Memòria de Pràctiques

Tunneling

QuantumLab UB

Febrer 2020 - Octubre 2020

Júlia Cabrera Cortada

Tutors: Bruno Juliá, Carles Calero i Muntsa Guilleumas

Entrega de l'informe: 17 de Febrer 2021

1 Introduction to the project

1.1 Motivation

At the beginning of the 6th term of the grade, I was really interested in getting to know the working possibilities for a physicist and experimenting how physicists work outside the classrooms and laboratories.

A great oportunity to do this was presented to me in February 2020, when Bruno Juliá, my *Física Computacional*'s teacher wrote an email asking if some of the students were interested in doing *Pràctiques a l'empresa* with him within the project *QuantumLabUB*. After some e-mails and a meeting, I was chosen for the project together with two other students; Jofre Vallès and Adrià Bravo.

1.2 QuantumLabUB

This project was iniciated in February 2018 under the supervision of Bruno Juliá and Muntsa Guilleumas, and it was built by undergraduate students developing applications with the following goals:

- Popularizing quantum physics
- Illustrating realistic quantum mechanical problems
- Working under the GitHub platform,in the QuantumLab repository.

We were then asked to present an idea for an application involving a realistic quantum problem with some kind of interaction for the popularizing purpose, like showing a problem in a high variety of cases or building a game on top of a problem.

1.3 Initial idea of the game

At the beginning, a precise idea of what our game was going to be like was not necessary. However, we had to have a general idea about what physical phenomenon would be its base. So I decided to make a game about quantum tunneling and this allowed me to program all the mathematical basis without having a precise idea about the game, which would be constructed slowly later on.

The idea of the game was at first to carry a particle from one side to the opposite side of a table with various potential barriers in between. The particle would have had to cross these barriers with the greatest possible probability, and so to get to the other side of the table having the maximum probability of having really reached there.

Later on, I saw that this game had no difficulty at all, because it was eventually very trivial to change wave function and barrier parameters in order to achieve a good probability of having crossed the potential barrier.

1.4 Final version of the game

After a lot of changes, the game has ended up as following; Two particles compete with each other to reach the end box of the table alive and with the maximum possible score. Each box has a target probability assigned, and when a particle crosses a potential barrier to enter the box, it has to change all the necessary parameters in order to get as close as possible to the target probability of having crossed the barrier. According to the error commited, the particle will be given more or less points.

Also, the particles have to try and pass through boxes in which the target probability is high, because in every box a portion of the particle is taken: the lower the target probability, the bigger the taken portion is.

The aim of the game is to gain some intuition about quantum tunneling and how the barrier and wave function parameters affect to the probability of eventually crossing the barrier.

1.5 Development of the project

The team was integrated by Bruno Juliá, Carles Calero and Munta Guilleumas as supervisors and the three undergraduates (including me) mentioned before. We all met once a week virtually as the internship period was during the confination. In these meetings each of us exposed our progress, asked questions and answered questions from the teacher, explained future ideas and recieved advice and feedback from the supervisors. Sometimes feedback was given between the undergraduates, which was also very enriching.

The meetings allowed us to keep up with the rythm of working that the supervisors demanded. Knowing that every week I had to make some progress and present it in a meeting made me work with a certain constancy. However, we didn't have a strict timetable. We could dedicate more time to the project during weeks in which we had no exams, and leave it in stand by when a lot of exams came from other subjects.

Every week, to present our progress, we had to upload or code to the GitHub platform. This platform has a principal branch controlled by Bruno and then each one of us had their fork, in which we uploaded the code each week, so that others could try our results in their computers. But before that, we had to ask to merge our code to the principal branch. Then Bruno confirmed the pull request and the code was shared.

Apart from that, we all kept a diary in which we wrote all the work we had done during the week; Changes in the code, new ideas, problems we had had and maybe solutions we had found, etc. Before each meeting, we sent this diary to all the team so that they could read it before the meeting took place. This document has been very useful when writing this memory and also every week for the rest of the team to refresh their memory about the project.

2 Physics basis

2.1 Evolution

The problem to solve was to compute the temporal evolution of a wave function. We initially had a wave function $\Psi(x,0)$ with a certain momentum and inside an infinite potential box with a finite potential barrier in the middle of it.

The equation to solve was therefore the Schrödinger equation with dependence on time:

$$-\frac{\hbar}{2m}\frac{\partial^2\Psi}{\partial^2 x}+V(x)\Psi=i\hbar\frac{\partial\Psi}{\partial t} \tag{1}$$

Cranck-Nickolson method was used to solve this equation and obtain the temporal evolution of the wave. The position space was discretized and therefore the second derivative of the wave function, which leads to an expression of the hamiltonian operator as following:

$$H\Psi_i=\Psi_i(V_i+\frac{\hbar}{m\Delta x^2})-(\frac{\hbar}{2m\Delta x^2})(\Psi_{i-1}+\Psi_{i+1}) \tag{2}$$

Also dicretizing the temporal derivative and organizing terms we obtain a matrix with three diagonals which allows us to obtain Ψ_{j+1} from Ψ_j, where j is the time index.

This last process is carried out by a function which is already programmed in the Wikipedia. In this function, the three vectors a, b, c that form the three diagonals are entered, as well as vector d which is the product between Ψ_j and the hamiltonian operator.

So every step of the Cranck-Nickolson method consists of solving the tridiagonal problem and imposing the boundary conditions, this is, imposing infinite potential at the limits of the box.

2.1.1 Norm and energy. Checking.

At the beginning Crank-Nicholson method was checked by calculating the norm of $\Psi(x,t)$ at each step to make sure everything was correctly calculated. The norm was calculated using the following expression:

$$\int_a^b \Psi^*\Psi dx \tag{3}$$

Where obviously a and b are the limits of the box and Ψ is at a given time. Simpson method for integration was used to calculate the norm and all the other integrals involved in the calculus of the game.

The energy was another parameter which was quite useful for the checking of the method as in each step the energy had to be conserved as well. The expression used was:

$$E=\int \Psi^* H\Psi \tag{4}$$

I can remark that in the early stages of my project these quantities were all I relied on to know if I was doing correctly, and until they were not perfectly conserved I didn't stop improving my code.

2.2 Probability of crossing the barrier

2.2.1 Integrated probability

To know what the probability was for the wave to be on the other side of the barrier at a given time, the same integral used to calculate the norm had to be calculated, but now taking only into account the terms of Ψ at the left of the potential barrier. This is, integrating the wave function at a given time in the space interval corresponding to the other side of the potential barrier.

2.2.2 Wkb prediction

The WKB solution to the Schrödinger's equation for a finite potential barrier provides an expression for a transmission coefficient T which only depends on the initial energy of the wave function:

$$T(E) = e^{-2\int_{x1}^{x2}\sqrt{2(V(x)-E)}} \tag{5}$$

Where x_1 and x_2 are the two points where the potential of the barrier equals the energy of the wave. The function that calculates this coefficient finds these two points and applies the formula to all the points of the barrier which are found between them. The integral is again calculated with the Simpson method.

2.2.3 Comparison

A study over the efficiency of the WKB prediction was done. The probability for the wave to have crossed the barrier was studied and compared to the WKB prediction, varying some barrier parameters See Figure 1 for some of the results of this comparison.

3 Kivy

To do the interface of the game a library called kivy was used. Kivy is an open source Python library for the rapid development of applications that use innovative user interfaces.

A kivy code works with two files; generally, in one file there are all the functions defined inside their respective classes, whereas on the other file (.kv), all the layout of widgets such as buttons and sliders are designed, but widgets can also be created from the first file, also inside their corresponding class. But despite this, the file with extention .kv is always necessary to build the application with the classes that have been created.

All the features and tools from kivy that were learned or added to the code will be explained in the following pages chronologically.

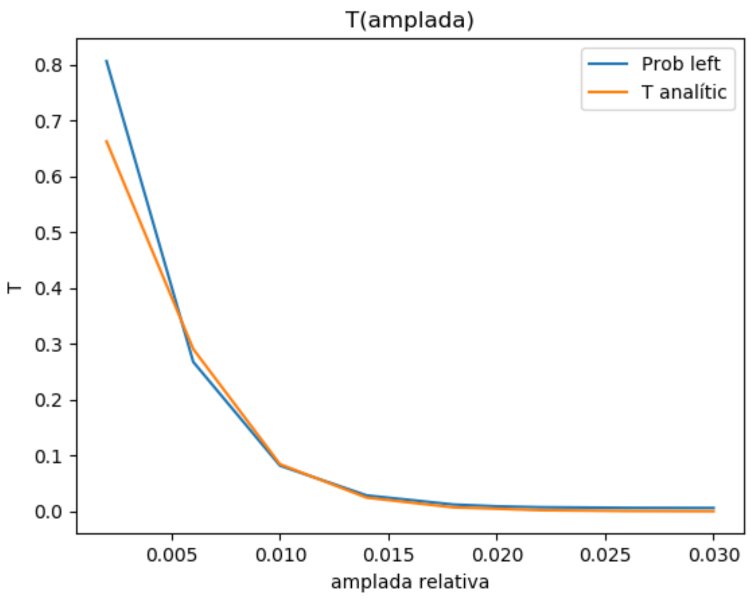

Figure 1: This graphic shows the integrated probability for the wave having crossed the barrier and WKB transmission coefficient variation in front of the width of the potential barrier relative to the dimension of the box.

3.1 Layouts

Before adding any element to the application, it was important to understand how elements were organized. There are several Layouts which can be used, and they are all seen different and as programmed diferent as well.

However, I have mainly used one of them, which is called BoxLayout as it is easy to understand and therefore easy to move and change sizes from the widgets. It follows the following rules:

- Once defined the orientation, horizontal or vertical, widgets are added in the order they are written inside the code.This has to be written in the .kv file.
- The size of the item, if not specified, is determined by the division of the available space by the number of widgets added.
- Boxes can be given an identification name so that they can be referenced from the .py file, and add widgets from there. This is in some cases very interesting, for example when we want to show a plot. Then the size can be modified multiplying by a number the size that would be given to that widget.

3.2 Evolution animation

Before implementing the evolution, I had to learn how to show a graphics in kivy. Firstly, a box without any Button or Slider was needed. Then, a figure was created as in matplotlib but not drawn. It was drawn by kivy, more specifically the renderer in kivy, called canvas.

So a figure was created and all its elements were added, and then this figure was passed to canvas in order to be able to draw it.

After knowing how to make a plot, I learned how to make an animation: with the *Clock*'s method *Scheduleinterval*, a function can be called every a determined period of time. So I created a function that from a given $\Psi(x,t)$ calculates $\Psi(x,t+dt)$, and therefore, relates a real step of time with a Cranck-Nickolson step.

In every step, apart from calculating the new data of the plot, the precieding plot had to be removed, in order not to have all the plots drawn ones above the others.

3.3 Changing parameters. Buttons and Labels

Once achieved the evolution animation, I learned how to make a Button in the .kv file and to call a function from the .py file every time the button was pressed (Or released). Then I started to make functions that would change some parameters from the potential barrier or the initial wave function. I added then buttons to add and subtract weight and height to the barrier, and functions to add or subtract initial momentum or change de sigma of the initial gaussian form of the wave.

All these changes could be applied either before starting the evolution or during it. This was a very useful property of *Clock*: as this function was doing Crank-Nicholson steps in real time, if the Hamiltonian changed during the evolution, the evolution would be adapted to that change.

To enrich this evolution, I also wanted to show some parameters values. It was then when I learned how to add a Label in a Box, but from the .py file, so that the content of the label could be refreshed whenever I wanted.

I added a Label with the norm of $\Psi(x,t)$, which was calculated every several steps of Cranck-Nickolson and the Label was actualized at the same time. I also added labels for the integrated probability at the other side of the barrier so that the probability of $\Psi(x,t)$ having crossed the barrier could be known at every moment. A Label with the WKB prediction was also added, but it was only refreshed when a barrier parameter or initial momentum of the wave was changed, as this coefficient doesn't depend on time.

3.4 Figures

The idea of my game was for two particles to compete one with the other. So the first I had to learn was how to draw a circle with the size I wanted in the place I wanted. This is done with canvas and importing the library *kivy.graphics*.

Then the color, size, position and many other parameters can be specified from the figure wanted. When the position is specified, it is referred to the center of the circle and the size to the radius.

I also had to learn to draw lines in order to construct my table of game, in which the particles would be moving during the game. It is done the same way as a circle, but to specify the positions, the coordinates of the two limit points have to be specified.

3.5 Screens and Popups

After having all the items and the game, I wanted to do a Starting Screen with a Menu so that the user could choose to play the game or not. This is shown in figure **??**. To do that, I had to create a class called ScreenManager which allowed me to define the different screens that I wanted to add. Simple screens were another type of class. The transitions between screens had to be defined inside the class from which the transition was done; for example, the transition from the menu screen to the gaming screen had to be defined inside the menu screen class while the inverse transition had to be defined from the gaming screen. Finally, I have used another type of class called Popup, which is like a window which can be opened and closed inside a screen. I have used it to write the game instructions so that in any moment the players can consult how the game is played.

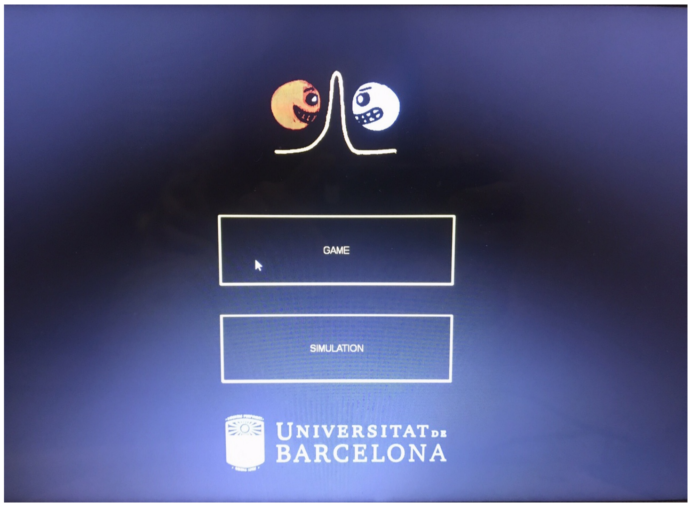

Figure 2: Gaming Screen

4 The game

4.1 Initiation of the game

In this section, the initial screen of the game is described. All the items listed below can be seen in Figure 4.1

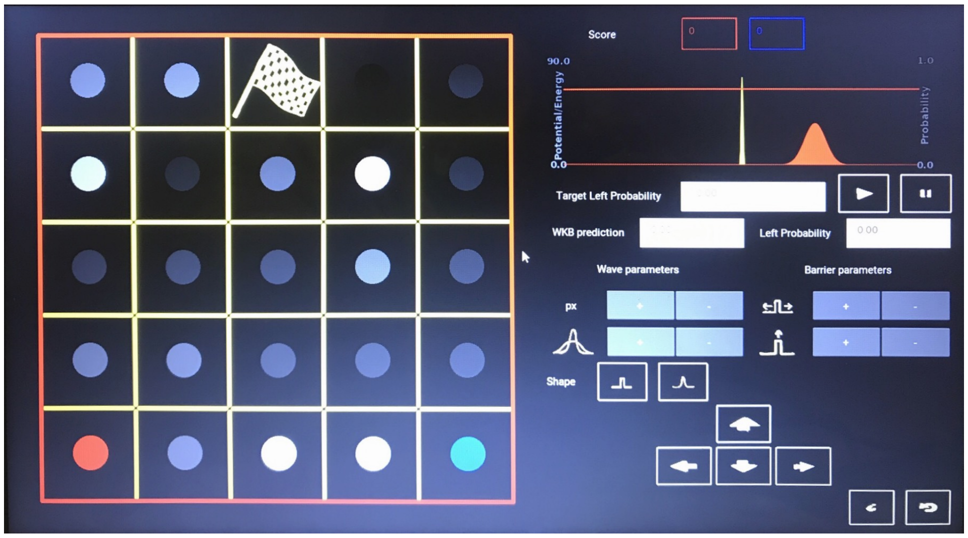

Figure 3: Gaming Screen

1. **Labels creation:** When the gaming screen is initiated, all the necessary label screens are created to display all the numbers that contribute to the game. More specifically, five Label screens are created; two for the scores of the two particles, and the rest are for the integrated probability of the wave across the barrier, the WKB transmission coefficient and the target probability.

2. **Evolution screen:** The evolution screen is also created and the first plot is drawn, with $\Psi(x, 0)$ and the potential barrier, as well two axis, one for the potential barrier and one for the wave function.

3. **Buttons:** All the buttons needed to control the game are also created. There are a set of buttons to control the movements of the particles and another set for the control of the parameters of both the potential barrier and $\Psi(x, t)$. Each of this buttons is bind to functions. Some of them will be explained later on.

4. **Game table:** There are also a group of functions that draw the game table and all the elements in it. There's a function for all the lines that represent the potential barriers and that are the basis of the table. Then another function draws a white circle inside each box, which represents

the target probability assigned to that box. Every time a particle wants to move to a particular box crossing the potential barrier, they will have to try to achieve a probability of having crossed as similar as possible to the target one. This target probability is reflected on the transparency of the circle; if the particle is white, the target probability will be near 1.0, whereas if it is dark gray, that means it will have a very low target probability.

5. **User particles**: The two players will control each one a particle, which will move through the game table crossing potential barriers. These two particles are also drawn at their initial positions at the beginning of the game.

4.2 Game development

4.2.1 Game Goal

The goal of each player is to arrive to the end box with the maximum score possible and with the maximum life.

- **Score:** When a player has to cross a barrier to pass to the following box, a target probability of crossing is given to the player. Then they have to change parameters of both the barrier and the wave function so that the portion of the wave that crosses the barrier is as similar as possible to the target probability.
- **Particle's life:** Every time a particle passes from one box to another, a portion of the particle is taken, inversely proportional to the target probability. So what would be interesting for the player is to find the path which is shorter until the end box, but also that goes through boxes with a high target probability. The particle's life is reflected on the transparency of the particle itself. The particles will have to try arriving to the end box without loosing too much of their lives, if not they will die. Life will be also taken in account if the score of the two particles at the end box is the same.

4.2.2 Playing

Now we will imagine it is the red particle's turn and we will explain all the steps the user has to take, as well as all the functions used in the code and all their utilities.

1. **Moving the particle:** The first step the player has to carry out is moving the particle from box. As we have said before, the particle will have to try to move to the box with the highest target probability sorrounding its actual box. There are four buttons, each one for each directions the particle can take and they are binded to a correspoding function. This function does the following:

- Changes the position of the particle whose turn is the actual.
- When drawing the particle in the new position, it identifies the target probability in the new box and subtracts transparency to the particle according to its value.
- Controls that the direction buttons have been already pressed, so that the player cannot press them again until the turn is given to the next particle.
- Identifies the random initial parameters given to the potential barrier (width and height) and refreshes the image of the plot.
- It writes the value of the WKB coefficient according to the initial values of the barrier parameters.

2. **Changing parameters before evolution:** The next step for the player is to try to change the initial conditions of $\psi(x,t)$ in order to get close to the target probability. The WKB coefficient will give an idea about how close the player is from it. The functions binded to the buttons that change the value of the parameters do the following:

   - Firstly, they change the parameter's value
   - They recalculate all the things that are affected by that change of parameter. For example, if the player changes the initial momentum of the barrier, the energy is recalculated.
   - Redraws all the items that are seen in the plot and that have been changed. Once again, if the initial momentum is changes and therefore energy, the energy in the plot will be redrawn.

3. **Start the evolution**: After the player has changed the value of the parameters and is ready to start evolution, they have to press the corresponding button and make the animation start. This button is binded to the function that uses *Clock* to start the animation.

4. **During the evolution**: During the evolution the player will be able to see the integrated probability at the other side of the barrier every very little periods of time. The player will also be able to change the barrier parameters in order to get closer to the target probability.

5. **Stop the evolution**: Once the player is satisfied with the portion of the wave function in the other side of the potential barrier, they can press the button that will stop the evolution. This is a very important button, as it is binded to a very important function. This function carries out the following tasks:

   - Calculates the error commited by making the difference between the target probability and the integrated probability on the other side of the barrier.

- Calls a function which assigns a score to the particle according to the error commited.
- It writes this score in the corresponding label screen, this is, the score screen of the red particle in this case.
- Resets all the parameters of the barrier and wave function back to the initial ones, and recalculates the points of the potential barrier and $\Psi(x,0)$
- Redraws the plot so that it is seen as the initial one.
- Changes the turn to the other particle. This implies some graphical changes to the game table color and to the wave function color in the plot.
- It resets the WKB coefficient value as well as the target probability and the integrated probability value.
- Finally, it checks if both of the particles are in the end box. If they are, then it calls the function that ends the game.

4.2.3 End of the game

When the function in charge of the end of the game is called, it means the game is over. What does that function do?

- It compares the scores of the two particles and decides which is the winner according to the highest score.
- If they have the same score, it compares the lives of the two particles and again decides which one is the winner.
- Once decided the winner, it opens a little Popup in which there is written the winner name. Inside this Popup there's a button which is bind to the transition from the gaming screen again to the starting screen.

4.2.4 Instructions:

The instructions of the game can be found when a button is pressed in the gaming screen. This button opens a Popup in which the basis of the game is explained and how it is played. It can be opened anytime during the game.

5 The simulation

In the starting screen users can choose to go to the gaming screen, where they can play the game, or go to the simulation screen, shown in Figure 5, where they can just see the evolution animation of $\Psi(x,t)$ and see some significant values and change some parameters:

- The norm of the wave function is actualized every very little intervals of time.

- The integrated probability at the other side of the barrier is also actualized at the same time as the norm
- The value of the WKB coefficient is also shown and refreshed every time a value is changed.
- There are buttons to change all the parameters that can also be changed in the game: barrier width, height and shape, and the wave function initial σ and momentum.
- At any time the evolution can be stopped and reset and started again.

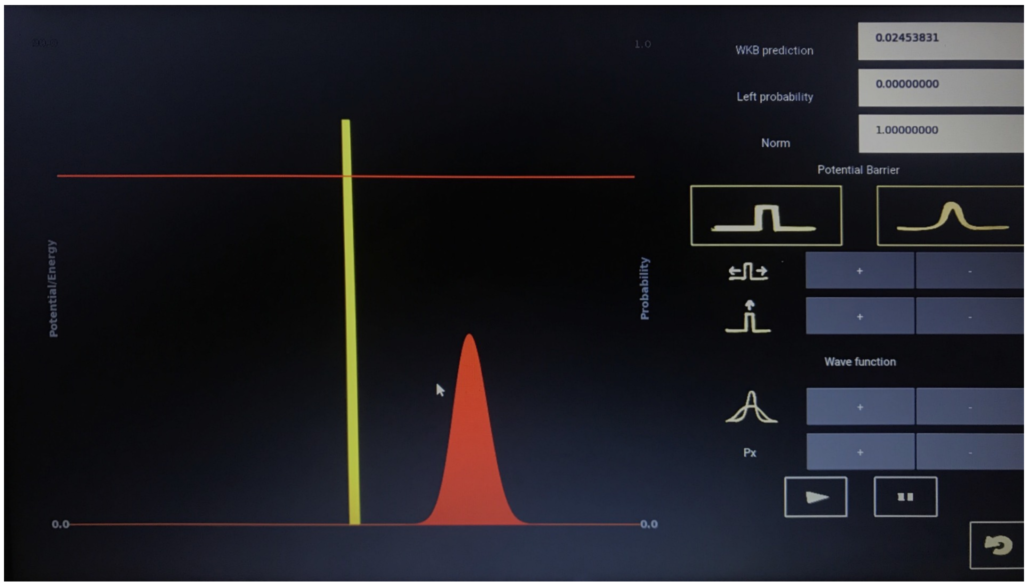
WKB prediction 0.02453831
Left probability 0.00000000
Norm 1.00000000

Figure 4: Gaming Screen

6 Conclusion

Firstly, I have to admit that this experience was highly enriching for me; not only because I profoundized on my knowledge in python and learned how to use kivy or because I learned quite a lot about computational methods of solving a physics problem. I learned a lot about how to work in a team, at the same time that I was learning to lead my own project. I had to work with a certain constancy to evolute in harmony with the team, at the same time that I had to organize myself to construct slowly but strongly my project, combining it with all the other subjects I was dealing with during that semester.

It has not been an easy task, partly because I had not heard about kivy before and I had to learn it all by myself with very few help from the basic documentation there is from it. Also, because every week I had to keep up with the expectations of the tutors, whatever the time needed for that was.

Finally, I am proud of the projects we have made; I think that making a game with a real physics problem but using it in a funny way to make a game is a very good idea to divulgate physics; The interface of the game attracts users at the same time that they are seeing and learning about something that goes beyond simple rules of a game; there are real mathematics being calculated in real time and physics phenomena taking place.