Universitat de Barcelona

Pràctiques en empresa

QuantumLabUB: Quant2D

Marina Orta Terré

Tutors:
Montserrat Guilleumas Morell
Bruno Juliá Díaz

January 2020

Contents

1 Introduction **3**

1.1 Quant2D 4

2 Methodology and development **5**

2.1 GitHub 5

2.2 Weekly meetings 5

2.3 Chronology 5

2.3.1 Project choice 6

2.3.2 Numerical implementation 6

2.3.3 Graphical interface 6

2.3.4 Changes evolution and final program 7

3 Numerical method **8**

3.1 2D time-dependent Schrödinger equation 8

3.2 Crank-Nicolson: Implicit method for 1D 8

3.3 Extrapolation for 2D and representation 9

3.3.1 Normalization and probability calculation 10

3.4 Verification of the method and parameters used 10

4 Quant2D **13**

4.1 Objective 13

4.2 View 13

4.3 Demos 14

4.3.1 Demo 1 14

4.3.2 Demo 2 14

4.3.3 Demos 3 15
4.3.4 Demo 4 16
4.3.5 Demos 5 17
4.4 Buttons and functions 18

5 Conclusions **19**
5.1 Things to improve 19

6 References **21**

Chapter 1

Introduction

This report contains all the information relative to the project Quant2D developed within the "Pràctiques en empresa" subject during the spring semester of the 2018-2019 academic year and the autumn semester of the 2019-2020 academic year. It has been supervised by Dr. Bruno Juliá and Dr. Munsta Guilleumas and I have dedicated about 250 hours.

The aim of the project is to develop a program that helps the popularizing of Quantum Mechanics knowledge among non-scientific people. This tool is thought as a complement of an oral exposition and is designed to be easily operated by anyone.

The project has been developed simultaneously along with two other made by Arnau Jurado (2dclas) who programmed the classical version of my program and by Manu Canals (JocQuàntic). We all have been working separately but having common meetings along with the teachers in order to check the progress and help each other. All programs are made using the *Python* programming language and with help of a the *Kivy* library for the graphic interface.

1.1 Quant2D

This project shows the time evolution of the density of probability of a quantum system in two dimensions. The application has two screens: a big one where the probability is showed and a small that shows the potential. Both representations are showed trough a gray-scale map. The user can navigate trough different systems and watch the evolution.

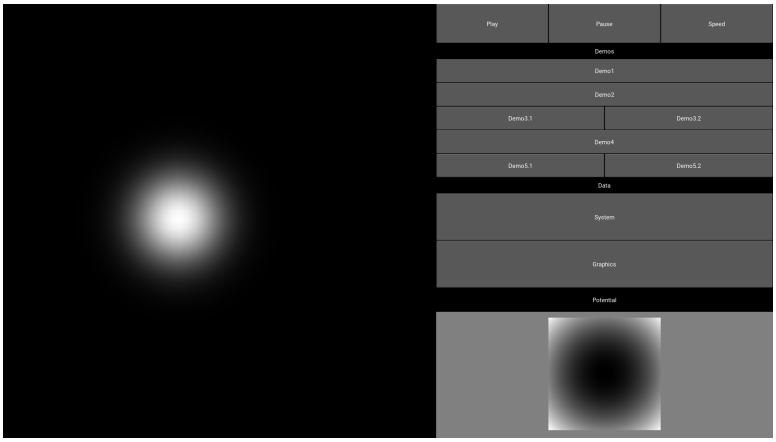

Figure 1.1: Screen shot of the view

Chapter 2

Methodology and development

2.1 GitHub

The way of working has been autonomous and, since each student works on its own project thus each one develops different code, all the progress has been updated using *GitHub*, a web to use *Git* repositories online. The teacher Bruno Júlia owns the *QuantumLabUB* repository where each one has a directory with the project. He is the responsible for accepting or rejecting a *merge* request (a request made by the students in order to change the code for a new version). The benefits of using this platform are that all changes are documented and can be reached even when the code has been changed multiple times. It also shows the changes done per *merge* and the activity of each contributor.

2.2 Weekly meetings

The autonomous work is completed with meetings (one per week) when all the students reunite with the teachers. In these meetings each student shows and checks the progress and proposes new improvements. Also changes and other suggestions are discussed by everyone allowing students help each other with similar problems.

In addition, we had a fluid communication along the week via email.

2.3 Chronology

Brief explanation of the evolution of the project.

2.3.1 Project choice

During the firsts weeks the teachers and students reunited in order to determine what project each student would develop. Since most of the previous projects made (available on the same repository on github) were one dimensional (like most of paradigmatic examples showed in Quantum Mechanics subject), we decided to increase the dimensions showed.

In one hand, Manu had the clear idea to implement a game-like application (in his case would be one dimensional). On the other hand, Arnau and I, went for the two dimensions concept and divided the job: one of us would make a classical point of view (something like the billiards game) and the other would show a 2D quantum particle. Finally Arnau did the classical version and I did the quantum version.

2.3.2 Numerical implementation

Firs of all, I had to implement the numerical method. The equation I was going to solve was the time-dependent Schrödinger equation for a single non relativistic particle in two dimensions. The method would be Crank-Nicolson, a finite difference method that solves partial differential equations. We already studied this method in the Computational Physics subject to solve the 1D heath diffusion equation (in this case we implemented the problem using *Fortran77* language).

The step of going to 1D to 2D is not trivial and I had different problems to implement it in two dimensions and even when I already started working on the interface, I had to correct some parts of the code. To test the quality of the implementation I checked the norm value for each time step of an Harmonic oscillator ground eigenstate. Although the norm doesn't fully remain constant at one (it reduces a little bit its value) its variation is acceptable considering the discretization that introduces the method.

2.3.3 Graphical interface

The graphical implementation was done using the *Kivy* library available in *Pyhton* programming language. This library works with two documents (in this case, both named Box) with different extension: in the *Box.py* document is where all functions and characteristics that the program needs to have must be specified; in the *Box.kv* is where the placement and the functions that each object refers to are specified. Both documents work together and is the least required to work with *Kivy*. In my case I used extra documents in order to separate the numerical part from the graphical implementation.

The graphical interface is based on different objects:

× Button: a click activates a function

× TEXTURE: used to represent the colour maps: the animated (evolution of the probability) and the static (potential applied).

× POPUP: usually used along with a button, is a window opened inside the application with extra information to show.

× LABEL: a static content, usually used to show titles and pictures.

× SLIDER: in my case later replaced with buttons, allows to introduce a numerical value inside a rank.

2.3.4 Changes evolution and final program

The first idea of the program was to let the user decide the initial configuration and the potential applied via sliders but after several tests I saw that the norm was not always conserved and the energy rank needed to be very accurate in order to conserve it so decided to replace the sliders with buttons that show different and paradigmatic demonstrations already set. It is seen in the final graphics that the norm always stays at 1.

After this change, I started improving the interface:

× Add a *speed* button that allows to double the speed and return to the initial value.

× Delete the *Stop* button (resets the animation) and introduce the function it in the initial parameters of the demos. The user can still reset the animation by clicking the same demo that was watching.

× Add information popups:
System button: triggers a popup with a simple explanation of the initial state and the time evolution.
Graphics button: triggers a popup with graphical information. Always shows the energy evolution and the norm evolution, it also shows the deviation or the expected value of the position in some axis depending on each demo.

× Usage of the fullscreen mode: first set on *'True'* but later changed to *'auto'* due to some problems with other computers.

× Center the plot of the potential.

Chapter 3

Numerical method

The equation that needs to be solved in order to represent the time evolution of the density of probability is the time-dependent Schrödinger equation for a single non relativistic particle:

$$\hat{H}\psi(x,y,t) = i\hbar\partial_t\psi(x,y,t) \tag{3.1}$$

3.1 2D time-dependent Schrödinger equation

In this problem, the potential is only function of x and y coordinates, then: $\hat{H} = \frac{\hbar^2}{2m}(\partial_x^2 + \partial_y^2) + V(x,y)$, and the equation will result as:

$$\frac{\hbar^2}{2m}(\partial_x^2\psi + \partial_y^2\psi) + V(x,y)\psi = i\hbar\partial_t\psi(x,y,t) \tag{3.2}$$

The method used to solve this equation is Crank-Nicolson, a finite difference method that solves partial differential equations. This method is implicit in time and is usually used in order to solve heath diffusion problems and similar equations. The method is first studied in 1D case and then is extrapolated for two dimensions.

3.2 Crank-Nicolson: Implicit method for 1D

To solve the equation 3.2 first is solved for the one-dimensional case and then extrapolated to two dimensions. To solve the 1D equation is used an approximation for the derivative expressions adopting a 2-point formula for the first time derivative

and a 3-point formula for the second space derivatives:

$$\begin{aligned} \partial_t \psi &= \frac{\psi(t+\Delta t) - \psi(t)}{\Delta t} \\ \partial_x^2 \psi &= \frac{\psi(x+\Delta x) - 2\psi(x) + \psi(x-\Delta x)}{(\Delta x)^2} \end{aligned} \tag{3.3}$$

Evolve the partial time derivative forward and backward half a step of time for $\psi(t)$ and $\psi(t+\Delta t)$ respectively and equalizing the $\psi(t+\frac{\Delta t}{2})$ terms:

$$(1 + \frac{\Delta t}{2\hbar} i\hat{H})\psi_i^{k+1} = (1 - \frac{\Delta t}{2\hbar} i\hat{H})\psi_i^k \tag{3.4}$$

Where $\psi_i^k = \psi(i\Delta x)$ at $t = k\Delta t$ and is vector-shaped.

Other way to express it might be:

$$\hat{A}\psi_i^{k+1} = \hat{B}\psi_i^k = \overline{\psi}_i^k \tag{3.5}$$

Where $\hat{A}$ and $\hat{B}$ are two tridiagonal matrices and being ψ_i^k the wave function at the current time and ψ_i^{k+1} the wave function after one step of time.

As $\hat{A}$ and $\hat{B}$ are tridiagonal they can be defined using their three main diagonals:

$$\hat{A} = \begin{cases} A_{sup} = r \\ A_{diag} = 1 - 2r + \frac{i\Delta t}{2\hbar} V(x) \\ A_{inf} = r \end{cases}$$

$$\hat{B} = \begin{cases} B_{sup} = -r \\ B_{diag} = 1 + 2r - \frac{i\Delta t}{2\hbar} V(x) \\ B_{inf} = -r \end{cases}$$

Both deduced using 3.3 and 3.4 and being $r = \frac{i\hbar\Delta t}{4m(\Delta x)^2}$

The solution ψ_i^{k+1} is found using the tridiagonal matrix algorithm or Thomas algorithm.

3.3 Extrapolation for 2D and representation

Working with two dimensions, the wave function is be matrix-shaped, being $\psi_{i,j}^k = \psi(i\Delta x, j\Delta y)$ at a time $t = k\Delta t$. Also $\hat{A}$ and $\hat{B}$ are three-dimension tensors. To solve the problem is used the same tridiagonal matrix algorithm as the one-dimensional problem twice: evolving first for x and then for y in each time step.

In this specific case, where the grid taken is uniform for both axes (same as saying $\Delta x = \Delta y$), the constant r that composes both three-dimension tensors will be the same for the two evolutions and only one $\hat{A}$ and one $\hat{B}$ tensors are needed.

To show the evolution in an animation, each frame will correspond to a specific time step of the evolution and the frame will correspond to the probability matrix. This can be represented as a colour map.

3.3.1 Normalization and probability calculation

This numerical method is based on unitary operators that maintain normalized the wave function. The fact that the norm keeps the value trough the time evolution is a good indicator of a well implemented method. The norm that has to be conserved is:

$$\sum_{i,j} \psi^*_{i,j}\psi_{i,j}\Delta x_i \Delta y_j = \Delta x^2 \sum_{i,j} |\psi_{i,j}|^2 = 1 \tag{3.6}$$

This leads to the calculation of the probability matrix:

$$\rho^k_{i,j} = \psi^*_{i,j}\psi_{i,j} = |\psi^k_{i,j}|^2 \tag{3.7}$$

And the formula to compute the probability accumulated in a certain area would be:

$$P\{(x,y) \in [x_0, x_1] \times [y_0, y_1]\} = \Delta x^2 \sum_{i_0,j_0}^{i_1,j_1} |\psi_{i,j}|^2 \tag{3.8}$$

About units of measure: The calculus are made using 'natural units' which means: $\hbar = 1$ and $m = 1$. Since the project is all about showing the qualitative behaviour, the units are not shown in both the report and the final program.

3.4 Verification of the method and parameters used

In order to fulfill the convergence condition, $|r| \leq 1/2$ so Δx and Δt cannot be any value.

Taking a constant $\Delta x = 0.05$ and testing the method for different Δt, the initial and final norms are compared. The evolution that is tested corresponds to an harmonic oscillator type of potential with frequency $\omega = 1$ and the wave function is the ground eigenstate relative to its potential so, in this case, the wave function is static so is the density of probability.

This are the values extracted being T the number of time steps so the total time is always a unit:

Δt	T	n_i	n_f
0.01	100	0.99996	0.96301
0.005	200	0.99996	0.98131
0.0025	400	0.99996	0.99059
0.002	500	0.99996	0.99246
0.00125	800	0.99996	0.99527
0.001	1000	0.99996	0.99621

After this test, the chose has been $\Delta t = 0.0025$ since is the balanced value between norm conservation and computing time (smaller Δt implies more steps to reach the same amount of time and being too large for the smallest steps).

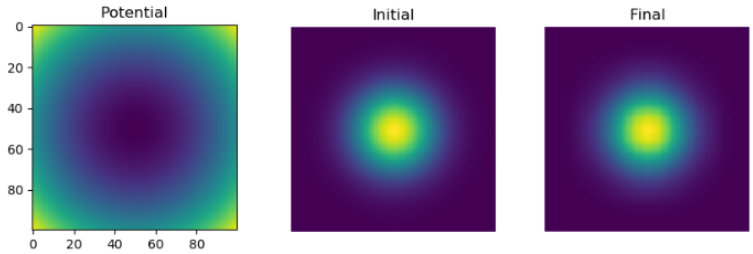

Figure 3.1: Evolution for $\Delta t = 0.0025$. As expected, the probability remains static.

Taking a squared system and being L the side and N the number of points of the discretization, the final parameters used in all the computations are the following:

L	N	Δx	Δt
5	100	0.05	0.0025

For all the cases implemented, and after a correction in the implementation of the numerical method, the norm remains constant at 1:

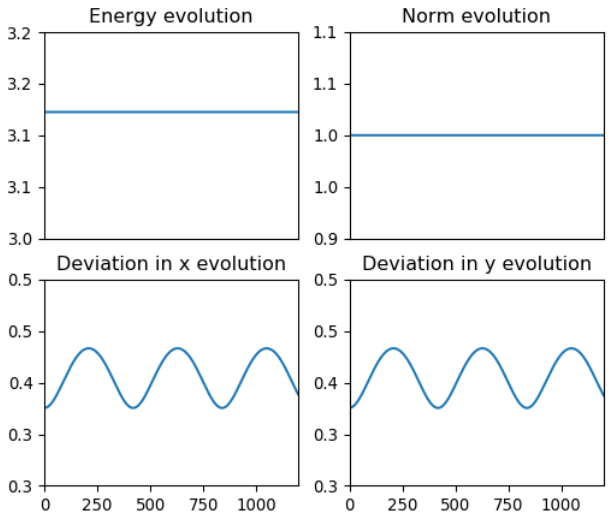

Figure 3.2: Example of the graphics showed. This case corresponds to the demo 2

The norm remains constant after 1260 steps of time.

Chapter 4

Quant2D

4.1 Objective

The final aim of the application is to show how Quantum Mechanics works via different demonstrations of paradigmatic cases where the behaviour differs from the classical point of view.

4.2 View

The program has a simple interface to make it easy to use for anyone.

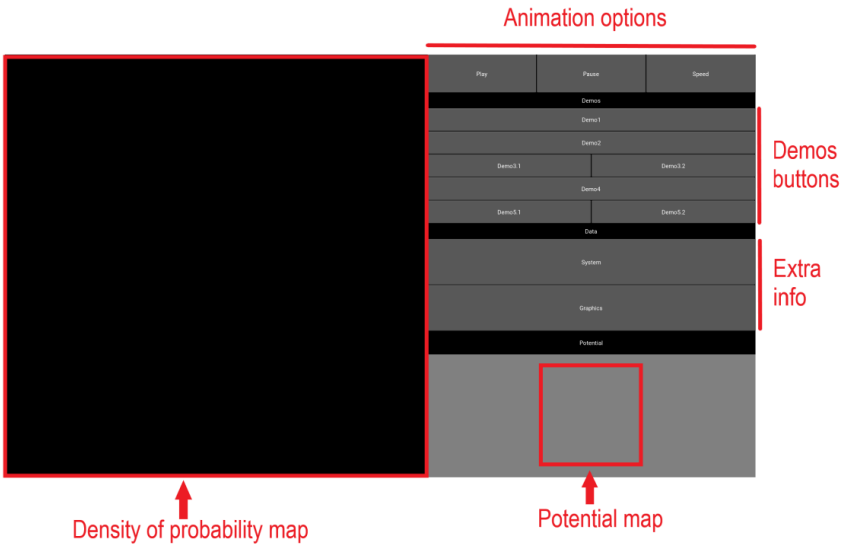

Figure 4.1: First view and distribution

The time evolution for each case is showed in the density of probability map as a

loop animation. Each frame corresponds to a time step of the numerical resolution. In addition, the application has simple animation controls (play, pause and change the speed) and extra information of each demo to help understand the behaviour (graphics that show the evolution of different quantities and a little explanation).

4.3 Demos

Schematic explanation of the systems used for each demonstration and the behaviour that can be observed.

4.3.1 Demo 1

- **Potential**: Harmonic oscillator type of potential.

- **Initial wave function**: Ground eigenstate of the potential but off-centre in relation to the potential.

- **Evolution**: The wave function oscillates around the center of the potential without change its shape.

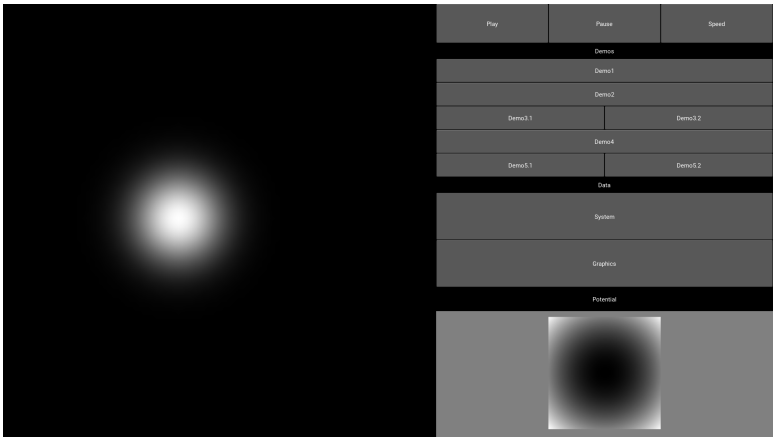

Figure 4.2: Screen shot of the demo 1

4.3.2 Demo 2

- **Potential**: Harmonic oscillator type of potential.

- **Initial wave function**: Centered ground eigenstate relative to an harmonic oscillator potential with bigger frequency than the actual potential.

- **Evolution**: The wave function pulsates without moving its center.

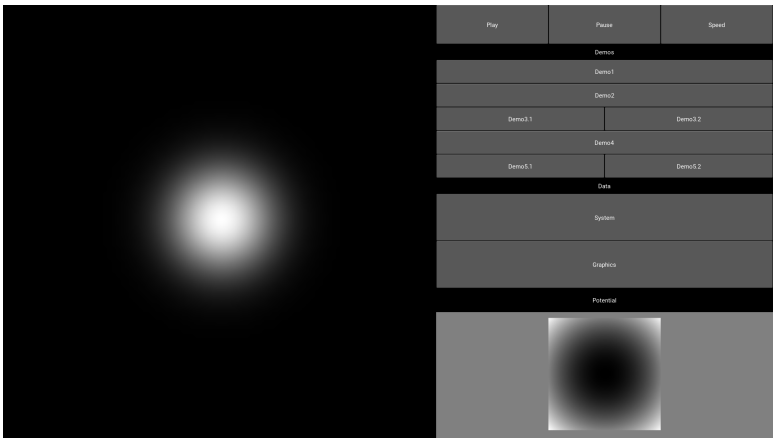

Figure 4.3: Screen shot of the demo 2

4.3.3 Demos 3

Two demos of a similar system.

Demo 3.1

- **Potential**: Centered square barrier half the size of the screen. The potential value is the same in every edge of the square being five times the energy of the wave function and zero in the other points.

- **Initial wave function**: Centered Gaussian that fits into the square of the potential.

- **Evolution**: The wave function expands until it sees the square. Even when the barrier is higher than the wave function, it can escape and exists the probability of measuring it outside the box.

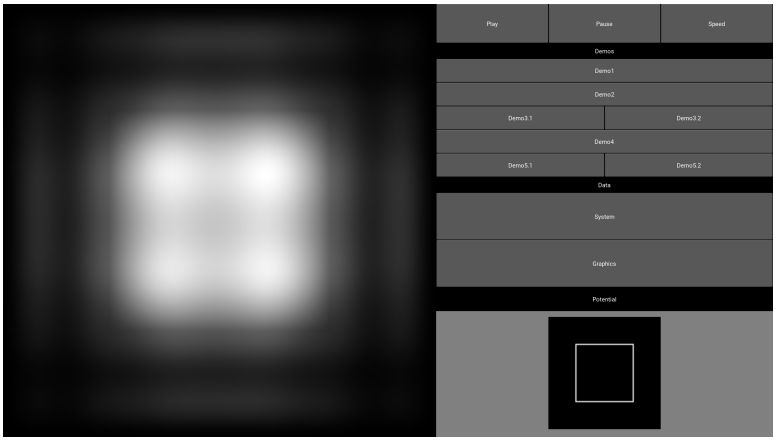

Figure 4.4: Screen shot of the demo 3.1

Demo 3.1

- **Potential**: Centered square barrier half the size of the screen. The potential value is the same in every edge of the square being fifty times the energy of the wave function and zero in the other points.

- **Initial wave function**: Centered Gaussian that fits into the square of the potential.

- **Evolution**: The wave function expands until it sees the square. The barrier is much higher than the wave function and it can barely escape. The probability of measuring it outside the box is much lower but it is still possible.

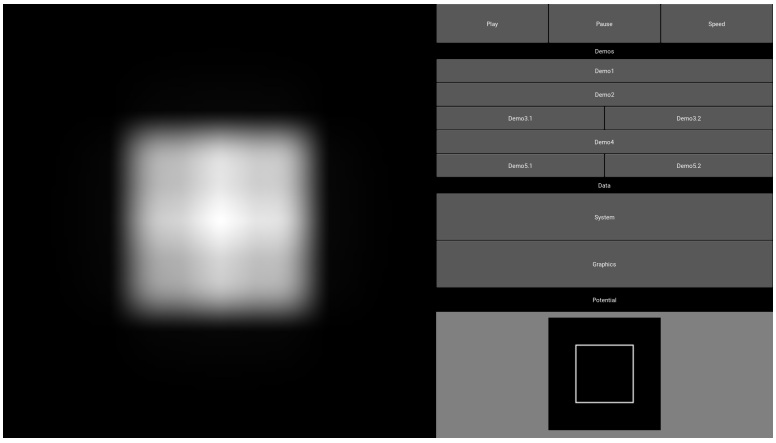

Figure 4.5: Screen shot of the demo 3.2

4.3.4 Demo 4

- **Potential**: Harmonic oscillator type of potential with different frequency in each axis.

- **Initial wave function**: Centered ground eigenstate relative to an harmonic oscillator potential with same frequency in both axis and equal to the x axis of the actual potential.

- **Evolution**: The wave function pulsates with different frequencies in each direction and without moving its center.

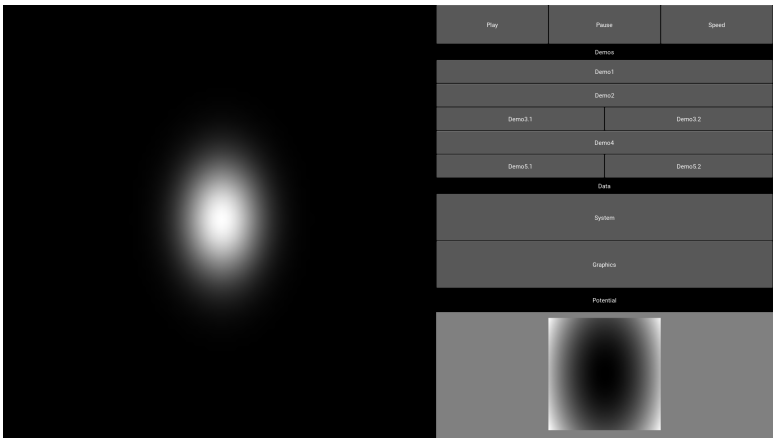

Figure 4.6: Screen shot of the demo 4

4.3.5 Demos 5

Demos of the single and double slit.

Demo 5.1

- **Potential**: Single slit: vertical barrier with a hole in it.
- **Initial wave function**: Gaussian with momentum towards the barrier.
- **Evolution**: The wave moves towards the barrier and diffract.

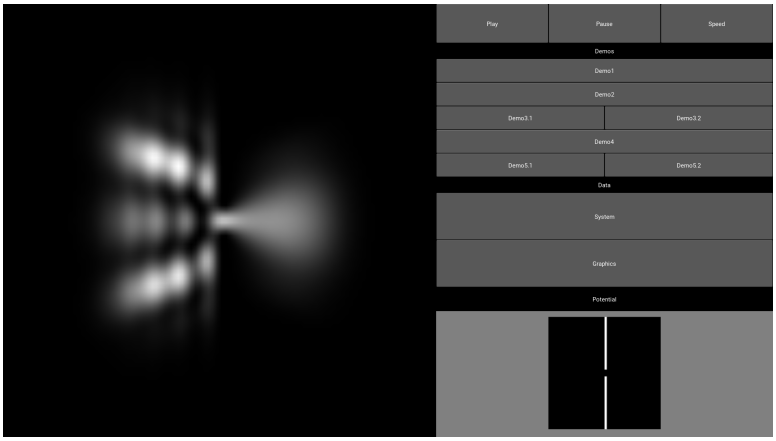

Figure 4.7: Screen shot of the demo 5.1

Demo 5.2

- **Potential**: Double slit: vertical barrier with two holes in it.
- **Initial wave function**: Gaussian with momentum towards the barrier.

- **Evolution**: The wave towards the barrier, diffract and cause interference since the probability of going trough a slit is equal to the other.

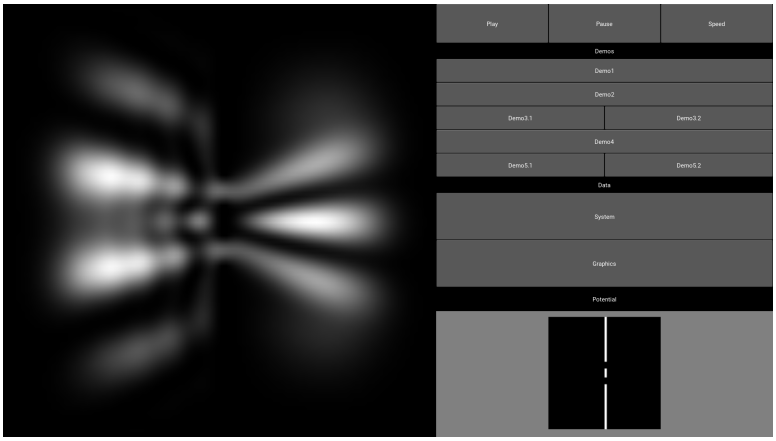

Figure 4.8: Screen shot of the demo 5.2

4.4 Buttons and functions

- **Speed**

The speed button changes the speed of the animation. The speed has two modes: the initial speed and the double of the initial speed. When first clicked the button the speed is doubled and when is clicked again it returns to its initial value.

- **Demo buttons**

Each demo has its relative button. Each time one of this buttons are clicked the screens change to the initial state of the wave function and the potential relatives to it. Once a demo button is clicked, the application auto pauses (if the animation was running) and the speed returns to its initial value.

- **System button**

When this button is clicked a brief description of what is represented is showed through a popup. Mainly it is explained what is the potential, what initial wave function is taken and the evolution trough the time expected. The reason why this button was finally created is to have a support and to help understand what happens in cases where no oral explanation is held.

- **Graphics button**

When this button is clicked different graphics are showed via popup. For each demo, a different set of graphics is represented. In all cases, four pictures are showed: always the evolution of the energy and the norm through time is viewed and two more depending on the demo that are the deviation for x or y and the expected value of x or y coordinates.

Chapter 5

Conclusions

At the end of the year, I got a project that can be used by anyone and that shows the behaviour of Quantum Mechanics accomplishing the main objective.

Taking part on this project I learned different things. On one hand, I learned a lot about how to work in a joint project and also how to present and expose the changes or improvements and the results of a project. On the other hand, I strengthened my knowledge of Quantum Mechanics and numerical methods. Also my programming skills got much better and I learned how to use the *Kivy* library to program a graphical interface.

Even though I had to change the initial program idea in order to be able to show the quantum behaviour of a particle with norm conservation, the objective of creating a program to show the operation of Quantum Mechanics is fulfilled, so I can say I am satisfied with the final program. Personally, I think the best use of the program would be as a teaching support that illustrates the concepts explained. Nevertheless, the addition of an explanation button with short descriptions allows a better autonomous usage.

This program can be polished in different aspects and I have enumerated some changes or improvements I would have done if had more time:

5.1 Things to improve

× **Language button**

Provide a language button that, at least, changes the explanations of each demo from the *System* button into other languages.

× **Squared screen**

Work on the *fullscreen* function to make a view that adds black stripes to keep the proportions rather than auto adjust to the size of each screen.

× **Add new demos**

It is always good to add new examples that use different types of potential or wave functions. It could be a good addition to show a wave function that represents an excited state rather than a ground state as usually showed.

× **Add a second mode**

The program now only offers the view of the evolution of different paradigmatic cases. It could be good to add a mode where the user customs the system always being able to change from this 'free' mode to the 'demo' mode whenever is wanted.

In order to successfully implement this improvement, the time of computation needs to be highly considered.

Chapter 6

References

× **Github repository *QuantumLabUB***

`https://github.com/brunojulia/quantumlabUB`

× **Quant2D project inside the repository**

`https://github.com/brunojulia/quantumlabUB/tree/master/Quant2D`

× **Python documentation**

`https://www.python.org/doc`

× **Kivy documentation**

`https://kivy.org/doc/stable`

× **Numerical method**

- Personal notes from the Computational Physics subject taken in autumn semester of 2018-2019 academic year and given by Bruno Júlia.

- Code from the *Doubleslit* project: a previous project inside QuantumLabUB program:

`https://github.com/brunojulia/quantumlabUB/tree/master/doubleslit`

- Other Thomas algorithm information:

`https://www.algorithm-archive.org/contents/thomas_algorithm/thomas_algorithm.html`

`https://en.wikipedia.org/wiki/Tridiagonal_matrix_algorithm`