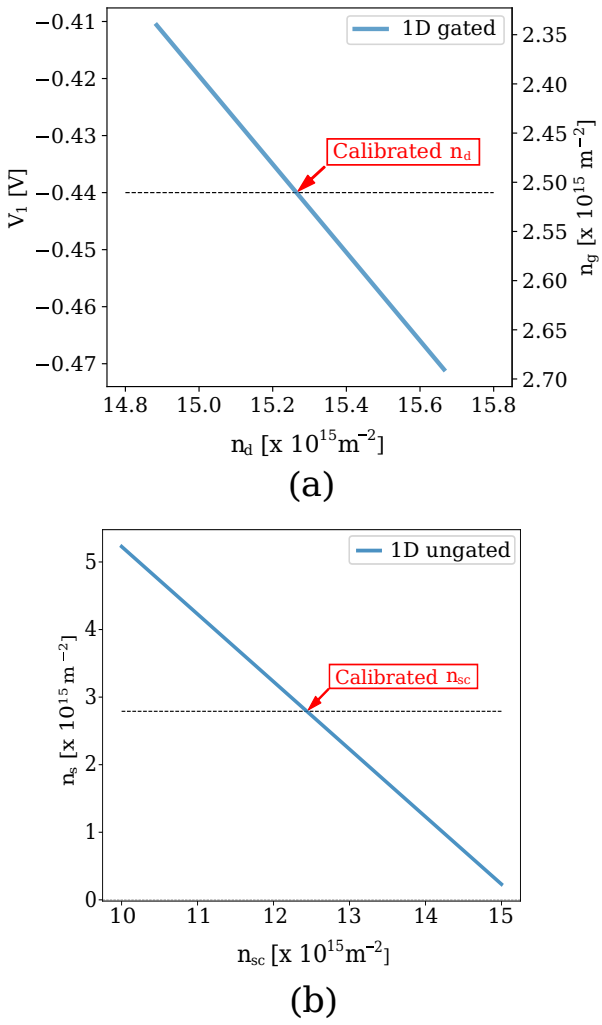


Figure 6.9: Illustration of the model calibration for sample A1a. Panel (a) shows the pinch-off V_1 (left axis) or equivalently n_g (right axis) calculated with the gated 1D model as a function of doping density n_d. The horizontal dashed line shows a typical experimental value of V_1. Panel (b) shows n_s as a function of the surface charge density n_{sc} using the 1D ungated model. The value of n_d is set to the intersection of the blue and dashed lines of panel (a). The horizontal dashed line shows the value of n_{bulk} and its intersection with the simulation result shows the calibrated value of n_{sc}.

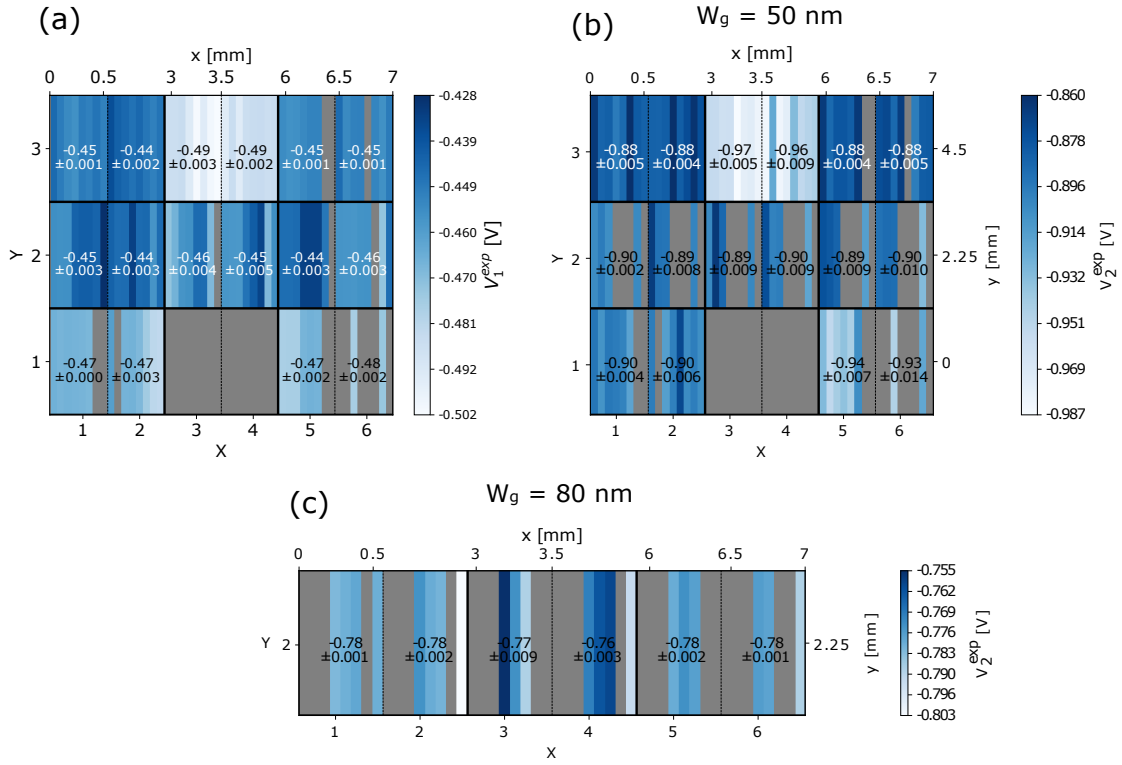


Figure 6.10: Colormaps of the variation of the experimental pinch-off values over the die. (a) ungated region values V_1^{exp}; (b) and (c) narrow gate region values V_2^{exp} with $W_{\text{g}} = 50$ nm and $W_{\text{g}} = 80$ nm respectively. Each thin stripe corresponds to a different sample. For each set of QPCs, the average value and the standard deviation between the different samples of the set is also shown. The top scale indicates the positions of the samples on the wafer in mm. The scale is not linear as consecutive pairs of samples are separated by 2 mm. No data is available in the gray regions (un-measured or distributed between (b) and (c).)

$n_{\text{bulk}} = 2.79 \times 10^{15}\text{m}^{-2}$. The calibration process is illustrated in Figure 6.9(a) (first step) and Fig. 6.9(b) (second step). It is repeated for each QPC.

Figure 6.10(a) shows the variations of V_1 for all the devices that have been measured. We find that the variations for QPCs within the same set are small, on the order of 0.5% (±2 mV). Therefore, using a unique average value of n_d to model a given set would give identical result with respect to the QPC per QPC calibration. However, the variations of V_1 for QPCs of different sets are larger - of the order of 10% (40 mV). They are of the same order of magnitude as typical variations observed between different cooldowns. They imply the presence of significant variations over large distances of n_g. We suspect that similar variations of n_s are also present, see the discussion in section 6.5. In the dies X = 3 and 4 with Y = 3, the calibration with $V_1 \approx -0.49$ V gives $n_g = n_s$ while in the other samples the two densities differ by less than 10%.

6.4.2 Simulations of the QPC regions pinch-off voltages V_3

After calibrating the model, we performed 3D simulations of the "QPC" region to calculate the pinch-off voltage V_3. Figure 6.2 shows the predicted (dashed lines) and measured (full lines) pinch-off voltages. We compare V_3 as a function of L (top panel, A samples), R (middle panel, B samples) and L (bottom panel, C samples). Figure 6.2 highlights the main results of this article. It shows that the simulations correctly capture the pinch-off voltages. The main features of interest of Fig. 6.2 are:

(P1) Overall the simulations predict the pinch-off voltages quantitatively with a precision of the order of 10%.

(P2) There are significant experimental V_3 variations in between QPCs with the same nominal characteristics. They are also of the order of 10%. For instance the values of V_3 observed for the four A2 samples (A2a, A2b, A2c and A2d) range from -2.2 V to -1.8 V, while the numerics predict a V_3 close to -1.8 V. We also observed similar variations of the values of V_3 (of the order of 0.1 V) on the same QPCs between different cooldown. Hence, the accuracy of the predictions is as good as the level of reproducibility of the experiments. Getting beyond this accuracy would involve a local in-situ calibration of the model so that any spatial variations of n_s, n_g within the wafer would be accounted for. One could, for instance, include an additional QPC in the device, close to the active part of interest, and use the associated V_3 value to calibrate the modeling with the actual local electronic density.

(P3) The V_3 dependence on the QPC nominal characteristics L, R and W are correctly reproduced qualitatively.

(P4) The predicted V_3 is almost always smaller (in absolute value) than the experimental one by an offset of the order of 0.1–0.2 V. This indicates that our calibration slightly underestimates the value of the electronic density by 5–10%. We attribute this fact to disorder as explained in section 6.5.3.

Figure 6.11 shows the V_3 data on samples with lengths $1\ \mu \leq L \leq 50\ \mu$m. For such long samples, the simulations predict that V_3 should not depend on L. This trend is already observed in Fig. 6.2 for lengths exceeding 1 μm. Indeed, the largest length scale in the problem is the distance between the 2DEG and the gate, *i.e.* d $\approx$ 110 nm. When $L \gg$ d, V_3 no longer depends on L. In practice, we have found that for $L \geq 5$d, one has already reached the infinite L limit in the simulations. Hence, the simulations for devices with $L \geq 5$d are done by supposing $L = \infty$, *i.e.* a system invariant by translation along the y direction. We have used two different calibrations of the model: the same one as described in the preceding section (black) and a different one where we calibrate n_g with the experimental V_1 and then set $n_s = n_g$. Both simulations give similar results and fail to capture the main experimental observation of Fig. 6.11 which is,

(P5) $V_3(L)$ has a large variation of $\approx +600$ mV as the sample length goes from $L = 1\ \mu$m to $L = 50\ \mu$m (from -3.34 V at L = 1 μm to -2.72 V at L = 50 μm for $W_{\mathrm{QPC}} = 750$ nm) .

Property (P5) cannot be explained by the model that we have used so far. In

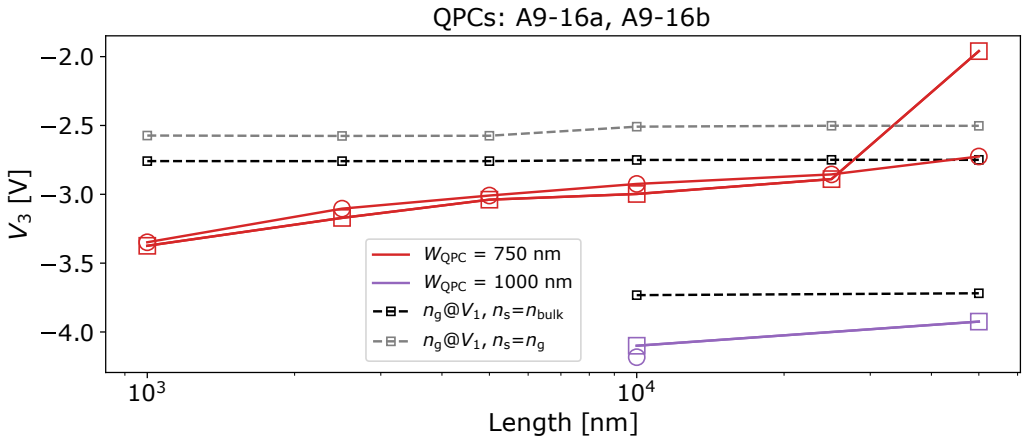


Figure 6.11: Experimental results of V_3 for the large A designs. The dashed lines correspond to the simulation for an infinitely long sample. The different symbols correspond to sample a: (X_1Y_2 , squares) and b (X_5Y_2, circles). Two different calibrations have been used in the simulations: the one described in section 6.3 (black dashed line) and a secondary calibration that enforces $n_s = n_g$ (gray dashed line).

order to account for (P5), one must take into account the smooth density fluctuations that take place on long scales. Indeed, in the presence of spatial variations of the density along the x direction, the pinch-off V_3 is determined by the position in x where the density is smallest. A model analyzing semi-quantitatively the role of the disorder will be presented in section 6.5.3.

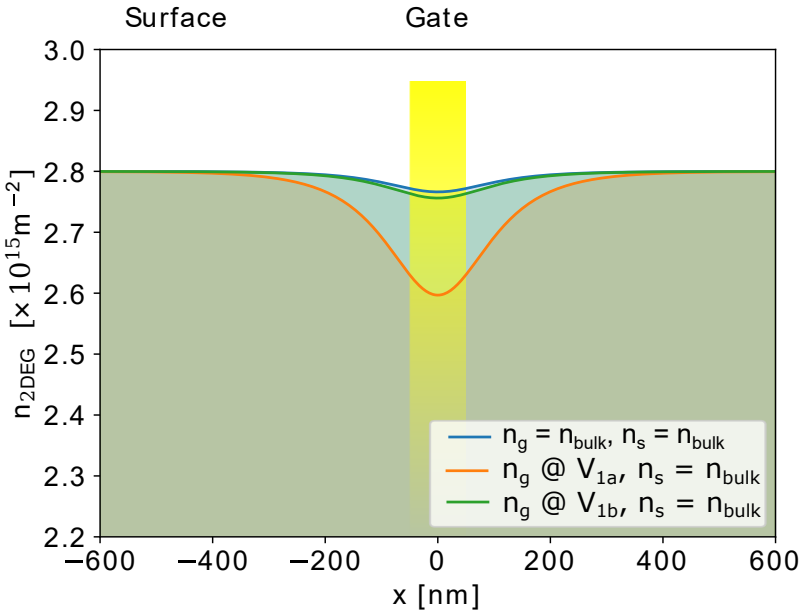
6.4.3 Simulations of the narrow gate region pinch-off voltages V_2

We now turn to the simulations of the "narrow gate" region. They correspond to the very long (> 20 μm) but thin (50 nm wide) gate. Figure 6.12(a) shows a typical simulation of the electronic density versus x at zero applied gate voltage. The different curves correspond to different densities of surface charge and dopants. Specifically, we have used different calibrations for the density n_s far away from the gate and the density n_g under a (wide) gate. While the simulated 2DEG density varies below the gate, this variation is smaller than 5% which corresponds to the variation of V_1 that we have observed on different samples. It follows that our findings are fully compatible with a uniform density at zero voltage.

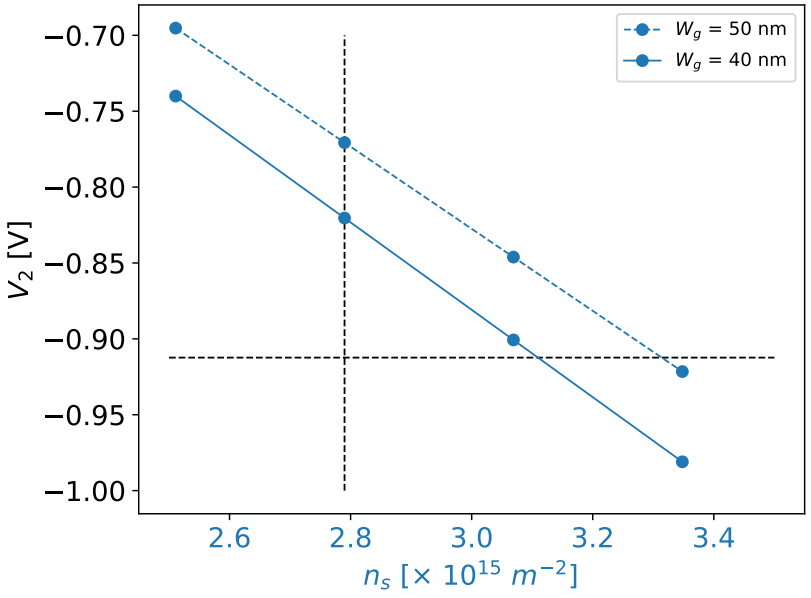
The main observation we make for the voltage V_2 is that our predictions are significantly lower than the experimental data for both values of width W_g, see Table 6.1. More precisely:

(P6) The simulations systematically underestimate the magnitude of V_2 by $\approx$ 0.15 V (20%).

The error in (P6) is the largest discrepancy we have observed between the sim-



(a)



(b)

Figure 6.12: (a) Simulated electronic density profile as a function of x at $V_g = 0$, in the narrow gate region with $W_g = 50$ nm. The orange and green 2DEG profiles are the simulation results obtained from the standard calibration for QPC B2a and B2b. The blue line uses a calibration where $n_s = n_g = n_{\text{bulk}}$. (b) V_2 as a function of $n_s = n_g$ for two different gate widths W = 50 nm (dashed blue line) and W = 40 nm (full blue lines). The vertical line shows $n_s = n_{\text{bulk}}$ and horizontal line shows the experimental value of V_2.

	V_2 exp.	σ_{V_2} exp.	V_2 sim. mean
50 nm	-0.90	0.032	-0.73
80 nm	-0.77	0.010	-0.63

Table 6.1: Comparison between experimental (exp) and simulated (sim) values of V_2 for the two different widths W_g. σ_{V_2} is the standard deviation of V_2 between different QPC. We observe a systematic deviation of ≈ 0.15V (20%) between the simulation and the experiments.

ulations and the experiments. We identify four possible origins for this discrepancy. (1) The bulk value n_s is higher than the one we used. (2) The width W_g is narrower than what is drawn in the design. Gate fabrication uses standard lithographic technique with e-beam insolation of a resist, chemical lift-off of the resist followed by metal deposition and chemical lift-off of the residual resist. This process should have an accuracy better than 10 nm in the width of the gates. (3) The width W_g fluctuates along the gate due to lithography accidents. (4) There are density fluctuations of the 2DEG due to disorder.

Figure 6.12(b) shows the predicted value of V_2 as a function of the 2DEG density n_s assuming a uniform 2DEG density at $V_g = 0$ ($n_s = n_g$). The vertical line corresponds to the nominal value $n_s = 2.8 \cdot 10^{15}\text{m}^{-2}$. The horizontal line is the measured value of V_2. We see that to obtain the experimental value of V_2 for $W_g = 50$ nm, one needs $n_s = 3.4 \cdot 10^{15}$ m^{-2} which is unreasonably high (This 21.5% higher than the nominal value while typical density variations inside a wafer are in the 5–10% range). Hence, we can rule out (1) as the origin of (P6). The straight blue line in Figure 6.12(b) shows the value of V_2 obtained when one reduces the width of the gate by 20%, *i.e.* $W_g = 40$ nm. We see that this is not sufficient to reproduce the experimental data and larger variations of W_g would be visible on the SEM images. In contrast the SEM images indicate a width that is slightly larger than 50 nm. Hence, we rule out (2). Finally, we do not observe sample to sample variations of V_2 and the SEM pictures do not show fluctuations of the width W_g along the gate. Hence we rule out (3).

The last scenario (4) corresponds to smooth spatial fluctuations of the density inside the sample. This could be due to *e.g.* doping density or background doping fluctuations [Zhou *et al.* 2015, Qian *et al.* 2017, Chung *et al.* 2019]. Indeed, if the electronic density varies underneath the 20 μm long gate, the corresponding V_2 pinch-off will be given by the region of *largest* density. This interpretation is fully consistent with the observation of the significantly large sample to sample fluctuations of V_3. In section 6.5.3 we perform a systematic analysis of the effect of long range disorder on V_1, V_2 and V_3. We find that a 5–10% density fluctuation consistently explain (P2), (P5) and (P6).

6.5 Critical discussion of the modeling

In this section we discuss various aspects of the modeling in more detail. We emphasize again that we refrained from the particularly difficult goal of trying to predict the bulk density of the device. That is, to develop a model capturing the microscopic details along the 1D z-direction of our samples. While the corresponding physics is well understood and has been studied rather extensively, the resulting electronic density depends on many parameters which are often poorly known. These microscopic parameters include the density of dopants, the fraction of dopants that are ionized (or equivalently the precise dopant ionization energies - including the so called DX centers), the residual doping in the bulk of the wafer, the density of surface charges (or equivalently the precise value of the binding energy of the surface states), the workfunction of the metals used in the electrostatic gate with respect to GaAs, the values of the band offsets, the effective masses, the relative dielectric constants of the different materials [Buks *et al.* 1994a, Chung *et al.* 2017, Davies 1997, Weisbuch & Vinter 1991] etc. Making quantitative predictive simulations with so many unknown parameters that depend on the growth condition of the wafer is very challenging. It also serves a very different purpose, more related to wafer characterization than to the understanding of the devices made out of it.

Our goal instead is to be able to predict the spatial variations along the 2D x- and y-directions. We use experimental measurements to tabulate the result of the interplay between all the above mentioned parameters. Indeed, and this is a very important point, while this interplay is quite subtle at room temperature, at sub-Kelvin temperatures on the other hand all the possible source of charges (surface, dopants) are essentially frozen. Hence, while they do contribute to the electronic density, their effect boils down to a contribution to the 2DEG electronic density that can be measured independently through *e.g.* Hall measurements. The fact that the charge sources are frozen, a well established experimental fact, coupled to the linearity of the Poisson equation, means that to predict the effect of the gate voltages on the 2DEG density one only needs: (i) the distance of the 2DEG with respect to the gates and (ii) the low temperature 2DEG density profile in the xy plane at zero applied gate voltage. This is precisely what we are trying to capture in our simulations. Below we discuss how the choices of (i) and (ii) made in our modeling affect the results.

6.5.1 Role of quantum capacitance and quantum fluctuations on the electronic density

Let us discuss a 1D minimum model to discuss the electronic density in the "ungated" (bulk) or "gated" regions. These regions are sufficiently large so that the 2DEG can be considered far from the gate boundaries. The spatial variation of the 2DEG density in the xy plane can thus be ignored, assuming no disorder. We are left with a, possibly complex, 1D problem along the z-direction. We describe the 2DEG

by its 2D density of states ρ. We suppose the vertical distance to the gate (at voltage V_g) to be d. Lastly, we assume there is an arbitrary distribution of doping charges $n_d(z)$. It includes the ionized dopants, the surface charge and any other frozen charge that might be present in the system. The 2DEG density n_s is given by $n_s = \rho\mu$ where μ is the chemical potential of the 2DEG. Assuming without loss of generality, that the 2DEG is grounded, the electrochemical potential vanishes so that $\mu - eU(z = 0) = 0$. The model reduces to solving the Poisson equation,

$$\frac{\partial^2}{\partial z^2}U(z) = \frac{e}{\varepsilon}n_{\mathrm{d}}(z) \tag{6.5}$$

with the boundary conditions $\mathrm{U(d)} = \mathrm{V_g}$ and $\partial_{\mathrm{z}}\mathrm{U}(0) = (\mathrm{e}^2\rho/\varepsilon)\mathrm{U}(0)$. This equation being linear, its solution can be written as a linear combination of two terms $\mathrm{U(z)} = \mathrm{U(z, V_g = 0)} + \mathrm{V_g U(z, V_g = 1)}$. We thus arrive at,

$$\left(\frac{1}{e^2\rho} + \frac{d}{\varepsilon}\right) en_g = V_1 - V_g \tag{6.6}$$

where the parameter V_1 is the pinch-off voltage. It corresponds to the contribution of $n_d(z)$ to the electronic density. In such a simple model, V_1 could be expressed explicitly in terms of $n_d(z)$. However, we will refrain from doing so and take it as an experimentally measurable parameter. Eq.(6.6) provides a direct conversion relation between 2DEG density to voltages. For a distance of d = 100 nm and using $\varepsilon = 12\varepsilon_0$, the density is $n_g = 6.6 \cdot 10^{15}\mathrm{m}^{-2}\mathrm{V}_1$. For our stack, *i.e.* a bulk density of $n_s = 2.8 \cdot 10^{15}$ m^{-2} and d = 110 nm, we calculate V_1 = (2.8/6.6)*(110/100) = 0.46 V. The latter is the predicted pinch-off voltage in the "gated" region. This simple calculation actually matches the measured value of V_1.

In the calculation above we have neglected the contribution from the density of states. Indeed, the contribution of the $1/(e^2\rho)$ term, *i.e.* the inverse of the quantum capacitance, is for most devices negligible compared to the inverse of the geometrical capacitance d/ε, see Chapter 2. For the QPCs studied in this paper, the quantum capacitance term adds a correction of 2% to the voltage pinch off. The latter is estimated using the effective mass approximation and assuming only the first subband as occupied when calculating ρ. That is $\rho = \frac{m^*}{\hbar^2\pi}$ with $m^* \approx 0.067m_0$ for GaAs. A 2% correction is smaller than our experimental resolution. We conclude that the various pinch-off voltages are almost entirely controlled by the electrostatics of the problem, *i.e.* the geometrical capacitance. They are enough to characterize the distribution of charges in the 2DEG.

The value d = 110 nm is the physical distance between the electrostatic gate and the GaAs/AlGaAs interface. In principle one should take into account the finite width of the 2DEG which is of the order of 10 nm. This effect is partially taken into account in the simulations at the Thomas-Fermi level, but would be more pronounced if the quantum fluctuations along the z-direction were included. We have performed various full self-consistent 1D Schrodinger-quantum simulations (not shown). They show a small correction of the final width of the 2DEG of less than 1%.

6.5.2 Fermi level pinning of the dopants at room temperature

In our model we have assumed that, at $V_g = 0$ the density underneath a gate could be different from the density away from a gate. This is reflected in the presence of the two parameters n_s (density in the ungated region) and n_g (density in the gated region) that can *a priori* take distinct values $n_s \neq n_g$. We have found *a posteriori* that the experiments are best fitted by $n_s \approx n_g$ indicating that the $V_g = 0$ density has small spatial variations inside a given sample. Here, we discuss the phenomena of "Fermi level pinning" in the dopant region at room temperature, i.e. the possibility that –at room temperature– the dopant layer behaves essentially as a (metallic like) equipotential. The existence of Fermi level pinning in an actual stack requires a sufficiently high concentration of dopant and sufficiently low disorder in the dopant region for the dopants to remain on the metallic side of the metal-insulator transition at room temperature (however so slightly). The presence of Fermi level pinning would imply $n_s = n_g$. We argue that there are strong experimental evidences for Fermi level pinning in our samples. This fact can be used to reduce the model to a single fitting parameter.

A homogeneous dopant distribution leads naturally to $n_s \neq n_g$ unless the surface charge density accidentally matches the contribution coming from the work-function at the gate-GaAs interface. An opposite hypothesis — Fermi level pinning — is that, at room temperature, the *electric potential* (not the ionized dopant density) is constant inside the dopant layer. Fermi level pinning happens when the charges in the dopant layer are sufficiently mobile to form a metallic-like equipotential. The associated charge distribution (now spatially dependent) gets frozen upon cooling the sample to low temperature. By construction, Fermi level pinning implies $n_s = n_g$ since the sources of spatial inhomogeneities (the gates) are situated above the dopant region, hence screened. Below, we list experimental evidences for the presence of Fermi level pinning in GaAs/AlGaAs heterostructures. These evidences are very strong for *some* heterostructures (in particular in the case of "delta doping" where the dopant concentration is very high hence more likely to form a band) but it is not certain that Fermi level pinning is present in *all* of them.

- In [Buks *et al.* 1994a, Buks *et al.* 1994b], the field effect of HEMTs made in these heterostructures is shown to disappear at high temperature, indicating that the dopant layer screens the effect of the gate, c.f. Figure 6.13.

- Fermi level pinning is also the natural explanation for the hysteretic effect known as "bias cooling" [Buks *et al.* 1994b]: when one applies a voltage on an electrostatic gate during the cooling of the sample, one observes that the low temperature current versus gate voltage characteristics gets shifted horizontally by the same amount, c.f. Figure 6.14. For instance if a sample normally pinches at -1.5 V for a regular cooling, a $+1$ V bias cooling will make it pinch at -0.5 V. This is a strong indication that the voltage applied at room temperature did not affect the electronic density. Upon cooling, the dopants get frozen and must be considered as a fixed charge density. Hence, to deplete the

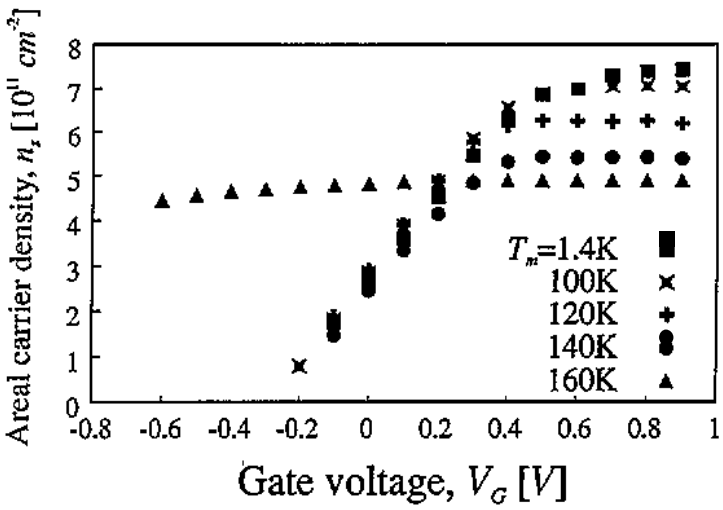


Figure 6.13: Taken from [Buks *et al.* 1994a]. Charge density at the 2DEG of an AlGaAs/GaAs heterostructure as a function of gate voltage (MIS junction geometry, delta Si doping) and device temperature. For a temperature higher than $T_m = 160K$ the charge density is constant. This indicates the absence of field effect at high temperatures. This is due to fermi level pinning at the dopants. Even at lower temperatures the screening of the gates due to the donor charge persists until a low enough V_g is reached (saturation regime).

gas, what matters is the *variation* of the voltage with respect to the value used during the cooling, not the absolute value of the voltage. Bias cooling is often used by experimentalists to reduce the pinch-off voltage and avoid leakages. It has been observed repeatedly including in wafers nominally identical to the one used in the experiments presented in this article.

- There are multiple experimental evidences showing that using different metals for the electrostatic gate, say gold and aluminum, give devices with very similar properties in terms of pinch-off voltages [Pierre *et al.* 2022]. This is an additional experimental evidence for Fermi level pinning. Indeed, different gate materials, such as gold and aluminum, have very different work functions of the order of 0.8 V. In the absence of Fermi level pinning, one would get very different pinch-off values as well as signature of a strongly varying spatial distribution of the 2DEG density (visible in *e.g.* quantum Hall effect experiments). None of these effects are observed experimentally.

- We have found that the best fit to our model implies $n_\mathrm{s} \approx n_\mathrm{g}$ within 5% which is unlikely to happen accidentally.

We conclude that Fermi level pinning of the dopants is very likely present in our samples. This could be used to further simplify our model to a single parameter

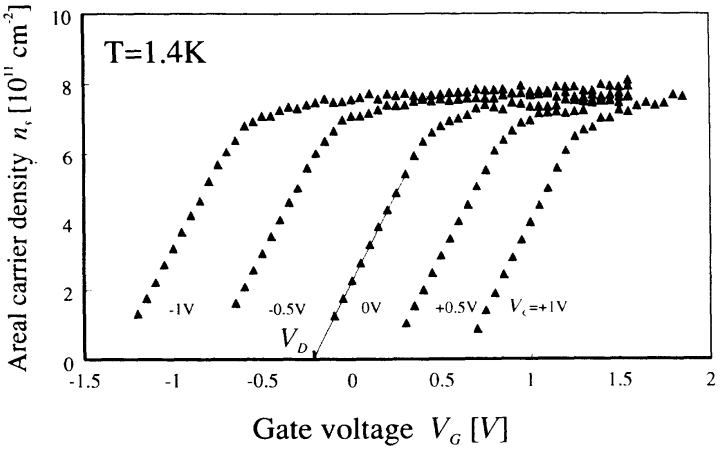


Figure 6.14: Taken from [Buks *et al.* 1994b]. Illustration of bias cooling. Charge density at the 2DEG of an AlGaAs/GaAs heterostructure as a function of gate voltage (MIS junction geometry, delta Si doping) and cooling gate voltage V_c. First V_c is applied by the gate during cooldown, as the sample temperature goes from above to bellow the donor freeze-out temperature (where the donor charge freezes). Then, at $T = 1.4K$, the charge as a function of V_g is measured. Notice the saturation regime, where the gate effect is screened, and the linear regime where the charge in the dopant layer is constant.

$n_s = n_g$ that can be calibrated in situ using pinch-off voltage.

We note that the phenomena of Fermi-level pinning could also be discussed with respect to the *surface states*. Our calculations show that such an effect, if present, could not account for phenomena such as the hysteresis observed in bias cooling but at most to half of the observed effect. The presence of Fermi-level pinning of the surface states would, however, further contribute to enforce $n_s = n_g$.

6.5.3 Long range disorder and density fluctuations.

We end this article with a discussion of the role of disorder in the system. So far, all the analysis has been done assuming a perfect 2DEG whose spatial density profile is only affected by the electrostatic gates. Despite the very high mobility of the 2DEG, there remains some disorder in the system. There are several types of disorder that can be present in the system[Buks *et al.* 1994a, Buks *et al.* 1994b] including inhomogeneities in the dopant density, interface roughness or background impurities. Note that some types of disorder such as interface roughness may very well affect the conductance. However as interface roughness varies mostly on short (atomic) scale it is unlikely to significantly affect the electronic density unless the disorder is very strong. This type of disorder can be ignored for the purpose of this discussion. Indeed, the goal of this section is to understand the effect of disorder on the pinch-off properties of the device. Hence we focus on the slowly varying

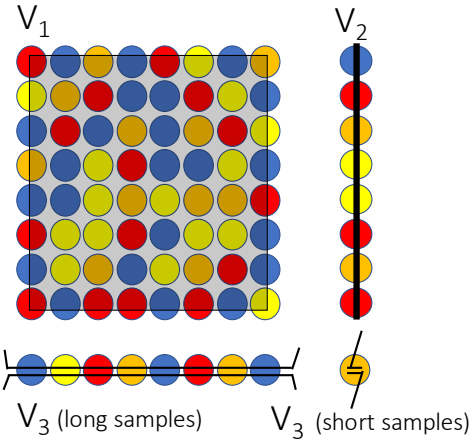


Figure 6.15: Schematic of the percolation model used to explain the effect of the smooth long range disorder. Each circle corresponds to a region of size ξ with a random electronic density (symbolized by different colors). As one increases the gate voltage towards negative values, the density decreases everywhere. The pinch-off is obtained when there is no path left with finite density to go from left to right. The fluctuations of density manifest themselves differently for the different pinch-off: 2D percolation problem for V_1 (upper left), many regions in parallel for V_2 (upper right) or in series for V_3 of long samples (lower left) while the fluctuations of density induce fluctuations of V_3 for short samples (lower right).

part of the disorder on scale larger than the Fermi wave length. We attribute this disorder mostly to variations of dopant density, but other sources could be present as well without affecting the discussion that follows. Recent experimental works that explicitly study the spatial variation of the electronic density include [Zhou *et al.* 2015, Qian *et al.* 2017, Chung *et al.* 2019]

We construct a simple percolation model to discuss the effect of long range disorder on the three thresholds V_1, V_2 and V_3. A schematic of the model is shown in Fig.6.15. We will see that this simple model can account for all the systematic discrepancies observed between the simulations and the experiments, at least qualitatively. Let's consider a $L_x \times L_y$ 2DEG sample. We suppose that the density is slowly varying on a typical length scale ξ, the disorder correlation length. The 2DEG can thus be considered as made of $L_x/\xi \times L_y/\xi$ small samples (the circles in Fig.6.15), hereafter referred to as "cells". Typically, we expect ξ to be of the order of a few hundred nanometers for a disorder due to dopant density fluctuations. Each cell has a constant density $\mathrm{n_{ij}}$ with $\mathrm{i} \in \{1....L_x/\xi\}$ and $\mathrm{j} \in \{1....L_y/\xi\}$. The value of the density $\mathrm{n_{ij}}$ in cell (i,j) is a random variable of mean n_g and variance σ_{g}^2, independent from the density in other cells. For definiteness, we suppose that the associated probability density is flat,

$$P(n < n_{\mathrm{ij}} < n + dn) = \frac{1}{2\sqrt{3}\sigma_g}\theta(n - n_g - \sqrt{3}\sigma_g)\theta(n_g + \sqrt{3}\sigma_g - n)dn \tag{6.7}$$

where $\theta(x)$ is the Heaviside function. Last, we suppose that the pinch-off voltage on each cell (i,j) is simply proportional to $\mathrm{n_{ij}}$ as found in the minimal model of section 6.5.1.

Let us first examine the implication of this model for the threshold V_1. In the absence of density fluctuations, the conductance through the gated region vanishes when the gate voltage depletes the 2DEG entirely. In presence of fluctuations, however, depleting only a fraction of the 2DEG cells suffice. That is, if the fraction p of remaining cells with non zero density is bellow the percolation threshold $p_c \approx 0.6$ of the 2D square lattice of cells, then the conductance vanishes. We introduce the probability $P(\mathrm{n_{ij}} \geq \mathrm{n})$ for a cell to have a density larger than $n = n_g + \delta n$,

$$P(n_{ij} \geq n_g + \delta n) = \frac{1}{2} - \frac{\delta n}{2\sqrt{3}\sigma_g} \tag{6.8}$$

for $|\delta n| \leq \sqrt{3}\sigma_g$. The percolation threshold corresponds to $P(n_{\mathrm{ij}} \geq n_g + \delta n) = p_c$. From the latter we obtain the variation δV_1 induced by the density fluctuations,

$$\frac{\delta V_1}{|V_1(\sigma_g = 0)|} = (2p_c - 1)\sqrt{3}\frac{\sigma_g}{n_g} \approx 0.34\frac{\sigma_g}{n_g} \tag{6.9}$$

Eq.(6.9) leads to a positive variation of V_1. Since $V_1 < 0$, it leads to a decrease of V_1 in absolute value. Conversely, not taking the density fluctuations into account when estimating V_1 leads us to underestimate the density n_g.

Next, we examine the implications of the disorder model for the threshold V_2. In the narrow gate region, L_x is smaller than the correlation length ξ. Therefore, the current travels through L_y/ξ cells in parallel. The pinch-off is reached when the voltage is sufficiently negative to deplete *all* the cells. Hence V_2 corresponds to the voltage needed to deplete the cell with largest density. The probability for the cell with highest density to have a density smaller than $n_g + \delta n$ is given by $[1/2 + \delta n/(2\sqrt{3}\sigma_g)]^{L_y/\xi}$. It corresponds to the probability that all cells have a density smaller than $n_g + \delta n$. For $L_y \approx 50\mu\text{m} \gg \xi$ this probability is strongly peaked around $\delta n = \sqrt{3}\sigma_g$, from which we obtain the variation δV_2 induced by the density fluctuations,

$$\frac{\delta V_2}{|V_2(\sigma_g = 0)|} = -\sqrt{3}\frac{\sigma_g}{n_g} \approx -1.7\frac{\sigma_g}{n_g} \tag{6.10}$$

The effect of disorder on V_2 is around 5 times larger than on V_1. It is also of opposite sign. We can calculate the standard deviation σ_{V_2} of V_2 due to sample to sample variations. We find,

$$\frac{\sigma_{V_2}}{|V_2(\sigma_g = 0)|} = 2\sqrt{3}\frac{\xi}{L_y}\frac{\sigma_g}{n_g} \tag{6.11}$$

Last, we look at the influence of disorder on V_3 in two different limits. For the small samples $L \leq 2$ μm, the QPC region corresponds essentially to a single cell. In that limit, the fluctuations σ_{V_3} of the threshold V_3 are simply given by the density fluctuations of a single cell and,

$$\frac{\sigma_{V_3}}{|V_3(\sigma_g = 0)|} = \frac{\sigma_g}{n_g}. \tag{6.12}$$

A second interesting limit corresponds to the very long samples 10 μm $\leq$ L $\leq$ 50 μm. These samples correspond to the dual situation to V_2: the different L_x/ξ cells are in parallel instead of being in series. Therefore the pinch-off is limited by the cell that has the smallest density. The probability for the smallest density to be larger than $n_g + \delta n$ is given by $[1/2 - \delta n/(2\sqrt{3}\sigma_g)]^{L_y/\xi}$. For $L_x \gg \xi$, we get,

$$\frac{\delta V_3}{|V_3(\sigma_g = 0)|} = \sqrt{3}\frac{\sigma_g}{n_g} \approx 1.7\frac{\sigma_g}{n_g} \tag{6.13}$$

i.e. the fluctuations make it easier to pinch-off a long wire. This ends our analysis. Note that the precise value of the prefactors in Eqs.(6.9), (6.10),(6.11) and (6.12) depend on the choice of distribution Eq.(6.7) so that the percolation model should be used for trends, not precised comparisons.

6.5.4 Comparison between the experiments and the percolation model

Let's now go back to the experimental data and show that the above percolation model accounts for all the imperfections of the no-disorder model at a semi-quantitative level. The largest discrepancy between our predictions and the ex-

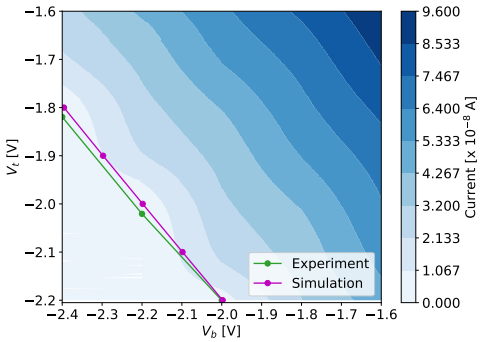


Figure 6.16: Colormap of the current versus the top (V_t) and bottom (V_b) gates that were biased separately for QPC A1a. Green (blue) lines show the experimental (simulated) pinch-off voltage. $V_3 \approx -2.1$ V corresponds to $V_t = V_b$ and is used to calibrate the simulations in a single parameter model $\mathrm{n_g} = \mathrm{n_s} \approx \mathrm{n_{bulk}} + 5\%$

periments is the one of V_2 (property P6). Indeed, the simulations for perfect systems systematically show values of V_2 that are around 20% smaller (in absolute value) than what is observed experimentally. To account for this $\delta V_2/|V_2| \approx 0.2$, Eq.(6.10) implies that density fluctuations with $\sigma_g/n_g \approx 0.12$ occur in the system. Density fluctuations of 12% is compatible with what is commonly believed by the community for this system if somewhat large [Zhou *et al.* 2015, Qian *et al.* 2017, Chung *et al.* 2019]. It is also compatible with what we have observed on larger scales on the fluctuations of $\mathrm{V_1}$ (see Fig. 6.10) . Eq.(6.12) then implies that the sample to sample variations of V_3 are also of the order of 12%. While we do not have enough statistics to properly estimate the variance of V_3, a rough estimate from our data is of the order of 6% (property P2, see Fig.6.2).

Eq.(6.9) then predicts a correction to V_1 of 4%. Taking that correction into account in our calibration would bring all our predictions in Fig.6.2 down by 4% (80 mV). This would significantly improve the match between experiments and simulations, see the discussion of property (P4). Another possible source of error in V_1 stems from an imprecision when extracting the experimental value from the conductance curve. Near V_1 the “gated" region contribution to the overall conductance is much smaller than that of the “narrow gate" region. The latter contribution thus obscures the conductance due to the “gated" region. This adds an error to the extracted value of V_1 that is not accounted by our theoretical model.

Using Eq.(6.11), the small observed fluctuations $\sigma_{V_2} \approx 5$ mV imply a correlation length $\xi \approx$ 1-2 μm. This is fully compatible with our expectations.

Last, Eq.(6.13) predicts that when going from small to large values of L (with respect to ξ), V_3 must increase by 0.6V ($\delta V_3/|V_3| \approx 0.2$) which is indeed what is observed experimentally (see property P5).

Overall, the above analysis is fully consistent with smooth density variations being the current bottleneck in our quantitative predictions of pinch-off voltages.

To go beyond this limitation, one needs to incorporate information about the *local* electronic density within the model. An example of such a procedure is shown in Fig. 6.16 where we use the experimental value of V_3 to calibrate a *single parameter* model with $n_s = n_g$ (see the discussion of section 6.5.2). Figure 6.16 shows the pinch-off "phase diagram" as a function of the bottom V_b and top V_t gate voltages when these two gates are biased independently. We find that the case $V_b \neq V_t$ is quantitatively predicted with an accuracy better than 1%, *i.e.* significantly improved with respect to a global calibration.

6.6 Conclusion

In this Chapter we have developed a minimal model capturing the electrostatics of QPCs. We validated our model by comparing its predictions to 110 different QPCs made in 48 different designs measured by the experimental group of Christopher Bauerle at Institut Néel. In this work we have restrained ourselves to studying the pinch-off voltages of their QPCs. The large and diverse dataset allowed us to develop a robust calibration protocol for the parameters in the numerical model. The model focused on reconstructing the charge distribution within the device. The calculated pinch off voltages are accurate within $5-10\%$ when using a single global calibration of the modeling. The limiting factor of our accuracy seems to be slow spatial variations (disorder) of the electronic density. Multiple aspects of the experiments point out to charge fluctuations on the order of $\pm 5-10\%$. This is likely due to inhomogeneities in the dopant layer, leading to fluctuations in the charge density at the 2DEG on a μm scale. In fact, the presence of disorder in the dopant layer is known. For instance, its effect on the transport through QPCs can be seen in scanning gate microscopy measurements [Topinka *et al.* 2001, Percebois & Weinmann 2021]. As a continuation of the work presented here, in the next chapter we show some of our recent results on reconstructing the disorder potential at the 2DEG from scanning gate microscopy measurements [Percebois *et al.* 2023].

Regarding future work. To improve the predictive power of the simulations, the simplest solution is to design samples small enough (≤ 2 μm) s.t the calibration of the model can be done in situ, e.g. using the value of V_3. For larger samples, one may design them so that the density in different parts of the device may be calibrated separately using independent gates. Finally, we have only used the pinch off voltages of the courant vs voltage behavior of the QPCs. It would be interesting to analyze how the conductance plateaus depend on the gate geometry and how well our model manages to capture them.

Chapter 7

PESCADO and Neural Networks : Extracting the disorder potential from scanning gate microscopy

Disorder in the electrostatic potential at the conducting region of nanoelectronic devices plays a significant role on how the conducting electrons flow. The microscopic origin of such disorder is complex and device dependent. This renders difficult and unpractical to simulate the disorder from microscopic models such as the ones in Chapter 6. A more straightforward way to obtain a sample disordered potential is to extract it from experiments. For samples whose conducting region are buried beneath layers of dielectrics there is no direct experimental method to obtain the disorder potential. However it has been shown [Topinka *et al.* 2001, Jura *et al.* 2007, Fratus *et al.* 2019, Percebois & Weinmann 2021] that SGM measurements contain information about the disordered potential where the electrons flow. In a collaboration with G.Percebois and D.Weinmann at Strasbourg we have developed a technique capable of using Neural Networks to extract the disorder configuration from SGM data. The work was led by the group at Strasbourg and has been published as :

> Gaëtan J. Percebois, Antonio Lacerda-Santos, Boris Brun, Benoit Hackens, Xavier Waintal, Dietmar Weinmann. **Reconstructing the potential configuration in a high-mobility semiconductor heterostructure with scanning gate microscopy.** SciPost Phys. 15, 242 (2023) · published 15 December 2023

This chapter is organized as follows. First in Section 7.1 we describe the SGM technique. Then in Section 7.2 we discuss the Neural Network approach developed by the G.Percebois from the Strasbourg group. Afterwards in Section 7.3 we describe the experimental sample and data of [Iordanescu *et al.* 2020]. Then in Section 7.4 we describe the technique we developed to calculate a large dataset of SGM maps using *PESCADO* . Finally, in Section 7.5 we show the predicted disorder potential and validate it using the experimental data of [Iordanescu *et al.* 2020]. We worked closely with B.Brun, one of the researchers in [Iordanescu *et al.* 2020], to understand their data and compare it with the Neural Network predictions. The machine learning aspect of the work was mainly done by the Strasbourg group. Here at Grenoble we worked mostly on the modeling of the device electrostatics.

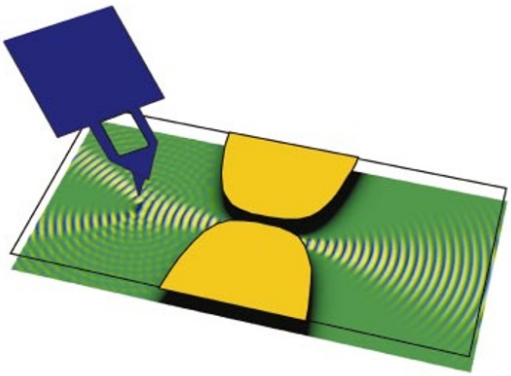
7.1 Scanning gate microscopy

SGM is an experimental technique for imaging the electron flow in devices whose conducting region is deep bellow the surface [Sellier *et al.* 2011]. It works by placing a biased metallic tip close to the device surface, e.g. that of an Atomic force microscope (AFM). The electric field the tip generates penetrates the device heterostructure and reaches its conducting region, e.g. 2DEG in AlGaAs/GaAs heterostructure. As the amplitude of the potential generated by the tip increases, it depletes the charge beneath the tip. If the depletion disk underneath the tip is larger than the electron fermi wavelength, then backscattering of incoming electrons occur [1] [Fratus *et al.* 2019]. This significantly alters electron transport through the sample.

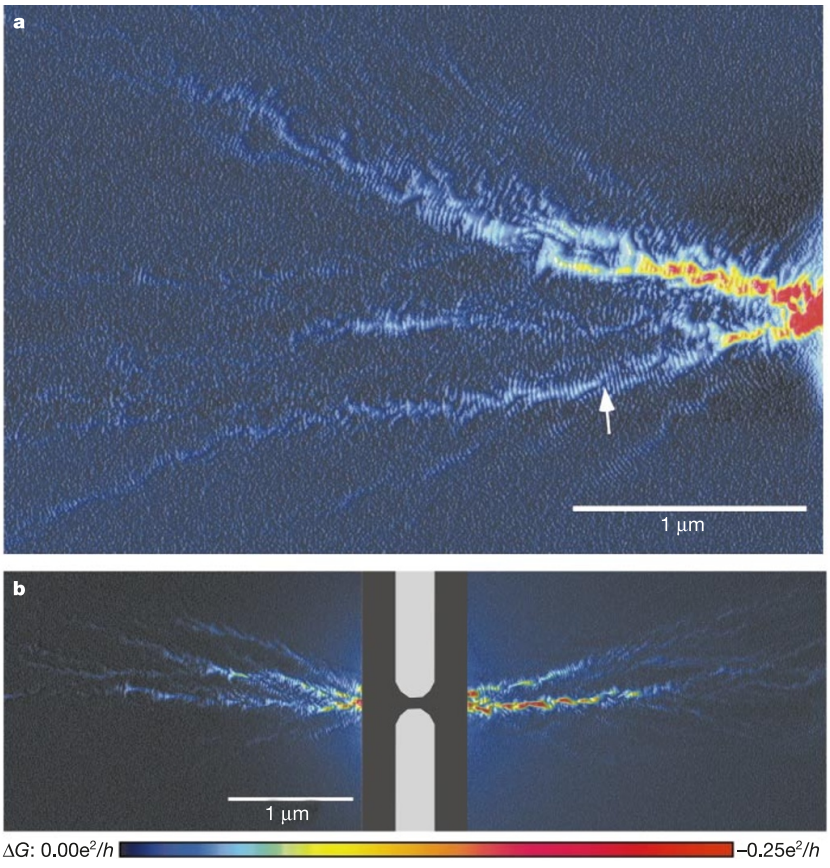
A typical transport quantity we study is the device conductance. Using the SGM setup we measure the impact of the metallic tip on the device conductance as a function of tip position and tip voltage. For instance, consider a quantum point contact fabricated in a AlGaAs/GaAs heterostructure. A typical SGM setup consists of hovering an AFM tip over the heterostructure and away from the QPC gates, see Figure 7.1(a). In a SGM measurement we first characterize the QPC conductance as a function of the QPC gate voltage for an unbiased AFM tip. Then we turn on the AFM tip voltage. For a given AFM tip voltage, AFM tip position and QPC gate voltage we measure the device conductance. We call ΔG the device conductance measured with a biased tip subtracted by that without the tip. By varying the tip position only, we obtain a 2D map of ΔG, hereafter referred to as a SGM (conductance) map. Figure 7.1(b) extracted from [Topinka *et al.* 2001] shows a 2D SGM map obtained by hovering the AFM tip away from the QPC gates. The higher the conductance variation due to the presence of the tip, higher is the electronic flow at that position in the device. Therefore, the dark regions in Figure 7.1(b) are areas of low electronic flow while the red regions are areas of high electronic flow. Hence the SGM map of 7.1(b) is a direct image of the electronic flow in a QPC.

One striking feature of Figure 7.1(b) is the branched pattern the electronic flow takes as it leaves the QPC. This is rather unintuitive, one would expect a fan like flow after the electrons go through the QPC constriction. Such branch-like structure has been shown to be due to disorder in the potential landscape where the electrons propagate [Topinka *et al.* 2001, Topinka *et al.* 2000]. For instance, [Jura *et al.* 2007] have shown a strong dependance of the branches on GaAs/AlGaAs 2DEG mobility. Figure 7.2, taken from [Jura *et al.* 2007], shows the SGM map for a dirty sample (left) and two high mobility samples (middle and right). In the dirty samples, the hard scattering centers, e.g. due to impurities, cause tall and sharp spikes in

[1] SGM experiments can be classified into a non-invasive and a invasive regime. In the first the tip voltage is weak, it only alters the local density of states and causes a small perturbation to electronic transport. In the invasive regime, the depletion disk formed underneath the tip is large enough to cause backscattering, thus strongly affecting transport. See [Steinacher *et al.* 2018, Fratus *et al.* 2019]



(a)



(b)

Figure 7.1: (a) Schematics of a SGM setup. In yellow a set of metallic gates creating a potential constriction in the middle of the sample conducting region (in green). In blue the atomic force microscope hovering over the sample surface. It scatters the electron waves as they leave the constriction. Taken from [Topinka *et al.* 2001] (b) SGM conduction map showing the electronic flow. Measured by [Topinka *et al.* 2001] for a QPC made on a AlGaAs/GaAs heterostructure. The QPC gate voltage is set such that the device conductance is $G = 2e^2/h$. The regions in red correspond to areas of high electronic flow and those in dark blue of low electronic flow.

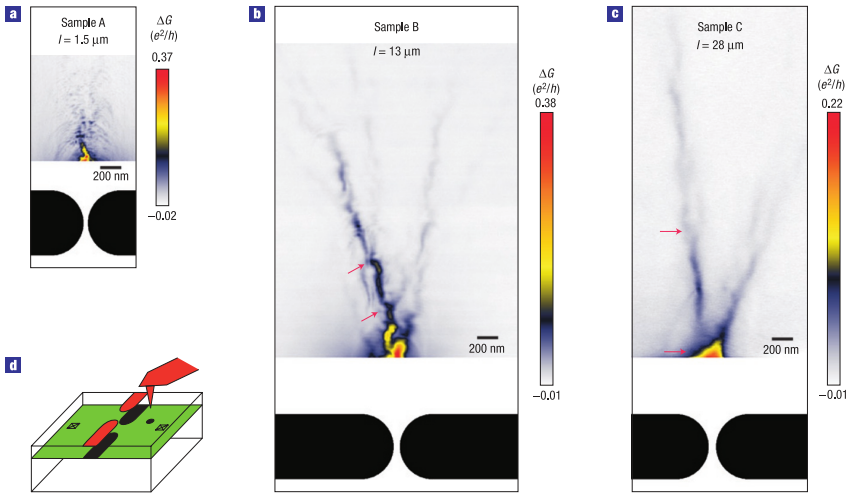


Figure 7.2: Taken from [Jura *et al.* 2007]. SGM maps for QPCs built on GaAs/AlGaAs 2DEG of different mobility. a) SGM map for a device with the 2DEG $57nm$ beneath the heterostructure surface and $22nm$ beneath the Si donors layer. It has a mobility $\mu = 0.14\ 10^6 cm^2 V^{-1} s^{-1}$ and mean free path $l = 1.5\mu m$. The electronic flow is twisted and diffusive. b) The 2DEG is placed $68nm$ beneath the heterostructure surface and $25nm$ beneath the Si donors layer. It has $\mu = 1.7\ 10^6 cm^2 V^{-1} s^{-1}$ and $l = 13\mu m$. The flow is much smoother and shows only few possible hard scattering sites, pointed by the arrows. c) The 2DEG is placed $100nm$ beneath the heterostructure surface and $68nm$ beneath the Si donors layer. It has $\mu = 4.4\ 10^6 cm^2 V^{-1} s^{-1}$ and $l = 28\mu m$. The flow is smooth and mostly straight. d) Schematics of the SGM setup.

the disordered potential. This in turn causes the diffusive and twisted flow of the dirty sample, see left Figure 7.2. For the clean samples, there are only small-angle scattering sites, e.g. uneven distribution of Si ionized donors, they cause smooth ondulations to the disordered potential. Such smooth ondulations cause the flow to be continuous and concentrated in straight and smooth branches, see middle and right Figure 7.2 [Jura *et al.* 2007]. Moreover, the branch structure has been shown stable under large changes of the 2DEG fermi energy [Fratus *et al.* 2019]. This indicates that the SGM response, in particular the form of the branches, mainly depends on the disorder in the potential seen by the propagating electrons.

This raises the question of wether we can use SGM maps to extract the disorder configuration of nanoelectronic devices. It is possible to calculate the SGM map for a given disorder configuration using numerical approaches, such as solving the SCQE problem discussed in the first part of this manuscript (left to right Figure 7.3). Solving the inverse problem, however, is a lot harder. One approach currently studied to extract the disorder potential from quantum transport measurements is to use Machine Learning techniques (right to left Figure 7.3).

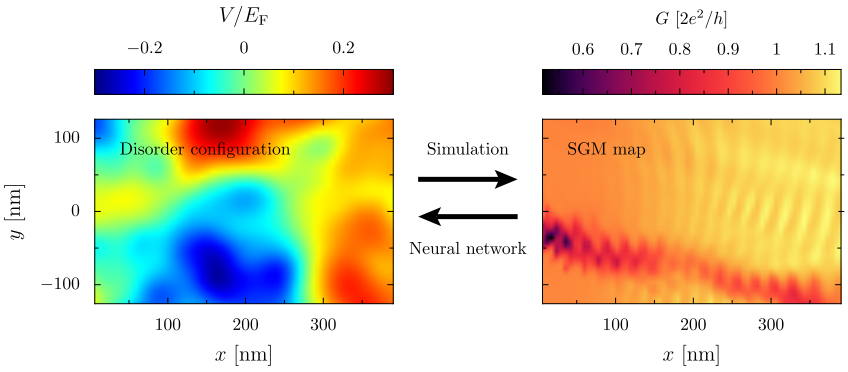


Figure 7.3: Example of the dataset we calculated to train the neural network. Left image shows a disorder potential configuration seen by the 2DEG. Right image shows the corresponding SGM map. For a given disorder configuration, kwant calculates the SGM map (left to right). For a given SGM map, a neural network trained to solve the inverse problem (right to left) predicts the associated disorder configuration. Simulations performed with *PESCADO* and Kwant for the GaAs/AlGaAs 2DEG QPC of Figure 7.6. Taken from [Percebois *et al.* 2023]

7.2 Neural Network approach for solving the inverse problem

Machine learning is a great tool to capture the behavior of a non-linear function $f(X)$ as long as the result $f(X) = A$ is known for a large number of X (thousands and tens of thousands). In a typical machine learning scheme, such as a neural network (NN), a large training data set X - where A is known, is used to train the NN into learning the behavior of the function $f(X)$. To calculate with which precision a NN has has learned $f(X)$, one typically uses a second (and typically smaller) set of X with known A as reference. Applied to our problem X is the SGM conductance map (right of Figure 7.3), A is the disorder potential (left of Figure 7.3) and $f(X)$ is a Neural Network mapping $X \to A$.

In order to train the Neural Network we need to generate thousands of SGM maps from known disorder configurations. However calculating one full SGM map means calculating the conductance through a QPC for hundreds of tip positions. Hence the computational cost of the machine learning based approaches is a considerable limitation. In a preliminary work by [Percebois & Weinmann 2021] they showed an artificial neural network is capable of extracting the disorder potential in QPCs from the samples partial local density of states (PLDOS), c.f. Eq.2 from [Percebois & Weinmann 2021]. The PLDOS is a fast to calculate quantity related to the disorder in the sample similarly to the SGM map. However the PLDOS can not be measured experimentally. After the proof-of-principle by [Percebois & Weinmann 2021], [Carlo R da Cunha & Lai 2022] and [da Cunha *et al.* 2023] trained an NN (with a different architecture compared to [Percebois & Weinmann 2021]) to extract the disorder potential from SGM maps.

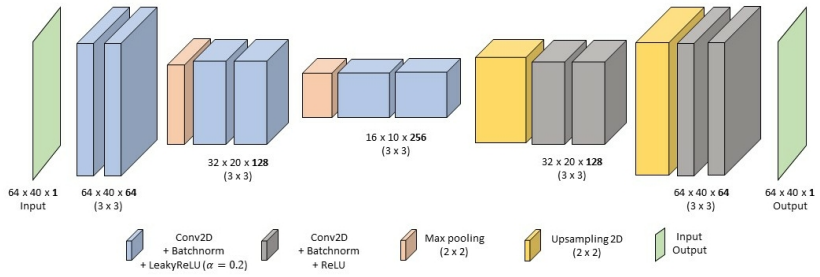


Figure 7.4: Convolutional encoder-decoder neural network used to solve the inverse problem, c.f. Figure 7.3. Taken from [Percebois *et al.* 2023].

Their work is a slight improvement on that by [Percebois & Weinmann 2021]. However when calculating the conductance of the QPC they used an effective potential to represent the effect of the gates and tip. A correct treatment of the device electrostatics requires a self-consistent treatment of the electronic screening. This is specially true underneath the tip, since depleting the charge changes considerably the electronic screening. The work we present in this Chapter builds up on the work by [Percebois & Weinmann 2021] and uses *PESCADO* to generate the SGM maps by solving the SCQE problem in the Thomas-Fermi approximation.

7.2.1 Neural Network architecture and training

The input and output of our neural network are both images (c.f. Figure 7.3), i.e. 2D matrices. Hence we used a NN of the convolutional encoder-decoder type, typically used for image analysis. A neural network can be divided into multiple layers, see Figure 7.4 for a schematics of the architecture we have used. In blue and grey are the convolutional type layers. Each convolutional layer can be thought of as applying a Kernel to its 2D input matrix. This allows the neural network to transform the input image data, e.g. to enhance a certain feature. In red are the encoder type layers, they reduce the number of pixels in the image, thus compressing the information. This forces the neural network to select the main features of the image. In yellow are the decoder type layers, they decompresses the image, forcing the neural network to extrapolate new image pixels. Therefore our NN first choses a feature of the SGM map to enhance, then compresses the image. This is the encoder part. Then the NN takes the compressed image, enhances a feature it deems pertinent, and then extrapolates from the image some new pixels. This is the decoder step. The specific features the NN enhances, removes after compression or extrapolate after decompression are optimized during training.

The training of our neural network happens in two steps. First it is pre-trained using the partial local density of states. We used 50000 PLDOS samples to train the neural network. Only after it is trained with the PLDOS, we train it using the SGM dataset. This considerably increases the accuracy of the neural network prediction. For more technical information regarding the neural network architecture, training

and statistical evaluation of the prediction quality, we refer to [Percebois *et al.* 2023, Percebois 2023].

Regarding the training dataset, for the PLDOS calculation we refer to [Percebois *et al.* 2023, Percebois & Weinmann 2021]. The SGM dataset is calculated from a numerical model reproducing the experiments of [Iordanescu *et al.* 2020]. In the next section we shall explain their experiments. Afterwards we shall describe the numerical model and the SGM calculations.

7.3 Experimental device and measurements

The QPCs of [Iordanescu *et al.* 2020] are fabricated over an $Al_{0.3}Ga_{0.7}As/GaAs$ heterostructure. Figure 7.5(a) shows a schematic of the experimental setup. In blue are the metallic gates defining the QPC, they are $300nm$ apart and located $10nm$ above the heterostructure surface. In grey is the metallic tip placed at a distance of $30nm$ from the heterostructure surface. In red is the 2DEG formed at the AlGaAs/GaAs interface located $57nm$ bellow the heterostructure surface and with an electron density of $n_s = 2.53.10^{15}m^2$. The electrons filling the 2DEG come from the Si doped AlGaAs $15nm$ thick layer placed $30nm$ above the 2DEG and $12nm$ bellow the heterostructure surface (not shown in the schematics). The conductance measurements are performed at a temperature of $100mK$. For further information regarding the device and technical details of the conductance measurements, we refer to [Iordanescu *et al.* 2020].

Figure 7.5(b) shows a conductance versus QPC gate voltage curve. The conductance plateaus are well defined at multiples of $G_0 = 2e^2/h$. For this QPC the tip starts to deplete the 2DEG beneath it for a $V_{tip} = -4.5V$. Figure 7.5(c) shows a SGM map for a QPC gate voltage of $V_g = -0.95V$ (black circle in Figure 7.5(b)) and tip voltage of $V_{tip} = -6V$. Figure 7.5(c) shows a single, well defined branch, as expected for a QPC in the first conductance plateau. For $V_{tip} = -6V$ the 2DEG is depleted up to an estimated radius of $60nm$ from the tip position. See [Iordanescu *et al.* 2020] for the method to extract the radius of depletion. Hence a electromagnetic cavity is formed between the QPC gates and the depleted 2DEG. For a device whose electronic coherence length is high enough [2] and a sufficiently high thermal length there are constructive / destructive interference between the backscattered electrons in the cavity [Jura *et al.* 2009]. Figure 7.5(d) shows the derivative of the data in Figure 7.5(c). It shows interference fringes characteristic of a Fabry-Pérot interferometer. Based on this observation we can extract the electron fermi wavelength from the period of the oscillations. For 7.5(d) one extracts $\lambda_F \approx 40nm$. This is similar to the theoretical value for $\lambda_F = \sqrt{2\pi/n_s} = 49.8nm$ calculated under the assumption of parabolic dispersion (valid considering we remain in the first conductance plateau).

[2] At least twice or four times the tip distance from the QPC depending on the interference mechanism [LeRoy *et al.* 2005]

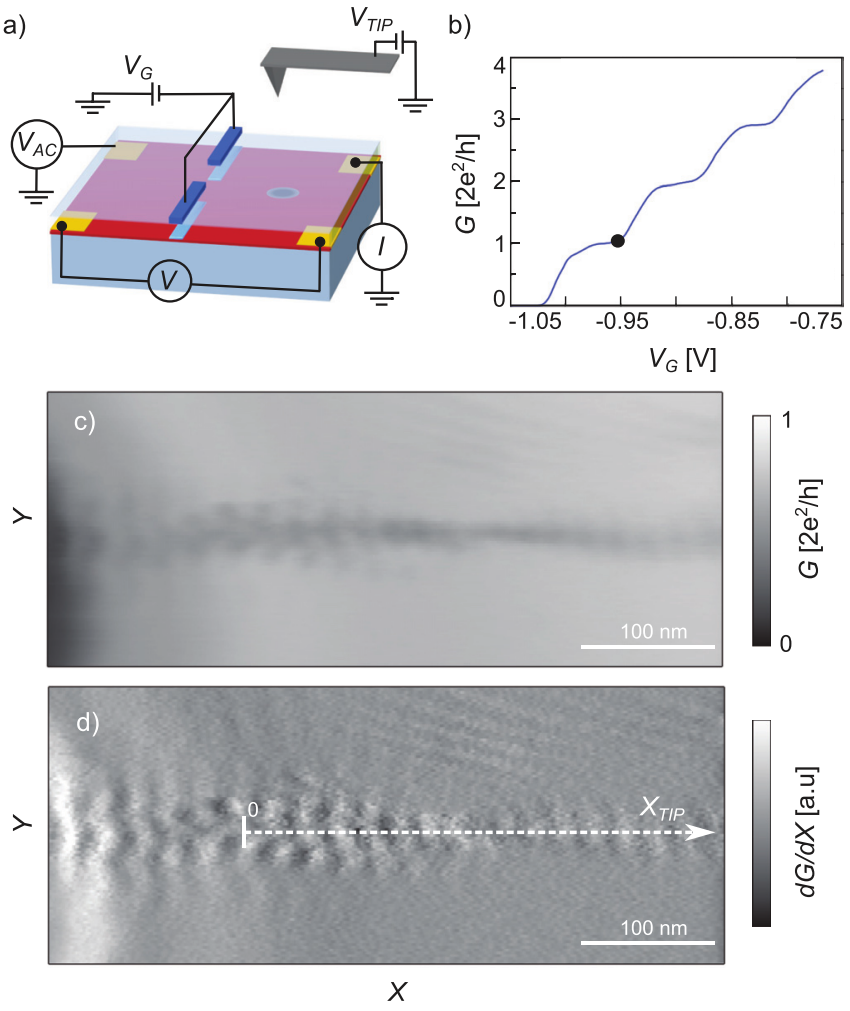


Figure 7.5: Taken from [Iordanescu *et al.* 2020]. (a) Schematics of a SGM setup of (b) Conductance versus QPC gate voltage V_g. (c) SGM conductance maps for $V_g = -0.95V$ and tip voltage $V_{tip} = -6V$. For a $V_g = -0.95V$ the QPC is placed in by the end of the first conductance plateau, see black circle in (c). (d) Numerical derivative of (c) with respect to X.

7.4 Generating the scanning gate microscopy training dataset

The transport data we have calculated is the SGM 2D conductance map shown in Figure 7.5(c). We have generated a large dataset of 2650 SGM conductance maps, each an image of 40×65 pixels, e.g. right panel of Figure 7.3. Each pixel correspond to calculating the device conductance for a given tip position. Hence to make one SGM map, we perform 2560 conductance calculations. Each conductance map is calculated for a unique randomly generated ionized dopant distribution, e.g. the disorder configuration on the left panel of Figure 7.3.

To generate such dataset we first calculate the electrostatics of the QPC device as a function of the tip position and dopant configuration, see third panel in Figure 7.7. To calculate the electrostatics we solve a 3D model inspired from the one developed in Chapter 6 using *PESCADO* under the Thomas Fermi approximation, see Section 7.4.1. Then we extract the electrostatic potential $U(r, r_{tip})$, c.f. Eq.(7.2), at the two-dimensional electron gas for a given dopant configuration and tip position r_{tip}, see Section 7.4.2. The third panel in Figure 7.7 shows an example of $U(r, r_{tip})$ extracted from the 3D *PESCADO* model. Finally we use a 2D tight binding non-interacting model built from the $U(r, r_{tip})$ (mean-field a.k.a thomas-fermi) potential to calculate the conductance (with Kwant), e.g. right panel of Figure 7.3. The conductance calculations were performed at zero temperature, i.e. for $E = E_F = 9.1meV$. To avoid backscattering from the leads in the tight-binding model, we have smoothly extended the potential extracted from *PESCADO* towards zero before sending it to Kwant. In the next two sections we shall explain respectively the 3D electrostatic model and how to extract from it the 2D electrostatic potential used for the tight-binding model.

7.4.1 Calculating the electrostatics of the quantum point contacts

We simulate the QPC of [Iordanescu *et al.* 2020] with the 3D model shown in Figure 7.6. The QPC metallic gates, in gold, have a width of 150nm, thickness of 40nm and are spaced from each other by 250nm. The dielectric stacking (blue, light blue and cyan) is, from top to bottom : 10nm Hafnium oxide, 5nm GaAs (vs 7nm experimentally), 5nm undoped AlGaAs, 15 doped AlGaAs, 30nm undoped AlGaAs and 80nm GaAs. The 2DEG is located between the AlGaAs/GaAs layer, as indicated in Figure 7.6. The dielectric permittivity is set to $\varepsilon_r = 12\varepsilon_0$ for the AlGaAs and GaAs and $\varepsilon_r = 20\varepsilon_0$ for the Hafnium oxide, with ε_0 the vacuum permittivity. The tip is modeled by a metallic half sphere with a radius of $50nm$ located $30nm$ from the top GaAs layer. The tip position on the (x, y) plane (referred to as r_{tip}) is not constant and can change from one simulation to another. The values r_{tip} can take are bounded by the grey region named "SGM zone" on the right side of Figure 7.6.

We solve the SCQE problem for this system under the Thomas Fermi approximation. At the 2DEG we fix the ILDOS to that of the bulk 2DEG, i.e. $N(\mu) = \rho\mu\theta(\mu)$,

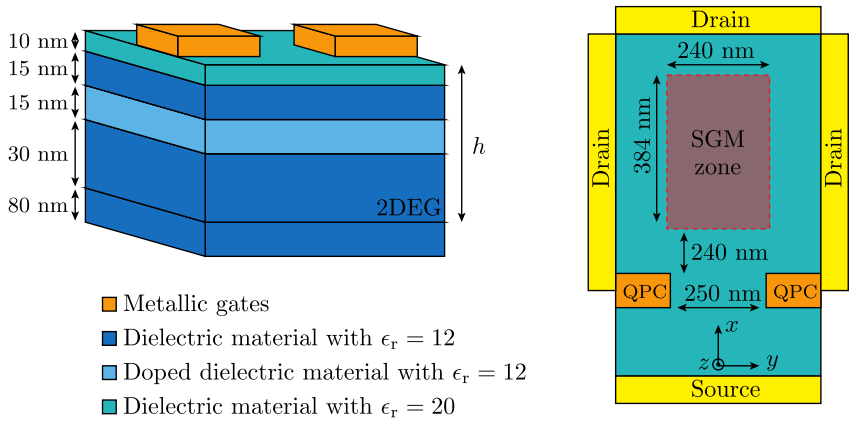


Figure 7.6: Sketch of the numerical model used to simulate the electrostatics of the QPC device measured by [Iordanescu *et al.* 2020]. Taken from [Percebois *et al.* 2023].

with μ the chemical potential, $\theta(\mu)$ the Heaviside step function and $\rho = m_e^*/(\pi\hbar^2)$. At the QPC gates we set Dirichlet b.c.s with $V_{QPC} = V_{gate} + V_{off}$, V_{gate} the experimental gate voltage and V_{off} an offset voltage to account for the materials workfunction and surface states (see Chapter 6). At the metallic tip we set Dirichlet b.c.s as well with V_{tip} the tip voltage. Everywhere else in the system we set Neumann b.c.s, with non-zero charge only at the dopants. We refer to Chapter 4 for tutorials on how to define this problem with *PESCADO* .

We set $V_{off} = 0.162V$ and fix $V_{gate} = -0.82V$, for which the QPC conductance is between the first and second plateaus depending on the ionized donor distribution. The V_{tip} is taken s.t. the depletion region it induces in the 2DEG layer has the same radius as the experimental depletion radius. To obtain a depletion radius of $60nm$ we set $V_{tip} = -5.8V$. For the dopant layer, we consider them not fully ionized, see the fermi level pinning argument of Chapter 6. We consider the position of the ionized donors to be random, following a Poissonian distribution s.t. the overall density of ionized donors is $n_{dop} = n_s/d_{dop} = 1.69.10^{23}m^{-3}$. The electrostatic model has two outputs: the potential $U(r, r_{tip})$ and charge density $n(r, r_{tip})$.

To discretized the 3D model on the left side of Figure 7.6 we use the finite volume mesher implemented in *PESCADO* . The electrostatic simulation box is of size $1068nm$ on $\vec{x}$, $564nm$ on $\vec{y}$ and $395nm$ on $\vec{z}$. At the boundaries of the simulation box the electric potential is considered constant, i.e. no flux leaves or enters the box. We only seek to reproduce the experimental result at the "SGM zone", making the simulation box size a good compromise between numerical precision and calculation time. We have used a non-uniform rectangular mesh. Along $\vec{x}$ and $\vec{y}$ the rectangle is $6nm$ long (in the entire system), this ensures the mesh is regular at the 2DEG region. Also, $6nm$ is much smaller than the electronic fermi wavelength at the 2DEG. Furthermore, the lattice spacing of the tight binding model Kwant solves is also $6nm$ along $\vec{x}$ and $\vec{y}$. This makes it trivial to use the electrostatic potential calculated

with *PESCADO* as input to the Kwant system. Regarding the discretization on $\vec{z}$, it follows the formula :

$$\begin{cases} 3z^2 + 2 & z < 0 \\ 5 & 0 < z < 115 \\ 20 & z > 115 \end{cases} \tag{7.1}$$

with $z = 115nm$ the vertical position delimiting the top of the QPC gates. For an example of the potential $U(r, r_{tip})$ calculated at the 2DEG, we refer to the two first panels (left to right) in Figure 7.7. The code can be made available upon request.

7.4.2 Approximation on the electrostatic potential

To generate a single SGM conductance map, we calculate the potential $U(r, r_{tip})$ for 2560 different tip positions. We need thousands of SGM conductance maps to train a neural network. Each SGM conductance map has a unique randomly calculated ionized dopant distribution. This puts the number of electrostatic calculations we would have to perform on the order of millions. This is unreasonable. There is however an approximation we can make that significantly reduces that number. We split the potential into three parts :

$$U(r, r_{tip}) \approx U(r) - U_{uni}(r) + U_{uni}(r, r_{tip}) \tag{7.2}$$

Where $U_{uni}(r)$ is the potential calculated for a system without the tip and with a uniform dopant distribution. Where $U(r)$ is the potential calculated for a system without the tip and with a disordered dopant distribution (see leftmost panel of Figure 7.7). Where $U_{uni}(r, r_{tip})$ is the potential calculated for a system with a tip placed at r_{tip} and with an uniform dopant distribution (see second, left to right, panel of Figure 7.7). This approximation implies we do not need to make a new electrostatic calculation for every tip position whenever we change the dopant disorder configuration. The third and fourth panels of Figure 7.7 compare respectively the approximated potential to the exact potential. It illustrates the case where the error induced by the approximation is the largest. As the tip is moved further away from the QPC gates, their cross talk is decreased and the approximation gains in accuracy.

7.5 Extracting the disordered potential from experimental data

With the SGM dataset of Section 7.4 we have trained the neural network of Section 7.2.1. In this section we shall first compare the disorder potential predicted from a numerical SGM map to the expected disorder potential. Then we shall extract the disorder potential from a experimental SGM map and evaluate the quality of the prediction.

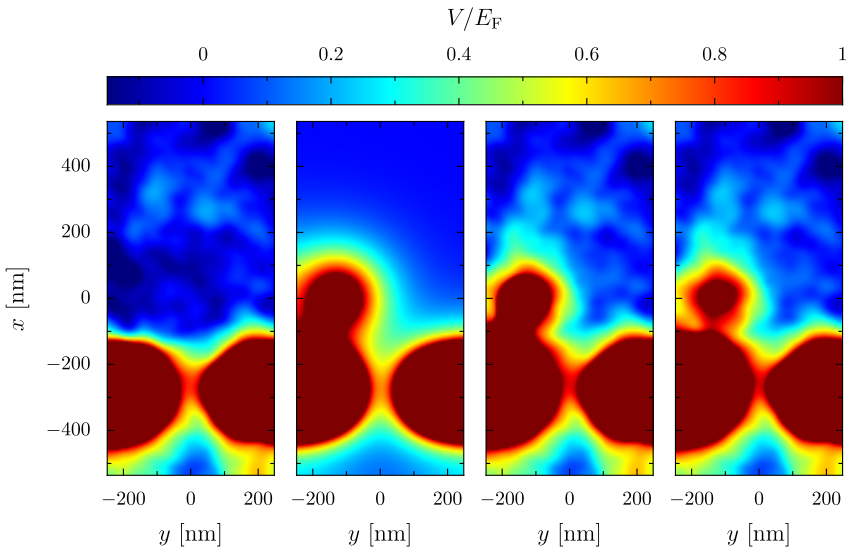


Figure 7.7: Electrostatic potential at the 2DEG calculated using the 3D model of Figure 7.6. First panel shows the disordered potential calculated for a system with zero tip voltage. The second panel shows the potential calculated for a system with non-zero tip voltage, but without disorder in the dopant layer. The third panel shows the approximated potential, c.f. Eq 7.2. The fourth panel shows the exact potential calculated for a system with disorder in the dopant layer and non-zero tip voltage. The error made by the approximated potential (third panel) is the largest when the tip is closest to the QPC gates. Taken from [Percebois *et al.* 2023]

During training the QPC gate voltage is fixed to $V_{gate} = -0.82V$. We change the gate voltage when testing the accuracy of the neural network to evaluate how it handles a slightly different QPC voltage profile. An unbiased neural network should predict the same disorder potential regardless of V_{gate}. Figure 7.8 shows the disorder potential predicted for a trained neural network as a function of the QPC gate voltage. The top panel is the calculated SGM map for one disorder potential (bottom panel) and four different QPC gate voltage values, corresponding to a QPC conductance of respectively $G = 1.0G_0$, $G = 1.3G_0$, $G = 1.7G_0$ and $G = 2.0G_0$. The middle panels show the disorder potential predicted by the neural network. It shows that the predicted disorder potential captures the main features of the expected disorder potential. Upon visual inspection it is possible to observe some discrepancies. However, it is difficult to evaluate from Figure 7.8 if the predicted potential contains all the meaningful information required to reproduce the initial SGM map. The SGM map does not depend only on local variations of the disorder potential. Furthermore, there is no reason to assume every pixel in the potential image has the same effect on the calculated SGM map.

To better evaluate the performance of the neural network, we decided to extract the disordered potential from an experimental SGM map (top panel of Figure 7.9) and then to use the extracted disordered potential to calculate (with Kwant) a simulated SGM map (four bottom lines in Figure 7.9). We compare the experimental SGM map with the calculated SGM map to assert the performance of the NN. Take the second line in in Figure 7.9. The disorder potential is the one predicted for the first experimental SGM map ($G = 1.0G_0$) on the top of Figure 7.9. The four SGM maps are calculated by using the predicted potential on the left and varying the QPC gate voltage s.t. the QPC conductance corresponds respectively to $G = 1.0G_0$, $G = 1.3G_0$, $G = 1.7G_0$ and $G = 2.0G_0$. For lines three, four and five we follow the exact same procedure, however using the disorder potential predicted from the experimental SGM map for respectively $G = 1.3G_0$, $G = 1.7G_0$, $G = 2.0G_0$. The main features of the branches are successfully captured. We do notice however some oscillations in the predicted SGM map compared to the experimental one. Considering the work shown here is the first of its kind, we consider this to be already of good resemblance. Of course, further investigation is required if this technique is to be used to predict the transport behavior of GaAs/AlGaAs 2DEG devices.

7.6 Conclusion

We have generated a large SGM dataset. Then we have used it to train a neural network into extracting the experimental disorder potential at the 2DEG layer of GaAs/AlGaAs QPCs. The work presented here is a good proof of principle. However for it to be used towards predictive modeling of nanoelectronic devices, we need to first improve the validation method of the neural network. The current method does not assess possible systematic bias in the neural network. To do so we would

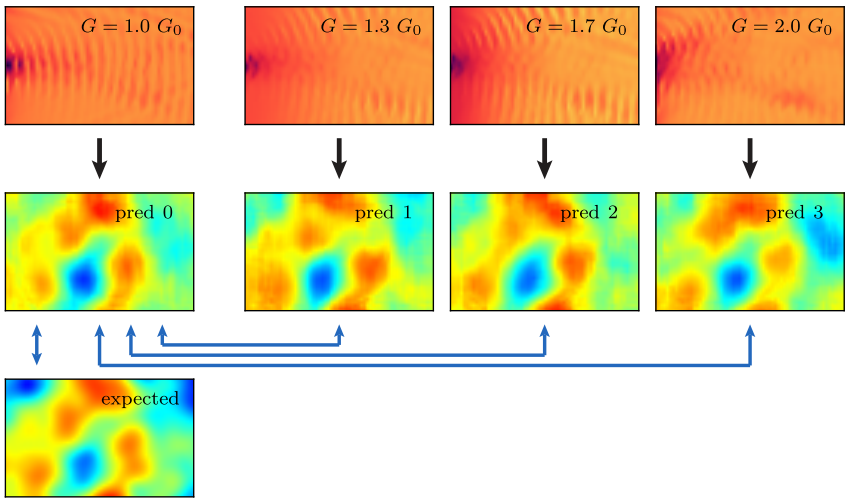


Figure 7.8: Top panels indicate the calculated SGM maps for a fixed disorder configuration (bottom panel) and four different QPC gate voltages. The middle panels contain the disorder potential predicted by the Neural network. Taken from [Percebois *et al.* 2023]

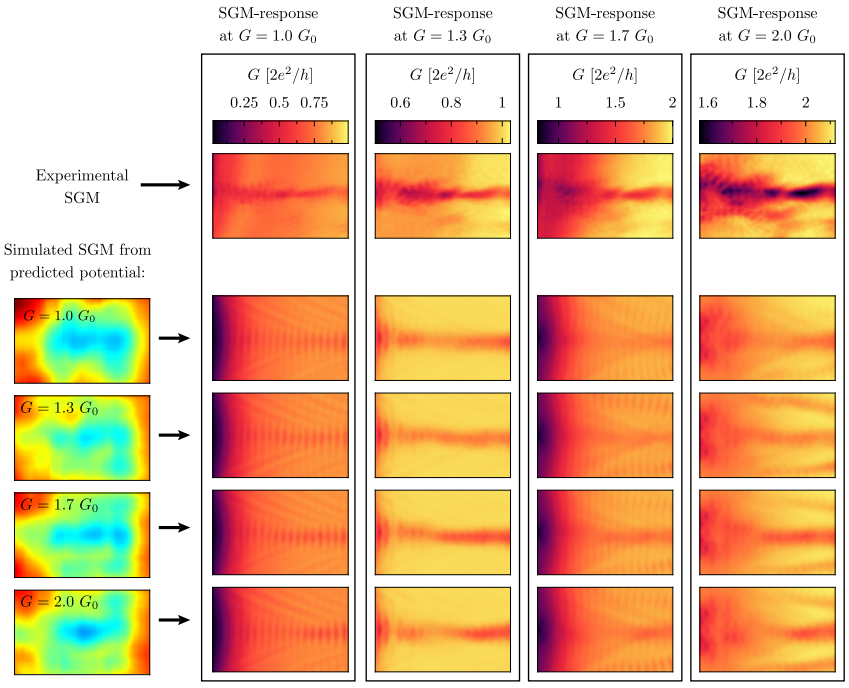


Figure 7.9: Top line four experimental SGM maps. The four bottom lines show the SGM maps calculated using the disorder potential extracted from the experimental SGM maps. Each one of the four bottom lines are calculated by keeping the disorder potential constant and changing the QPC gate voltage s.t. the conductance of the QPC matches the experimental value. Taken from [Percebois *et al.* 2023].

need a dedicated set of experiments. For instance, one where the SGM maps were measured under different tip voltages and magnetic fields. Once we have been able to assess the bias in the neural network, we can think about improving the quality of the training dataset or modify the NN architecture.

Regarding the numerical dataset, we could relax the approximations we made to the numerical model, such as the one to the disorder potential, c.f. Eq. 7.2. We could also change the voltage of the tip, in order to change the depletion radius of the 2DEG; increase the resolution of the SGM maps, i.e. number of pixels; or increase the size of the SGM scanning zone, c.f. grey rectangle of Figure 7.6.

Chapter 8

Conclusion

In this thesis we have worked on the predictive modeling of quantum transport in nanoelectronic devices. To this avail we have: i) developed an algorithm to solve the self-consistent quantum electrostatics problem in non-linear regimes (e.g. Quantum Hall Effect), ii) developed an open source python software *PESCADO* to model the electrostatics of quantum devices and iii) developed a minimalistic model with no fitting parameters to account for the effect of unexpected charges in experimental nanoelectronic devices.

The SCQE problem is known to be difficult to solve due to the non-linearities in the quantum part. The first main conclusion of this thesis is that to stabilize an algorithm attempting to solve the SCQE problem, one needs to use the information the local density of states (LDOS) has on the energy dependance of the quantum problem to detect where those non-linearities will appear, see Chapter 2 and 3. To be more precise, we introduce first the quantum adiabatic approximation (a generalization of Thomas Fermi), this simplifies the SCQE to a non-linear Helmholtz (NLH) equation. To solve the NLH equation then we use the LDOS to isolate the non-linearities. This is in contrast to most current approaches that rely only on the charge. We have implemented two algorithms to do so, the Piecewise Newton Raphson and Piecewise Linear. The first is very similar to the Newton-Raphson algorithm, is robust and converges quickly. The second is slower, but its convergence is guaranteed. Both methods rely on repeatedly solving the linear helmholtz (LH) equation. Hence, during my thesis we developed *PESCADO* , it allows one to solve the LH equation for a 1D, 2D or 3D model of nanoelectronic devices.

The *PESCADO* software, see Chapter 4, implements a versatile geometrical engine and a memory efficient finite volume mesher. It also implements the Piecewise Newton Raphson and Piecewise Linear algorithms. Hence *PESCADO* can solve the non-linear Helmholtz problem for nanoelectronic devices. We have extensively tested *PESCADO* during its development, either on the form of integrated testing (automatic tests for specific methods of the code) or through the collaboration with experimentalists and other theoreticians (comparison with experimental results). From the beginning of my thesis, where we worked on the graphene PN junctions (Chapter 5), to the end, the scanning gate microscopy application (Chapter 7), we have always searched to confront *PESCADO* 's calculations to experiments. This allowed us to not only validate our software, but also realize which features of *PESCADO* to develop in priority.

The difficulty in modeling quantum devices does not only comes from the non-linearity of the SCQE. It is also due to the non-local electrostatic interaction. In

one hand, this non-local aspect means we can use the gate field effect to change the behavior of conducting electrons. On the other hand, any other charge in the device will do so as well. During this thesis we have worked on figuring out what is the minimum level of modeling required to correctly simulate the rich electrostatic environment where electrons propagate. We have restricted our question to AlGaAs/GaAs heterostructures. We have shown that a minimalistic model should not seek to predict the charge density of the 2DEG, but rather use the experimental value to calibrate the theoretical model. More precisely, we have developed a model whose dopant charge density is calibrated from the experimental 2DEG charge density (Fermi level pinning) and used it to predict experimental pinch-off voltages with an accuracy on the order of $5-10\%$, see Chapter 6. With the same modeling technique, we have used *PESCADO* to generate a numerical data-set to train a Neural Network into reconstructing the disorder potential (due to dopant disorder) from scanning gate microscopy experiments, see Chapter 7.

Continuing the work of this Thesis can be done in three paths, applications, software and formalism. Regarding the first path it is about extending on the exploratory work done in this thesis regarding the predictive modeling of experimental devices and the effect of disorder. We have ongoing collaborations with the Lateqs (experimental nanoelectronics) group within Pheliqs (specially Boris Brun) as well with Christopher Bäuerle at Institut Néel. For the group at Lateqs the work is on its early stages, we have only started to work on the quantum dots they fabricate on SiGe/Ge/SiGe heterostructures [1] and we are still developing the appropriate model to capture the charges in the device (similar to the work in Chapter 6). The collaboration with C.Bäuerle in contrast is much more developed (Xavier's work with him pre-dates me by quite some years), the nearest next step is to predict the actual shape of the conductance plateaus of their devices. The longer term goal is to actually use *PESCADO* to guide the design of future flying qubit devices. The work done on the scanning gate microscopy data is quite exploratory and to continue it would be important to develop a collaboration with an experimental group so that we can design some experiments to better asses the fidelity of the neural network and the *PESCADO* simulations. Last on the applications path, one interesting long term goal is to predict the 0.7 conductance anomaly [Thomas *et al.* 1996].

Regarding the software path, it is focused on expanding *PESCADO* capabilities as a software. On the immediate term it is its full release as an open-source software, there is still mainly some documentation work to be finished before end of summer. Then, for a future version of *PESCADO* we need to revisit the implementation of the Piecewise Linear algorithm (it is too slow) and the linear helmholtz solver (we can probably speed it up). We also need to improve the visualization and ILDOS integration tools as well as the compatibility with Kwant (it works, but there is some work to do).

Regarding the formalism path, we have solved only the Hartree term of the Hartree-Fock (HF) equation. That is, we do not include the effect of the exchange

[1] For a relevant review on the type of devices they work with see [Scappucci *et al.* 2021]

interaction [2]. The most immediate solution is to include the Fock term, however the long range of the exchange interaction makes this approach computationally expensive. A lower cost approach, and that might suffice, is to use density functional theory (DFT). Through the local density approximation we can write a potential (function of the charge density) that accounts for the exchange interaction. This potential is inserted in the Schrödinger equation of the SCQE problem and hence is quite straightforward to implement.

Lastly, to go beyond DFT or HF, we could use the mean field (or even DFT or HF) potentials calculated with *PESCADO* to obtain the lowest energy propagating modes, in e.g. a QPC device. With this information we can map the 3D model into a effective 1D system, to which we can add correlation. More precisely, current is carried only through the lowest modes, hence the contribution of the rest can be neglected. We can then use the algorithm proposed by [Darancet *et al.* 2010] to derive a basis set for the restricted Hilbert space spanned by the contributing conduction modes where the system Hamiltonian is tridiagonal, with only 1D first neighbors hopping. To this 1D Hamiltonian we can add correlation and then solve the many-body problem using state of the art techniques such as tensor train decomposition being currently developed by Xavier's Waintal group [Núñez Fernández *et al.* 2022, Ritter *et al.* 2024]. This is an interesting approach to use what we have learned in this thesis to bridge the gap between (semi) classical models and full quantum models.

[2] In Chapter 5 we have accounted for the effect of the exchange interaction on the QH graphene ILDOS phenomenologically.

Appendices

Appendix A

Quantum Point Contacts ETC ///

In this appendix, we collect the values of the different pinch-off voltages V_1, V_2 and V_3 that have been extracted from the experimental $I(V_g)$ curves. The raw experimental data can be found in [dat 2022].

QPC	W(nm)	L(nm)	V3a	V2a	V1a	V3b	V2b	V1b	V3c	V2c	V1c
A1	250	50	-2.10	-0.86	-0.44	-1.95	-0.86	-0.45	-2.20	-0.89	-0.46
A2	300	100	-2.09	-0.88	-0.44	-1.87	-0.88	-0.45	-2.02	-0.90	-0.46
A3	300	250	-1.41	-0.88	-0.45	-1.29	-0.87	-0.45	-1.52	-0.89	-0.46
A4	300	500	-1.22	-0.89	-0.45	-1.17	-0.87	-0.45	-1.22	-0.89	-0.46
A5	500	1000	-1.96	-0.88	-0.45	-1.84	-0.88	-0.45	-2.05	-0.90	-0.46
A6	500	2500	-1.46	-0.86	-0.45	-1.82	-0.86	-0.45	-1.97	-0.91	-0.46
A7	500	5000	-1.83	-0.88	-0.44	-	-	-	-	-	-
A8	500	1e4	-1.79	-0.88	-0.45	-	-	-	-	-	-

QPC	W(nm)	L(nm)	V3d	V2d	V1d
A1	250	50	-2.00	-0.93	-0.47
A2	300	100	-2.13	-0.95	-0.47
A3	300	250	-1.47	-0.94	-0.47
A4	300	500	-1.24	-0.93	-0.46
A5	500	1000	-1.97	-0.94	-0.46
A6	500	2500	-1.90	-0.90	-0.46

QPC	W(nm)	R(nm)	V3a	V2a	V1a	V3b	V2b	V1b	V3c	V2c	V1c
B1	250	25	-1.94	-0.88	-0.43	-2.43	-0.95	-0.48	-	-	-
B2	300	50	-2.35	-0.88	-0.44	-2.42	-0.95	-0.48	-	-	-
B3	300	125	-1.71	-0.88	-0.44	-1.77	-0.96	-0.48	-1.59	-0.94	-0.47
B4	300	250	-1.51	-0.86	-0.44	-1.57	-0.96	-0.49	-	-	-
B5	500	500	-2.31	-0.88	-0.44	-2.49	-0.98	-0.50	-	-	-
B6	500	1250	-1.98	-0.87	-0.44	-2.20	-0.97	-0.49	-	-	-
B7	500	2500	-1.97	-0.86	-0.45	-2.08	-0.97	-0.50	-2.00	-0.91	-0.47
B8	500	5000	-1.93	-0.87	-0.44	-2.00	-0.98	-0.50	-	-	-

QPC	W(nm)	R(nm)	L(nm)	V3a	V2a	V1a	V3b	V2b	V1b	V3c	V2c
C1	250	1000	50	-1.0	-0.98	-0.49	-0.96	-0.87	-0.44	-1.05	-0.92
C2	300	1000	100	-1.64	-0.90	-0.48	-1.09	-0.87	-0.45	-	-
C3	300	1000	250	-1.21	-0.98	-0.49	-1.09	-0.88	-0.45	-1.21	-0.94
C4	300	1000	500	-1.18	-0.97	-0.48	-1.08	-0.86	-0.45	-1.15	-0.92
C5	500	1000	1000	-1.98	-0.93	-0.48				-2.02	-0.89
C6	500	1000	2500	-1.94	-0.95	-0.48	-1.79	-0.89	-0.45	-1.93	-0.92
C7	500	1000	5000	-1.91	-0.95	-0.48	-1.68	-0.88	-0.45	-1.92	-0.91
C8	500	1000	10000	-1.85	-0.94	-0.48	-1.70	-0.88	-0.45	-1.87	-0.92

Table A.1: Experimental pinch-off voltages for the short designs.

QPC	W(nm)	R(nm)	L(nm)	V3a	V2a	V1a	V3b	V2b	V1b
A9	750	0	1000	-3.37	-0.90	-0.45	-3.34	-0.87	-0.44
A10	750	0	2500	-3.17	-0.89	-0.45	-3.10	-0.88	-0.44
A11	750	0	5000	-3.03	-0.90	-0.45	-3.00	-0.88	-0.44
A12	750	0	10000	-2.99	-0.78	-0.44	-2.92	-0.78	-0.43
A13	750	0	25000	-2.89	-0.78	-0.44	-2.85	-0.77	-0.43
A14	750	0	50000	-1.96	-0.77	-0.44	-2.72	-0.78	-0.43
A15	1000	0	10000	-4.1	-0.90	-0.44	-4.18	-0.91	-0.44
A16	1000	0	50000	-3.92	-0.78	-0.42	-	-	-
B9	750	500	0	-3.81	-0.86	-0.44	-4.00	-0.90	-0.47
B10	750	1250	0	-3.50	-0.89	-0.44	-3.40	-0.86	-0.46
B11	750	2500	0	-3.25	-0.89	-0.44	-3.18	-0.88	-0.45
B12	750	5000	0	-3.21	-0.77	-0.43	-3.10	-0.75	-0.45
B13	750	12500	0	-3.11	-0.78	-0.44	-3.02	-0.77	-0.44
B14	750	25000	0	-3.08	-0.78	-0.44	-3.04	-0.78	-0.44
B15	1000	5000	0	-4.00	-0.90	-0.45	-7.0	-0.90	-0.46
B16	1000	25000	0	-3.47	-0.80	-0.44	-	-	-
C9	750	1000	1000	-3.07	-0.87	-0.45	-3.20	-0.89	-0.45
C10	750	1000	2500	-2.96	-0.90	-0.45	-3.02	-0.89	-0.45
C11	750	1000	5000	-2.92	-0.89	-0.45	-1.10	-0.89	-0.45
C12	750	1000	10000	-1.73	-0.77	-0.44	-2.93	-0.77	-0.45
C13	750	1000	25000	-0.78	-0.76	-0.44	-2.82	-0.77	-0.45
C14	750	1000	50000	-1.91	-0.76	-0.43	-	-	-
C15	1000	1000	10000	-4.13	-0.91	-0.46	-4.17	-0.93	-0.47
C16	1000	1000	50000	-3.88	-0.79	-0.45	-3.55	-0.78	-0.44

Table A.2: Experimental pinch-off voltages for the long designs. Designs that have $W_g = 80$ nm in the narrow gate region are highlighted in blue. The rest have $W_g = 50$ nm like the short designs.

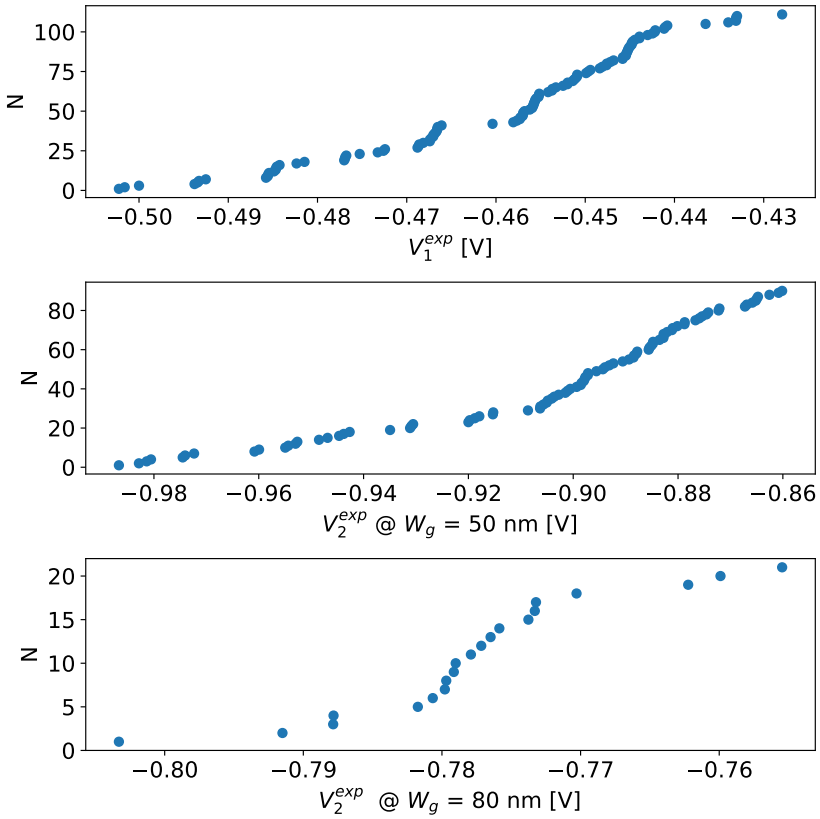


Figure A.1: Cumulative distribution of V_1 (top), V_2 with $W_g = 50$ nm (middle) and V_2 with $W_g = 80$ nm (bottom) for the entire set of measured samples. The N versus x plots show the number of samples N(x) whose pinch-off voltage V_1/V_2 is smaller than x.

Appendix B

Integrated of the local density of states

In this Appendix we derive an analytical expression for the integrated local density of states based on the Kernel Polynomial method (KPM) [Weiße *et al.* 2006]. The KPM consists of writing the LDOS as a finite series of Chebyshev polynomials whose expansion coefficients (called moments) are modified by damping kernels to improve convergence.

Let $\rho_i(E)$ be the LDOS at the site i of a tight-binding system we can solve with e.g. Kwant [Groth *et al.* 2014]. First we write the LDOS as a infinite series of Chebyshev polynomials that we truncate at M:

$$\rho_i(\tilde{E}) \approx \frac{1}{\pi\sqrt{1-\tilde{E}^2}}\left[\mu_{0,i} + 2\sum_{m=1}^{M-1}\mu_{m,i}T_m(\tilde{E})\right] \tag{B.1}$$

with $\mu_{m,i}$ the moments and $T_m(\tilde{E})$ the Chebyshev polynomials of first kind :

$$T_m(\tilde{E}) = cos(m\arccos(\tilde{E})) \tag{B.2}$$

Since $T_m(\tilde{E})$ are only defined on the real interval $[-1,1]$, $\tilde{E}$ are rescaled energies:

$$\begin{aligned}\tilde{E} &= (E-\beta)/\alpha \\ \alpha &= (E_{max}-E_{min})/(2-\varepsilon) \\ \beta &= (E_{max}+E_{min})/2\end{aligned} \tag{B.3}$$

with ε a small cutoff added to avoid instabilities near the borders of the interval $[-1,1]$. The moments $\mu_{m,i}$ we calculate with Kwant [Groth *et al.* 2014], otherwise see [Weiße *et al.* 2006] for an analytical expression.

Using Eq.(B.1) leads to fluctuations when $\rho_i(\tilde{E})$ is not continuously differentiable (Gibbs oscillations). To damp these oscillations we modify the moments $\mu_m \to g_m\mu_m$, with g_m depending on the order of the approximation M. The truncation of the series up to M and the modification of the moments is equivalent to the convolution of $\rho_i(\tilde{E})$ with a Kernel (see equations 48 and 49 of [Weiße *et al.* 2006]). For the integration of the LDOS in question, we best use the Jackson-Kernel [1]. In the Jackson-Kernel we write :

[1] We refer to Figure 1 of [Weiße *et al.* 2006] for a good comparison of different kernels effect on damping the oscillations that arise when Chebyshev expanding the delta function.

$$g_m = \frac{1}{M+1}\left((M-m+1)\cos\left(\frac{\pi m}{M+1}\right) + \sin\left(\frac{\pi m}{M+1}\right)\cot\left(\frac{\pi}{M+1}\right)\right) \quad \text{(B.4)}$$

It is possible to integrate the KPM LDOS with Kwant, it uses the Chebyshev-Gauss quadrature rule. Although it works for any distribution function, e.g. Fermi distribution, it has the drawback of defining the ILDOS only at the energy values of the nodes of the Chebyshev polynomial. It makes the ILDOS look like a staircase. Fortunately at $T = 0$ it is rather straightforward to derive an analytical expression to the ILDOS, we do so bellow.

First we write the primitive function of $\rho_i(\tilde{E})$:

$$P_i(\tilde{E}) = \int \rho_i(\tilde{E})d\tilde{E} = \int \frac{1}{\pi\sqrt{1-\tilde{E}^2}}\left[\mu_{0,i} + 2\sum_{m=1}^{M-1}\mu_{m,i}T_m(\tilde{E})\right]d\tilde{E} \quad \text{(B.5)}$$

where we introduce the change of variables $\tilde{E} = \cos(\theta)$ s.t.

$$P_i(\tilde{E}) = \frac{-1}{\pi}\left[\mu_{0,i}\theta + 2\sum_{m=1}^{M-1}\mu_{m,i}\int\cos(m\theta)d\theta\right] \quad \text{(B.6)}$$

hence the primitive reads :

$$P_i(\tilde{E}) = \frac{-1}{\pi}\left[\mu_{0,i}\arccos(\tilde{E}) + 2\sum_{m=1}^{M-1}\frac{\mu_{m,i}}{m}\sin(m\arccos(\tilde{E})) + \mathrm{C}\right] \quad \text{(B.7)}$$

with C a constant.

At $T = 0$ the fermi distribution is a step function, hence the ILDOS reads :

$$n_i(\tilde{E}) = \int_{-1}^{\tilde{E}_F}\rho_i(\tilde{E})d\tilde{E} = \frac{-1}{\pi}\left[\mu_{0,i}\left[\arccos(\tilde{E}_F) - \pi\right] + 2\sum_{m=1}^{M-1}\frac{\mu_{m,i}}{m}\sin(m\arccos(\tilde{E}_F))\right] \quad \text{(B.8)}$$

with $\tilde{E}_F = (E_F - \beta)/\alpha$ and E_F the fermi energy.

Bibliography

[Ahn *et al.* 2021] Seongjin Ahn, Haining Pan, Benjamin Woods, Tudor D. Stanescu and Sankar Das Sarma. *Estimating disorder and its adverse effects in semiconductor Majorana nanowires.* Physical Review Materials, vol. 5, no. 12, page 124602, 2021. (Cited on page 147.)

[Al'tshuler & Spivak 1985] B. L. Al'tshuler and B. Z. Spivak. *Variation of the random potential and the conductivity of samples of small dimensions.* JETP Lett, vol. 42, no. 9, pages 447–450, November 1985. (Cited on page 19.)

[Amestoy *et al.* 2001] P. R. Amestoy, I. S. Duff, J. Koster and J.-Y. L'Excellent. *A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling.* SIAM Journal on Matrix Analysis and Applications, vol. 23, no. 1, pages 15–41, 2001. (Cited on pages 58 and 102.)

[Amestoy *et al.* 2006] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent and S. Pralet. *Hybrid scheduling for the parallel solution of linear systems.* Parallel Computing, vol. 32, no. 2, pages 136–156, 2006. (Cited on pages 58 and 102.)

[Anderson 1965] Donald G. Anderson. *Iterative Procedures for Nonlinear Integral Equations.* J. ACM, vol. 12, no. 4, page 547–560, oct 1965. (Cited on page 12.)

[Andrei & Mayergoyz 2004] Petru Andrei and Isaak Mayergoyz. *Analysis of fluctuations in semiconductor devices through self-consistent Poisson-Schrödinger computations.* Journal of Applied Physics, vol. 96, no. 4, pages 2071–2079, 08 2004. (Cited on page 12.)

[Antipov *et al.* 2018] Andrey E. Antipov, Arno Bargerbos, Georg W. Winkler, Bela Bauer, Enrico Rossi and Roman M. Lutchyn. *Effects of Gate-Induced Electric Fields on Semiconductor Majorana Nanowires.* Phys. Rev. X, vol. 8, page 031041, Aug 2018. (Cited on page 12.)

[Armagnat & Waintal 2020] Pacome Armagnat and Xavier Waintal. *Reconciling edge states with compressible stripes in a ballistic mesoscopic conductor.* Journal of Physics: Materials, vol. 3, no. 2, page 02LT01, mar 2020. (Cited on pages 5, 6, 7, 9, 45, 69, 130, 140 and 141.)

[Armagnat *et al.* 2019] Pacome Armagnat, A. Lacerda-Santos, Benoit Rossignol, Christoph Groth and Xavier Waintal. *The self-consistent quantum-electrostatic problem in strongly non-linear regime.* SciPost Phys., vol. 7, page 031, 2019. (Cited on pages 8, 13, 14, 18, 20, 22, 27, 33, 45, 48, 49, 54, 69, 140, 141 and 160.)

[Armagnat 2019] Pacôme Armagnat. *Self-consistent quantum-electrostatics*. Theses, Université Grenoble Alpes, June 2019. (Cited on pages 13 and 20.)

[Atalla *et al.* 1959] M. M. Atalla, E. Tannenbaum and E. J. Scheibner. *Stabilization of silicon surfaces by thermally grown oxides*. The Bell System Technical Journal, vol. 38, no. 3, pages 749–783, 1959. (Cited on pages 2 and 17.)

[Barber *et al.* 1996] C Bradford Barber, David P Dobkin and Hannu Huhdanpaa. *The quickhull algorithm for convex hulls*. ACM Transactions on Mathematical Software (TOMS), vol. 22, no. 4, pages 469–483, 1996. (Cited on pages 88 and 95.)

[Bardeen & Brattain 1948] J. Bardeen and W. H. Brattain. *The Transistor, A Semi-Conductor Triode*. Phys. Rev., vol. 74, pages 230–231, Jul 1948. (Cited on page 1.)

[Bauer *et al.* 2013] Florian Bauer, Jan Heyder, Enrico Schubert, David Borowsky, Daniela Taubert, Benedikt Bruognolo, Dieter Schuh, Werner Wegscheider, Jan von Delft and Stefan Ludwig. *Microscopic origin of the '0.7-anomaly' in quantum point contacts*. Nature, vol. 501, no. 7465, pages 73–78, Sep 2013. (Cited on page 2.)

[Bäuerle *et al.* 2018] Christopher Bäuerle, D Christian Glattli, Tristan Meunier, Fabien Portier, Patrice Roche, Preden Roulleau, Shintaro Takada and Xavier Waintal. *Coherent control of single electrons: a review of current progress*. Reports on Progress in Physics, vol. 81, no. 5, page 056503, May 2018. (Cited on pages 2, 28, 69 and 129.)

[Bautze *et al.* 2014] Tobias Bautze, Christoph Süssmeier, Shintaro Takada, Christoph Groth, Tristan Meunier, Michihisa Yamamoto, Seigo Tarucha, Xavier Waintal and Christopher Bäuerle. *Theoretical, numerical, and experimental study of a flying qubit electronic interferometer*. Phys. Rev. B, vol. 89, page 125432, Mar 2014. (Cited on pages 2, 3, 28, 31 and 69.)

[Bhattacharyya *et al.* 2019] Rajarshi Bhattacharyya, Mitali Banerjee, Moty Heiblum, Diana Mahalu and Vladimir Umansky. *Melting of Interference in the Fractional Quantum Hall Effect: Appearance of Neutral Modes*. Physical Review Letters, vol. 122, no. 24, page 246801, June 2019. (Cited on page 3.)

[Birner *et al.* 2007] Stefan Birner, Tobias Zibold, Till Andlauer, Tillmann Kubis, Matthias Sabathil, Alex Trellakis and Peter Vogl. *nextnano: General Purpose 3-D Simulations*. IEEE Transactions on Electron Devices, vol. 54, no. 9, pages 2137–2142, 2007. (Cited on pages 13 and 157.)

[Bolotin *et al.* 2009] Kirill I. Bolotin, Fereshte Ghahari, Michael D. Shulman, Horst L. Stormer and Philip Kim. *Observation of the fractional quantum*

Hall effect in graphene. Nature, vol. 462, no. 7270, pages 196–199, November 2009. (Cited on page 3.)

[Broyden 1965] C. G. Broyden. *A class of methods for solving nonlinear simultaneous equations.* Math. Comp., vol. 19, pages 577–593, 1965. (Cited on page 12.)

[Brun & Fouad Kalo 2023] Boris Brun and Ahmad Fouad Kalo. Project meeting. 2023. (Cited on page 15.)

[Buks *et al.* 1994a] E Buks, M Heiblum, Y Levinson and H Shtrikman. *Scattering of a two-dimensional electron gas by a correlated system of ionized donors.* Semiconductor Science and Technology, vol. 9, no. 11, page 2031, nov 1994. (Cited on pages 16, 17, 167, 169, 170 and 171.)

[Buks *et al.* 1994b] E. Buks, M. Heiblum and Hadas Shtrikman. *Correlated charged donors and strong mobility enhancement in a two-dimensional electron gas.* Phys. Rev. B, vol. 49, pages 14790–14793, May 1994. (Cited on pages 16, 160, 169 and 171.)

[Büttiker 1986] M. Büttiker. *Four-Terminal Phase-Coherent Conductance.* Phys. Rev. Lett., vol. 57, pages 1761–1764, Oct 1986. (Cited on page 7.)

[Büttiker 1988] M. Büttiker. *Absence of backscattering in the quantum Hall effect in multiprobe conductors.* Phys. Rev. B, vol. 38, pages 9375–9389, Nov 1988. (Cited on pages 5 and 7.)

[Carlo R da Cunha & Lai 2022] David K Ferry Carlo R da Cunha Nobuyuki Aoki and Ying-Cheng Lai. *A method for finding the background potential of quantum devices from scanning gate microscopy data using machine learning.* Machine Learning: Science and Technology, vol. 3, no. 025013, 2022. (Cited on page 181.)

[Carrega *et al.* 2021] Matteo Carrega, Luca Chirolli, Stefan Heun and Lucia Sorba. *Anyons in Quantum Hall Interferometry.* Nature Reviews Physics, vol. 3, no. 10, pages 698–711, October 2021. arXiv: 2109.13427. (Cited on page 3.)

[Castro Neto *et al.* 2009] A. H. Castro Neto, F. Guinea, N. M. R. Peres, K. S. Novoselov and A. K. Geim. *The electronic properties of graphene.* Reviews of Modern Physics, vol. 81, no. 1, pages 109–162, January 2009. (Cited on page 141.)

[Chadi & Chang 1989] D. J. Chadi and K. J. Chang. *Energetics of DX-center formation in GaAs and* $Al_xGa_{1-x}As$ *alloys.* Phys. Rev. B, vol. 39, pages 10063–10074, May 1989. (Cited on page 17.)

[Chand *et al.* 1984] Naresh Chand, Tim Henderson, John Klem, W. Ted Masselink, Russ Fischer, Yia-Chung Chang and Hadis Morkoĉ. *Comprehensive analysis*

of Si-doped $Al_xGa_{1-x}As$ *(*$x = 0to1$*): Theory and experiments.* Phys. Rev. B, vol. 30, pages 4481–4492, Oct 1984. (Cited on page 16.)

[Chang 1990] A. M. Chang. *A unified transport theory for the integral and fractional quantum hall effects: Phase boundaries, edge currents, and transmission/reflection probabilities.* Solid State Communications, vol. 74, no. 9, pages 871–876, June 1990. (Cited on page 3.)

[Chatzikyriakou *et al.* 2022] Eleni Chatzikyriakou, Junliang Wang, Lucas Mazzella, Antonio Lacerda-Santos, Maria Cecilia da Silva Figueira, Alex Trellakis, Stefan Birner, Thomas Grange, Christopher Bäuerle and Xavier Waintal. *Unveiling the charge distribution of a GaAs-based nanoelectronic device: A large experimental dataset approach.* Phys. Rev. Res., vol. 4, page 043163, Dec 2022. (Cited on pages 21, 23, 31, 69 and 147.)

[Chklovskii *et al.* 1992a] D. B. Chklovskii, B. I. Shklovskii and L. I. Glazman. *Electrostatics of edge channels.* Phys. Rev. B, vol. 46, pages 4026–4034, Aug 1992. (Cited on pages 3, 5, 7, 44, 45, 130, 139 and 141.)

[Chklovskii *et al.* 1992b] D. B. Chklovskii, B. I. Shklovskii and L. I. Glazman. *Erratum: Electrostatics of edge channels.* Phys. Rev. B, vol. 46, pages 15606–15606, Dec 1992. (Cited on pages 3 and 7.)

[Chklovskii *et al.* 1993] D. B. Chklovskii, K. A. Matveev and B. I. Shklovskii. *Ballistic conductance of interacting electrons in the quantum Hall regime.* Phys. Rev. B, vol. 47, pages 12605–12617, May 1993. (Cited on pages 3, 7, 44, 45 and 141.)

[Chung *et al.* 2017] Yoon Jang Chung, K. W. Baldwin, K. W. West, D. Kamburov, M. Shayegan and L. N. Pfeiffer. *Design rules for modulation-doped AlAs quantum wells.* Phys. Rev. Mater., vol. 1, page 021002, Jul 2017. (Cited on page 167.)

[Chung *et al.* 2019] Yoon Jang Chung, Kirk W. Baldwin, Kenneth W. West, Nicholas Haug, Johannes van de Wetering, Mansour Shayegan and Loren N. Pfeiffer. *Spatial Mapping of Local Density Variations in Two-dimensional Electron Systems Using Scanning Photoluminescence.* Nano Letters, vol. 19, no. 3, pages 1908–1913, 2019. PMID: 30785759. (Cited on pages 19, 166, 173 and 175.)

[Curatola & Iannaccone 2003] G. Curatola and G. Iannaccone. *NANOTCAD2D: Two-dimensional code for the simulation of nanoelectronic devices and structures.* Computational Materials Science, vol. 28, no. 2, pages 342–352, 2003. Proceedings of the Symposium on Software Development for Process and Materials Design. (Cited on page 12.)

[da Cunha *et al.* 2023] C.R. da Cunha, N. Aoki, D.K. Ferry, A. Velasquez and Y. Zhang. *An investigation of the background potential in quantum constrictions using scanning gate microscopy and a swarming algorithm.* Physica A: Statistical Mechanics and its Applications, vol. 614, page 128550, 2023. (Cited on page 181.)

[Darancet *et al.* 2010] Pierre Darancet, Valerio Olevano and Didier Mayou. *Quantum transport through resistive nanocontacts: Effective one-dimensional theory and conductance formulas for nonballistic leads.* Phys. Rev. B, vol. 81, page 155422, Apr 2010. (Cited on page 195.)

[Das Sarma & Hwang 2014] S. Das Sarma and E. H. Hwang. *Mobility versus quality in two-dimensional semiconductor structures.* Phys. Rev. B, vol. 90, page 035425, Jul 2014. (Cited on page 14.)

[Das Sarma & Stern 1985] S. Das Sarma and Frank Stern. *Single-particle relaxation time versus scattering time in an impure electron gas.* Phys. Rev. B, vol. 32, pages 8442–8444, Dec 1985. (Cited on page 14.)

[dat 2022] *Experimental dataset, doi: 10.5281/zenodo.6498343.* Apr 2022. (Cited on pages 147, 150, 157 and 199.)

[Davies 1997] John H. Davies. The Physics of Low-dimensional Semiconductors, volume 34. Cambridge University Press, Dec 1997. (Cited on page 167.)

[de C. Chamon *et al.* 1997] C. de C. Chamon, D. E. Freed, S. A. Kivelson, S. L. Sondhi and X. G. Wen. *Two point-contact interferometer for quantum Hall systems.* Phys. Rev. B, vol. 55, pages 2331–2343, Jan 1997. (Cited on page 129.)

[Dean *et al.* 2010] C. Dean, A. Young, I. Meric and al. *Boron nitride substrates for high-quality graphene electronics.* Nature Nanotech, vol. 5, page 722–726, 2010. (Cited on page 130.)

[Deviatov & Lorke 2008] E. V. Deviatov and A. Lorke. *Experimental realization of a Fabry-Perot-type interferometer by copropagating edge states in the quantum Hall regime.* Phys. Rev. B, vol. 77, page 161302, Apr 2008. (Cited on page 129.)

[Duprez *et al.* 2019] H. Duprez, E. Sivre, A. Anthore, A. Aassime, A. Cavanna, A. Ouerghi, U. Gennser and F. Pierre. *Macroscopic Electron Quantum Coherence in a Solid-State Circuit.* Phys. Rev. X, vol. 9, page 021030, May 2019. (Cited on page 2.)

[Edlbauer *et al.* 2022] Hermann Edlbauer, Junliang Wang, Thierry Crozes, Pierre Perrier, Seddik Ouacel, Clément Geffroy, Giorgos Georgiou, Eleni Chatzikyriakou, Antonio Lacerda-Santos, Xavier Waintal, D. Christian Glattli, Preden Roulleau, Jayshankar Nath, Masaya Kataoka, Janine Splettstoesser,

Matteo Acciai, Maria Cecilia da Silva Figueira, Kemal Öztas, Alex Trellakis, Thomas Grange, Oleg M. Yevtushenko, Stefan Birner and Christopher Bäuerle. *Semiconductor-based electron flying qubits: review on recent progress accelerated by numerical modelling.* EPJ Quantum Technology, vol. 9, no. 1, page 21, dec 2022. (Cited on page 2.)

[Escribano *et al.* 2019] Samuel D. Escribano, Alfredo Levy Yeyati, Yuval Oreg and Elsa Prada. *Effects of the electrostatic environment on superlattice Majorana nanowires.* Phys. Rev. B, vol. 100, page 045301, Jul 2019. (Cited on pages 12 and 13.)

[Eyert 1996] V. Eyert. *A Comparative Study on Methods for Convergence Acceleration of Iterative Vector Sequences.* Journal of Computational Physics, vol. 124, no. 2, pages 271–285, 1996. (Cited on page 12.)

[Flór *et al.* 2022] I. M. Flór, A. Lacerda-Santos, G. Fleury, P. Roulleau and X. Waintal. *Positioning of edge states in a quantum Hall graphene pn junction.* Phys. Rev. B, vol. 105, page L241409, Jun 2022. (Cited on pages 3, 13, 23, 60 and 69.)

[Fouad Kalo 2023] Ahmad Fouad Kalo. First year master thesis. Master's thesis, 2023. (Cited on page 126.)

[Fratus *et al.* 2019] Keith R. Fratus, Rodolfo A. Jalabert and Dietmar Weinmann. *Energy stability of branching in the scanning gate response of two-dimensional electron gases with smooth disorder.* Phys. Rev. B, vol. 100, page 155435, Oct 2019. (Cited on pages 19, 177, 178 and 180.)

[Groth *et al.* 2014] Christoph W Groth, Michael Wimmer, Anton R Akhmerov and Xavier Waintal. *Kwant: a software package for quantum transport.* New Journal of Physics, vol. 16, no. 6, page 063065, jun 2014. (Cited on pages 11, 23, 54, 68, 76 and 203.)

[Gudmundsson 1990] Vidar Gudmundsson. *Oscillating impurity spectra caused by non-linear screening in the quantum hall regime.* Solid State Communications, vol. 74, no. 2, pages 63–67, 1990. (Cited on page 11.)

[Gummel 1964] H.K. Gummel. *A self-consistent iterative scheme for one-dimensional steady state transistor calculations.* IEEE Transactions on Electron Devices, vol. 11, no. 10, pages 455–465, 1964. (Cited on page 11.)

[Halperin 1982] B. I. Halperin. *Quantized Hall conductance, current-carrying edge states, and the existence of extended states in a two-dimensional disordered potential.* Phys. Rev. B, vol. 25, pages 2185–2190, Feb 1982. (Cited on page 7.)

[Ionicioiu *et al.* 2001] R Ionicioiu, G Amaratunga and F Udrea. *Computation with ballistic electrons.* Int. J. Mod. Phys. B, vol. 15, pages 125–133, 2001. (Cited on page 143.)

[Iordanescu *et al.* 2020] A. Iordanescu, S. Toussaint, G. Bachelier, S. Fallahi, C. G. Gardner, M. J. Manfra, B. Hackens and B. Brun. *Accurate characterization of tip-induced potential using electron interferometry.* Applied Physics Letters, vol. 117, no. 19, page 193101, 11 2020. (Cited on pages 23, 177, 183, 184, 185 and 186.)

[Ji *et al.* 2000] Yang Ji, M. Heiblum, D. Sprinzak, D. Mahalu and Hadas Shtrikman. *Phase Evolution in a Kondo-Correlated System.* Science, vol. 290, no. 5492, pages 779–783, 2000. (Cited on page 129.)

[Ji *et al.* 2003] Yang Ji, Yunchul Chung, D. Sprinzak, M. Heiblum, D. Mahalu and Hadas Shtrikman. *An electronic Mach–Zehnder interferometer.* Nature, vol. 422, no. 6930, pages 415–418, March 2003. (Cited on pages 2, 129, 131 and 132.)

[Jo *et al.* 2021] M. Jo, P. Brasseur, A. Assouline, G. Fleury, H.-S. Sim, K. Watanabe, T. Taniguchi, W. Dumnernpanich, P. Roche, D. C. Glattli, N. Kumada, F. D. Parmentier and P. Roulleau. *Quantum Hall Valley Splitters and a Tunable Mach-Zehnder Interferometer in Graphene.* Phys. Rev. Lett., vol. 126, page 146803, Apr 2021. (Cited on pages 2, 22, 130, 132, 133, 134, 136, 137, 138, 141 and 142.)

[Jura *et al.* 2007] M. P. Jura, M. A. Topinka, L. Urban, A. Yazdani, H. Shtrikman, L. N. Pfeiffer, K. W. West and D. Goldhaber-Gordon. *Unexpected features of branched flow through high-mobility two-dimensional electron gases.* Nature Phys, vol. 3, 2007. (Cited on pages 177, 178 and 180.)

[Jura *et al.* 2009] M. P. Jura, M. A. Topinka, M. Grobis, L. N. Pfeiffer, K. W. West and D. Goldhaber-Gordon. *Electron interferometer formed with a scanning probe tip and quantum point contact.* Phys. Rev. B, vol. 80, page 041303, Jul 2009. (Cited on pages 20 and 183.)

[Knoll & Keyes 2004] D.A. Knoll and D.E. Keyes. *Jacobian-free Newton–Krylov methods: a survey of approaches and applications.* Journal of Computational Physics, vol. 193, no. 2, pages 357–397, 2004. (Cited on page 12.)

[Koch *et al.* 2007] Jens Koch, Terri M. Yu, Jay Gambetta, A. A. Houck, D. I. Schuster, J. Majer, Alexandre Blais, M. H. Devoret, S. M. Girvin and R. J. Schoelkopf. *Charge-insensitive qubit design derived from the Cooper pair box.* Phys. Rev. A, vol. 76, page 042319, Oct 2007. (Cited on page 2.)

[Kumar *et al.* 1990] Arvind Kumar, Steven E. Laux and Frank Stern. *Electron states in a GaAs quantum dot in a magnetic field.* Phys. Rev. B, vol. 42, pages 5166–5175, Sep 1990. (Cited on page 12.)

[Lake *et al.* 1997] R. Lake, G. Klimeck, R. C. Bowen, D. Jovanovic, D. Blanks and M. Swaminathan. *Quantum Transport with Band-Structure and Schottky Contacts.* physica status solidi (b), vol. 204, no. 1, pages 354–357, 1997. (Cited on page 12.)

[Landau & Lifshitz 1981] L D Landau and L M Lifshitz. Quantum mechanics non-relativistic theory, volume 3. Pergamon, 3 édition, 1981. (Cited on page 5.)

[Lawaetz 1971] P. Lawaetz. *Valence-Band Parameters in Cubic Semiconductors.* Phys. Rev. B, vol. 4, pages 3460–3467, Nov 1971. (Cited on page 4.)

[Lee & Stone 1985] P. A. Lee and A. Douglas Stone. *Universal Conductance Fluctuations in Metals.* Phys. Rev. Lett., vol. 55, pages 1622–1625, Oct 1985. (Cited on page 19.)

[LeRoy *et al.* 2005] B. J. LeRoy, A. C. Bleszynski, K. E. Aidala, R. M. Westervelt, A. Kalben, E. J. Heller, S. E. J. Shaw, K. D. Maranowski and A. C. Gossard. *Imaging Electron Interferometer.* Phys. Rev. Lett., vol. 94, page 126801, Apr 2005. (Cited on page 183.)

[MacLeod *et al.* 2009] S. J. MacLeod, K. Chan, T. P. Martin, A. R. Hamilton, A. See, A. P. Micolich, M. Aagesen and P. E. Lindelof. *Role of background impurities in the single-particle relaxation lifetime of a two-dimensional electron gas.* Phys. Rev. B, vol. 80, page 035310, Jul 2009. (Cited on pages 14 and 17.)

[Martinez & Niquet 2022] Biel Martinez and Yann-Michel Niquet. *Variability of Electron and Hole Spin Qubits Due to Interface Roughness and Charge Traps.* Phys. Rev. Appl., vol. 17, page 024022, Feb 2022. (Cited on pages 3 and 17.)

[Mikkelsen *et al.* 2018] August E. G. Mikkelsen, Panagiotis Kotetes, Peter Krogstrup and Karsten Flensberg. *Hybridization at Superconductor-Semiconductor Interfaces.* Phys. Rev. X, vol. 8, page 031040, Aug 2018. (Cited on pages 12 and 13.)

[Nakamura *et al.* 2020] J. Nakamura, S. Liang, G. C. Gardner and M. J. Manfra. *Direct observation of anyonic braiding statistics.* Nature Physics, vol. 16, no. 9, pages 931–936, September 2020. (Cited on pages 3 and 129.)

[Niquet *et al.* 2014] Yann-Michel Niquet, Viet-Hung Nguyen, François Triozon, Ivan Duchemin, Olivier Nier and Denis Rideau. *Quantum calculations of the carrier mobility: Methodology, Matthiessen's rule, and comparison with semi-classical approaches.* Journal of Applied Physics, vol. 115, no. 5, page 054512, 02 2014. (Cited on page 12.)

[Núñez Fernández *et al.* 2022] Yuriel Núñez Fernández, Matthieu Jeannin, Philipp T. Dumitrescu, Thomas Kloss, Jason Kaye, Olivier Parcollet and

Xavier Waintal. *Learning Feynman Diagrams with Tensor Trains.* Phys. Rev. X, vol. 12, page 041018, Nov 2022. (Cited on page 195.)

[Pacelli 1997] A. Pacelli. *Self-consistent solution of the Schrodinger equation in semiconductor devices by implicit iteration.* IEEE Transactions on Electron Devices, vol. 44, no. 7, pages 1169–1171, 1997. (Cited on page 12.)

[Percebois & Weinmann 2021] Gaëtan J. Percebois and Dietmar Weinmann. *Deep neural networks for inverse problems in mesoscopic physics: Characterization of the disorder configuration from quantum transport properties.* Phys. Rev. B, vol. 104, page 075422, Aug 2021. (Cited on pages 3, 20, 176, 177, 181, 182 and 183.)

[Percebois *et al.* 2023] Gaëtan J. Percebois, Antonio Lacerda-Santos, Boris Brun, Benoit Hackens, Xavier Waintal and Dietmar Weinmann. *Reconstructing the potential configuration in a high-mobility semiconductor heterostructure with scanning gate microscopy.* SciPost Phys., vol. 15, page 242, 2023. (Cited on pages 23, 60, 69, 176, 181, 182, 183, 186, 188 and 190.)

[Percebois 2023] Gaëtan Percebois. *Quantum tranTransport quantique dans les gaz bi-dimensionnels d'électrons : une approche par l'intelligence artificielle pour la caractérisation des échantillons.* PhD thesis, 2023. Thèse de doctorat dirigée par Weinmann, Dietmar Physique Strasbourg 2023. (Cited on page 183.)

[Pfeiffer *et al.* 1989] Loren Pfeiffer, K. W. West, H. L. Stormer and K. W. Baldwin. *Electron mobilities exceeding 107 cm2/Vs in modulation-doped GaAs.* Applied Physics Letters, vol. 55, no. 18, pages 1888–1890, 10 1989. (Cited on pages 2 and 17.)

[Pierre *et al.* 2022] Fréderic Pierre, Christopher Bäuerle and Patrice Roche. *Multiple Discussions.* 2022. (Cited on pages 38 and 170.)

[Pulay 1980] Péter Pulay. *Convergence acceleration of iterative sequences. the case of scf iteration.* Chemical Physics Letters, vol. 73, no. 2, pages 393–398, 1980. (Cited on page 12.)

[Qian *et al.* 2017] Q. Qian, J. Nakamura, S. Fallahi, G. C. Gardner, J. D. Watson, S. Lüscher, J. A. Folk, G. A. Csáthy and M. J. Manfra. *Quantum lifetime in ultrahigh quality GaAs quantum wells: Relationship to $\Delta_{5/2}$ and impact of density fluctuations.* Phys. Rev. B, vol. 96, page 035309, Jul 2017. (Cited on pages 19, 166, 173 and 175.)

[Ralls *et al.* 1984] K. S. Ralls, W. J. Skocpol, L. D. Jackel, R. E. Howard, L. A. Fetter, R. W. Epworth and D. M. Tennant. *Discrete Resistance Switching in Submicrometer Silicon Inversion Layers: Individual Interface Traps and Low-Frequency ($\frac{1}{f}$?) Noise.* Phys. Rev. Lett., vol. 52, pages 228–231, Jan 1984. (Cited on page 19.)

[Rana *et al.* 1996] Farhan Rana, Sandip Tiwari and D. A. Buchanan. *Self-consistent modeling of accumulation layers and tunneling currents through very thin oxides.* Applied Physics Letters, vol. 69, no. 8, pages 1104–1106, 08 1996. (Cited on page 12.)

[Rassekh *et al.* 2019] Amin Rassekh, Farzan Jazaeri, Morteza Fathipour and Jean-Michel Sallese. *Modeling Interface Charge Traps in Junctionless FETs, Including Temperature Effects.* IEEE Transactions on Electron Devices, vol. 66, no. 11, pages 4653–4659, 2019. (Cited on page 17.)

[Ren *et al.* 2003] Zhibin Ren, R. Venugopal, S. Goasguen, S. Datta and M.S. Lundstrom. *nanoMOS 2.5: A two-dimensional simulator for quantum transport in double-gate MOSFETs.* IEEE Transactions on Electron Devices, vol. 50, no. 9, pages 1914–1925, 2003. (Cited on page 12.)

[Ritter *et al.* 2024] Marc K. Ritter, Yuriel Núñez Fernández, Markus Wallerberger, Jan von Delft, Hiroshi Shinaoka and Xavier Waintal. *Quantics Tensor Cross Interpolation for High-Resolution Parsimonious Representations of Multivariate Functions.* Phys. Rev. Lett., vol. 132, page 056501, Jan 2024. (Cited on page 195.)

[Roussely *et al.* 2018] Gregoire Roussely, Everton Arrighi, Giorgos Georgiou, Shintaro Takada, Martin Schalk, Matias Urdampilleta, Arne Ludwig, Andreas D. Wieck, Pacome Armagnat, Thomas Kloss, Xavier Waintal, Tristan Meunier and Christopher Bäuerle. *Unveiling the bosonic nature of an ultrashort few-electron pulse.* Nature Communications, vol. 9, no. 1, jul 2018. (Cited on page 28.)

[Saminadayar *et al.* 1997] L. Saminadayar, D. C. Glattli, Y. Jin and B. Etienne. *Observation of the $e/3$ Fractionally Charged Laughlin Quasiparticle.* Phys. Rev. Lett., vol. 79, pages 2526–2529, Sep 1997. (Cited on page 3.)

[Scappucci *et al.* 2021] Giordano Scappucci, Christoph Kloeffel, Floris A. Zwanenburg, Daniel Loss, Maksym Myronov, Jian-Jun Zhang, Silvano De Franceschi, Georgios Katsaros and Menno Veldhorst. *The germanium quantum information route.* Nature Reviews Materials, vol. 6, no. 10, pages 926–943, Oct 2021. (Cited on page 194.)

[Schuster *et al.* 1997] R. Schuster, E. Buks, M. Heiblum, D. Mahalu, V. Umansky and Hadas Shtrikman. *Phase measurement in a quantum dot via a double-slit interference experiment.* Nature, vol. 385, no. 6615, pages 417–420, Jan 1997. (Cited on page 129.)

[Sellier *et al.* 2011] H Sellier, B Hackens, M G Pala, F Martins, S Baltazar, X Wallart, L Desplanque, V Bayot and S Huant. *On the imaging of electron transport in semiconductor quantum structures by scanning-gate microscopy: successes and limitations.* Semiconductor Science and Technology, vol. 26, no. 6, page 064008, apr 2011. (Cited on page 178.)

[Shayegan *et al.* 1988a] M. Shayegan, V. J. Goldman, C. Jiang, T. Sajoto and M. Santos. *Growth of low density two dimensional electron system with very high mobility by molecular beam epitaxy.* Applied Physics Letters, vol. 52, no. 13, pages 1086–1088, 03 1988. (Cited on pages 2 and 17.)

[Shayegan *et al.* 1988b] M. Shayegan, V. J. Goldman, M. Santos, T. Sajoto, L. Engel and D. C. Tsui. *Two-dimensional electron system with extremely low disorder.* Applied Physics Letters, vol. 53, no. 21, pages 2080–2082, 11 1988. (Cited on page 3.)

[Steinacher *et al.* 2018] R. Steinacher, C. Pöltl, T. Krähenmann, A. Hofmann, C. Reichl, W. Zwerger, W. Wegscheider, R. A. Jalabert, K. Ensslin, D. Weinmann and T. Ihn. *Scanning gate experiments: From strongly to weakly invasive probes.* Phys. Rev. B, vol. 98, page 075426, Aug 2018. (Cited on page 178.)

[Stern 1970] Frank Stern. *Iteration methods for calculating self-consistent fields in semiconductor inversion layers.* Journal of Computational Physics, vol. 6, no. 1, pages 56–67, 1970. (Cited on page 12.)

[Stone 1985] A. Douglas Stone. *Magnetoresistance Fluctuations in Mesoscopic Wires and Rings.* Phys. Rev. Lett., vol. 54, pages 2692–2695, Jun 1985. (Cited on page 19.)

[Tan *et al.* 1990] I-H. Tan, G. L. Snider, L. D. Chang and E. L. Hu. *A self-consistent solution of Schrödinger–Poisson equations using a nonuniform mesh.* Journal of Applied Physics, vol. 68, no. 8, pages 4071–4076, 10 1990. (Cited on page 11.)

[Thomas *et al.* 1996] K. J. Thomas, J. T. Nicholls, M. Y. Simmons, M. Pepper, D. R. Mace and D. A. Ritchie. *Possible Spin Polarization in a One-Dimensional Electron Gas.* Phys. Rev. Lett., vol. 77, pages 135–138, Jul 1996. (Cited on pages 148 and 194.)

[Topinka *et al.* 2000] M. A. Topinka, B. J. LeRoy, S. E. J. Shaw, E. J. Heller, R. M. Westervelt, K. D. Maranowski and A. C. Gossard. *Imaging Coherent Electron Flow from a Quantum Point Contact.* Science, vol. 289, no. 5488, pages 2323–2326, 2000. (Cited on pages 2 and 178.)

[Topinka *et al.* 2001] M. A. Topinka, B. J. LeRoy, R. M. Westervelt, S. E. J. Shaw, R. Fleischmann, E. J. Heller, K. D. Maranowski and A. C. Gossard. *Coherent branched flow in a two-dimensional electron gas.* Nature, vol. 410, 2001. (Cited on pages 19, 176, 177, 178 and 179.)

[Trellakis & Ravaioli 1999] A. Trellakis and U. Ravaioli. *Lateral scalability limits of silicon conduction channels.* Journal of Applied Physics, vol. 86, no. 7, pages 3911–3916, 10 1999. (Cited on page 12.)

[Trellakis *et al.* 1997] A. Trellakis, A. T. Galick, A. Pacelli and U. Ravaioli. *Iteration scheme for the solution of the two-dimensional Schrödinger-Poisson equations in quantum structures.* Journal of Applied Physics, vol. 81, no. 12, pages 7880–7884, 06 1997. (Cited on pages 12 and 13.)

[Trellakis *et al.* 2007] Alex Trellakis, Tobias Zibold, Till Andlauer, Stefan Birner, R. Kent Smith, Richard Morschl and Peter Vogl. *The 3D nanometer device project nextnano: Concepts, methods, results.* Journal of Computational Electronics, vol. 5, no. 4, pages 285–289, May 2007. (Cited on page 157.)

[Trifunovic & Brouwer 2019] Luka Trifunovic and Piet W. Brouwer. *Valley isospin of interface states in a graphene p n junction in the quantum Hall regime.* Physical Review B, vol. 99, no. 20, page 205431, May 2019. (Cited on page 135.)

[Tworzydło *et al.* 2007] J. Tworzydło, I. Snyman, A. R. Akhmerov and C. W. J. Beenakker. *Valley-isospin dependence of the quantum Hall effect in a graphene p-n junction.* Physical Review B, vol. 76, no. 3, page 035411, July 2007. (Cited on page 135.)

[van Houten & Beenakker 1996] Henk van Houten and Carlo Beenakker. *Quantum Point Contacts.* Physics Today, vol. 49, no. 7, pages 22–27, 07 1996. (Cited on page 2.)

[van Wees *et al.* 1988] B. J. van Wees, H. van Houten, C. W. J. Beenakker, J. G. Williamson, L. P. Kouwenhoven, D. van der Marel and C. T. Foxon. *Quantized conductance of point contacts in a two-dimensional electron gas.* Phys. Rev. Lett., vol. 60, pages 848–850, Feb 1988. (Cited on pages 2 and 148.)

[Vuik *et al.* 2016] A Vuik, D Eeltink, A R Akhmerov and M Wimmer. *Effects of the electrostatic environment on the Majorana nanowire devices.* New Journal of Physics, vol. 18, no. 3, page 033013, mar 2016. (Cited on page 12.)

[Wang *et al.* 2006] Lingquan Wang, Deli Wang and Peter M. Asbeck. *A numerical Schrödinger–Poisson solver for radially symmetric nanowire core–shell structures.* Solid-State Electronics, vol. 50, no. 11, pages 1732–1739, 2006. (Cited on page 11.)

[Wang *et al.* 2013] D. Q. Wang, J. C. H. Chen, O. Klochan, K. Das Gupta, D. Reuter, A. D. Wieck, D. A. Ritchie and A. R. Hamilton. *Influence of surface states on quantum and transport lifetimes in high-quality undoped heterostructures.* Phys. Rev. B, vol. 87, page 195313, May 2013. (Cited on pages 14 and 17.)

[Wei *et al.* 2017] Di S. Wei, Toeno van der Sar, Javier D. Sanchez-Yamagishi, Kenji Watanabe, Takashi Taniguchi, Pablo Jarillo-Herrero, Bertrand I. Halperin and Amir Yacoby. *Mach-Zehnder interferometry using spin- and valley-polarized quantum Hall edge states in graphene.* Science Advances, vol. 3,

no. 8, page e1700600, 2017. (Cited on pages 2, 22, 23, 130, 132, 133, 134, 136, 138, 139 and 140.)

[Weisbuch & Vinter 1991] Claude Weisbuch and Borge Vinter. Quantum Semiconductor Structures: Fundamentals and Applications, volume 45. Elsevier, 1991. (Cited on page 167.)

[Weiße *et al.* 2006] Alexander Weiße, Gerhard Wellein, Andreas Alvermann and Holger Fehske. *The kernel polynomial method.* Rev. Mod. Phys., vol. 78, pages 275–306, Mar 2006. (Cited on page 203.)

[Werner & Oswald 2020] Daniel Werner and Josef Oswald. *Size scaling of the exchange interaction in the quantum Hall effect regime.* Phys. Rev. B, vol. 102, page 235305, Dec 2020. (Cited on page 135.)

[Wharam *et al.* 1988] D A Wharam, T J Thornton, R Newbury, M Pepper, H Ahmed, J E F Frost, D G Hasko, D C Peacock, D A Ritchie and G A C Jones. *One-dimensional transport and the quantisation of the ballistic resistance.* Journal of Physics C: Solid State Physics, vol. 21, no. 8, page L209, mar 1988. (Cited on pages 2 and 148.)

[Winkler *et al.* 2019] Georg W. Winkler, Andrey E. Antipov, Bernard van Heck, Alexey A. Soluyanov, Leonid I. Glazman, Michael Wimmer and Roman M. Lutchyn. *Unified numerical approach to topological semiconductor-superconductor heterostructures.* Phys. Rev. B, vol. 99, page 245408, Jun 2019. (Cited on page 13.)

[Yamamoto *et al.* 2012] Michihisa Yamamoto, Shintaro Takada, Christopher Bäuerle, Kenta Watanabe, Andreas D. Wieck and Seigo Tarucha. *Electrical control of a solid-state flying qubit.* Nature Nanotechnology, vol. 7, no. 4, pages 247–251, April 2012. (Cited on pages 2 and 129.)

[Zhou *et al.* 2015] Wang Zhou, H. M. Yoo, S. Prabhu-Gaunkar, L. Tiemann, C. Reichl, W. Wegscheider and M. Grayson. *Analyzing Longitudinal Magnetoresistance Asymmetry to Quantify Doping Gradients: Generalization of the van der Pauw Method.* Phys. Rev. Lett., vol. 115, page 186804, Oct 2015. (Cited on pages 19, 166, 173 and 175.)

[Élisée Reclus (1830-1905) Geographe 1905] Élisée Reclus (1830-1905) Geographe. L'homme et la terre - 1905. None, 1905. (Cited on page 1.)

Predictive modeling of nanoelectronic devices

Abstract:

With the progress of mesoscopic physics we have gained precise control over the environment where electrons propagate. Through careful device design we are able to control the electrostatic environment with a precision within the electrons fermi wavelength and draw actual highways allowing for coherent transport over long distances, going as far as hundreds of μm. This allows for experimental physicists to use nanoelectronic devices to probe complex quantum states, such as fractional quantum hall quasiparticles, or to manipulate single electron states. The theoretical modeling of such devices however is not nearly as advanced, there is no single model capable of quantitatively correlating the volts applied at the electrostatic gates to the quantum transport observed experimentally. Most approaches model the quantum particle behavior independently from their electrostatic environment. They account for the field effect of the gates and charges in the device as an effective potential added to the hamiltonian of the system in the form of a fitting parameter. The actual form of the fitting potential is usually derived from analytical calculations or complex semiconductor material models. A single model capturing both the gate field effect and the quantum particle behavior has to correlate the eV physics at the gates to the meV quantum physics. This is not an easy task and requires solving the Schrödinger equation self-consistently with the Poisson equation (Schrödinger-Poisson problem). In this thesis we propose an algorithm and a software (PESCADO) to solve the Schrödinger-Poisson problem. The Schrödinger-Poisson problem is highly non-linear and most approaches to solve it are unstable at low temperatures and near the device regions where the charge density is depleted. We have managed to stabilized our algorithm by developing a method to first find where the non-linearities lie in the energy space and then isolate them s.t. they can be dealt with appropriately. During this thesis we have also developed a model capable of predicting the experimental gate (pinch-off) voltages required to deplete the two-dimensional electron gas beneath quantum point contact devices. We have applied it to predict the pinch-off voltages of 110 experimental quantum point contacts of 48 different designs and to study the effect of disorder in scanning gate microscopy conductance maps.

Keywords: Quantum device, Schrödinger-Poisson, predictive modeling

Modélisation prédictive des dispositifs nanoélectroniques

Abstract:

Grâce aux progrès de la physique mésoscopique, nous avons acquis un contrôle précis sur l'environnement dans lequel les électrons se propagent. Avec un design judicieux des dispositfs nous sommes capables de contrôler l'environnement électrostatique avec une précision de l'ordre de la longueur d'onde de Fermi électronique et de tracer de véritables autoroutes permettant le transport quantique cohérent sur de longues distances, allant jusqu'à des centaines de μm. Cela permet d'utiliser des dispositifs nanoélectroniques pour sonder des états quantiques corrélées, tels que les états de hall quantique fractionnaire, ou pour manipuler des états électroniques uniques. La modélisation théorique de tels dispositifs n'est cependant pas aussi avancée, il n'y existe pas encore un modèle capable de corréler quantitativement les volts appliqués aux électrodes au transport quantique observé expérimentalement. La plupart des approches modélisent le comportement des particules quantiques indépendamment de leur environnement électrostatique. Ils prennent en compte l'effet de champ des grilles et les charges dans le dispositif comme un potentiel effectif qu'on ajoute au hamiltonien du système sous la forme d'un paramètre d'ajustement. La forme du potentiel est généralement dérivée de calculs analytiques ou de modèles complexes de matériaux semi-conducteurs. Un modèle unique capturant à la fois l'effet de champ et le comportement des particules quantiques doit corréler les eV fixées aux électrodes avec la physique quantique qui est généralement de l'ordre du meV. Cela n'est pas une tâche facile et nécessite de résoudre l'équation de Schrödinger de manière auto-cohérente avec l'équation de Poisson (problème de Schrödinger-Poisson). Dans cette thèse nous proposons un algorithme et un logiciel (PESCADO) pour résoudre le problème de Schrödinger-Poisson. Le problème de Schrödinger-Poisson est hautement non linéaire et la plupart des approches pour le résoudre sont instables à basse température et à proximité des régions du dispositif où la densité de charge est depleté. Nous avons réussi à stabiliser notre algorithme en développant une méthode permettant d'abord de trouver où se situent les non-linéarités dans l'espace énergétique, puis de les isoler afin de les traiter de manière appropriée. Au cours de cette thèse, nous avons également développé un modèle capable de prédire les tensions de grille expérimentales(tensions de pincement) nécessaires pour depleter le gaz électronique bi-dimensionnel dans un point de contact quantique. Nous l'avons appliqué pour prédire les tensions de pincement de 110 dispositifs expérimentaux avec 48 designs de grille différentes et pour étudier l'effet du désordre dans les cartes de conductance en microscopie de grille à balayage.

Keywords: Dispositifs Quantiques, Schrödinger-Poisson, Modélisation Prédictive