Simulating the Schrodinger Equation with Quantum Computing

Berta Moreno
Tutor: José Maria Gómez and Bruno Julia

April 25, 2023

1 Aims

I worked with Institut de Ciències del Cosmo de Barcelona during July and the beginning of September 2022, with the aim of learning about quantum computation. I worked independently, and met with my tutors weekly to discuss my progress. I started by reading about quantum computation in general, before focusing on a specific physics problem. The end product was an app which visually represented the time-dependent Schrodinger's equation for a particle in a square potential well, which was simulated using quantum computing.

2 Theoretical Introduction: Quantum Computing

The basic unit of information in quantum computing is the qubit. It can be represented as a two-component unit vector containing the probability amplitudes for the states $|0\rangle$ and $|1\rangle$ respectively.

The states of qubits can be modified using gates, which are represented as square matrices. The new state-vector is calculated by multiplying the matrix that corresponds to a given gate and the vector representing the qubit's initial state.

The X-gate is a single-qubit gate, is represented by the Pauli X matrix:

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

It's analogous to a NOT gate in classical computing; it swaps the amplitudes of $|0\rangle$ and $|1\rangle$.

The Hadamard gate, or H-gate is also a single qubit gate, which is represented with:

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

This can be interpreted as a transformation from the Z basis to the X basis.

In multiple-qubit gates, there will usually be one qubit, called the control, which remains unchanged and determines whether an action will take place, and the target, which will be modified according to the state of the control.

The C-NOT gate is a two-qubit gate. If the control qubit's state is $|1\rangle$, it will apply an X gate to the target qubit. It can be represented with the following matrix:

$$\text{C-NOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

At the end of a circuit, a measurement made. The end result will be either a 0 or 1.

To determine the state right before the measurement was taken, the circuit must be repeated multiple times. The number of "shots" refers to the number of repetitions of the circuit.

I will be using Qiskit, IBM's python library for quantum computation. It can be used to either simulate a circuit, which means the end state vector will be calculated analytically and random numbers will be generated following that distribution to simulate the result of an ideal quantum computer, or to send a circuit to a real computer.

3 Preliminary approach

Before I focused on a larger problem I wanted to studied, I practiced with simpler circuits.

3.1 Superdense Coding

3.1.1 Theoretical Basis

The aim of this circuit is to send 2 bits of information with a single qubit.

To accomplish this, we must begin with two entangled qubits in a Bell state. A series of gates are then applied to the first qubit, after which it is sent with the second qubit. Once they are measured together, depending on what gates were applied, it should only be possible to obtain one of the four possible results.

To entangle the qubits, which are initially in the state $|00\rangle$, the first qubit must pass through an H gate, which will result in the state:

$$|00\rangle + |01\rangle$$

We then apply a C-NOT gate, with the first qubit as the control and the second as the target. The new state will be:

$$|00\rangle + |11\rangle$$

In qiskit, qubits are numbered from right to left, so in a state $|01\rangle$, 1 corresponds to the state of the first qubit, and 0 to the state of the second one.

To encode the state we want to send, we now apply a series of single-qubit gates to the first qubit.

To send "00", no additional gates need to be applied. To measure the qubits together, first we apply a C-NOT gate to get the state:

$$|00\rangle + |01\rangle$$

By applying an H-gate to the first qubit, we can see that the only state that can be measured is $|00\rangle$:

$$|00\rangle + |01\rangle + |00\rangle - |01\rangle = |00\rangle$$

To send "01", after the qubits are entangled, we must apply a Z-gate to the first qubit.

$$|00\rangle - |11\rangle$$

We once again take a measurement following the procedure described above:

$$|00\rangle - |01\rangle$$

$$|00\rangle + |01\rangle - |00\rangle + |01\rangle = |01\rangle$$

To send "01", we must apply an X-gate:

$$|01\rangle + |10\rangle$$

After the CNOT, the state is:

$$|11\rangle + |10\rangle$$

After the H gate, the final state is:

$$|10\rangle - |11\rangle + |10\rangle + |11\rangle = |10\rangle$$

The remaining combinations ("10" and "11") can be similarly sent by applying different single-qubit gates in the place of the X-gate.

3.1.2 Number of Shots

Given that the probability of measuring a given result should be 100% for all four circuits, this is good to study the uncertainty associated with modifying certain parameters.

As mentioned earlier, the number of shots refers to the number of times a circuit is repeated to get an estimation of the state vector before the measurements were taken.

There is no accepted number of shots that are necessary to obtain a good estimation of the result, although most studies seem to use between 2048 and 8192.

I want to explore how number of shots affects uncertainty for each superdense coding circuit using IBM-Oslo, which was the quantum computer with the shortest queue at the time.

Here, I'm considering the uncertainty to be the percentage of times the expected pair of bits isn't measured.

We would expect uncertainty to vary greatly for smaller number repetitions, and then stabilize after a certain point.

We can see in Figure 1 that the uncertainty does stabilize after around 2000 shots. However, the uncertainty after this point is still very large considering these are fairly small circuits.

It's interesting to note that in the "00" circuit, which is composed of 4 gates, the final uncertainty was the smallest, at around 2%. "01" and "10", which both contain 5 gates had the uncertainty stabilize at around 7%, and "11", which had 6 gates had the largest end uncertainty at around 10%.

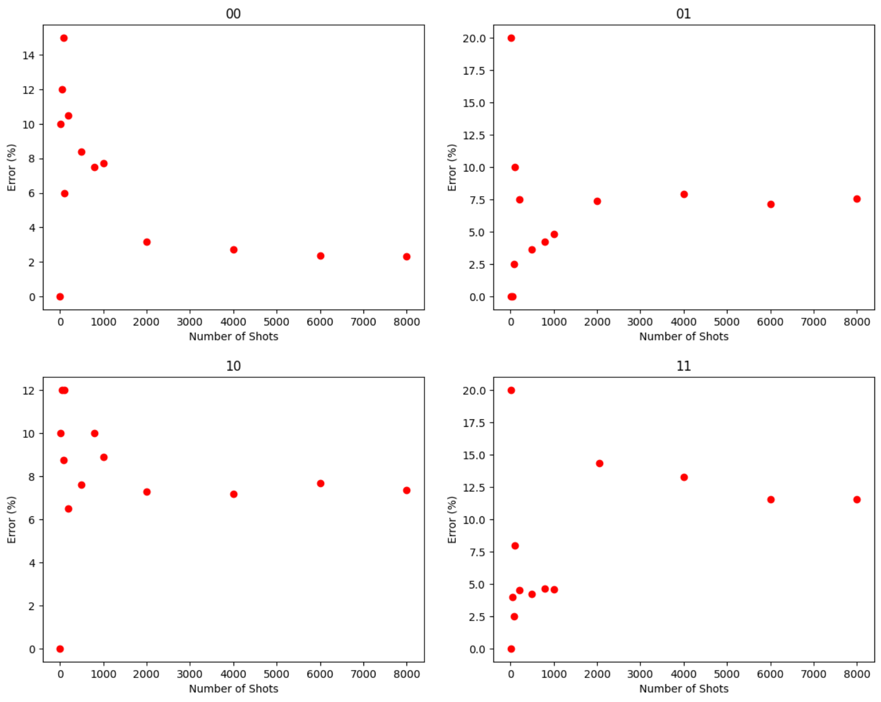

Figure 1: Uncertainty as a function of number of shots for each superdense coding circuit.

3.1.3 Connectivity

Not all qubits in a quantum computer are physically connected. When we try to send a circuit containing controlled gates between qubits that are unconnected, the transpiller, which translates the qiskit circuit into instructions for the quantum computer, will create equivalent circuits to connect them. Because of this, we would expect circuits that involve controlled gates between unconnected qubits to have higher uncertainties.

Figure 2 shows which qubits in IBM Oslo are connected and the strength of the connections. I compared the uncertainties of all superdense circuits using qubits 0 and 1 of IBM Oslo, which are connected, and 0 and 4, which are unconnected. I repeated the circuits 8192 times.

The results in Table 1 show that, as expected, the precision for the circuits with connected qubits is slightly higher than for unconnected qubits, although the results are fairly close and the uncertainty even for connected qubits can be fairly high. This suggests that for the small circuits that I will be using, it's unnecessary to optimize the circuits based on connectivity.

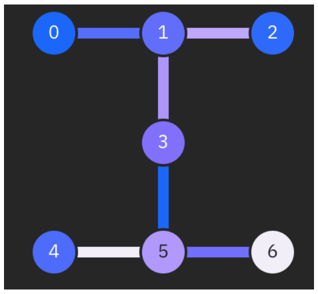

Figure 2: Topology of IBM Oslo.

8192 shots	11	10	01	00
connectivity	0.884	0.901	0.891	0.989
no connectivity	0.839	0.863	0.868	0.987

Table 1: Proportion of times where the correct 2-bit pair was measured in each superdense coding circuit, for connected and unconnected qubits.

3.2 Fast Fourier Transform

The Fourier transform decomposes a function of time or space into the frequencies that make it up. A discrete Fourier transform will similarly map a vector onto another vector according to:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega_j^{jk}$$

Where y_k and x_k are the components of the output and input vectors respectively, N is the number of elements in the vector and $\omega_j^{jk} = e^{2\pi i \frac{jk}{N}}$. The Quantum Fourier Transform (QFT) is a discrete Fourier transform of state vectors, containing probability amplitudes.

It can be shown that the QFT of a given state is equivalent to ([1]):

$$|y\rangle = \frac{1}{\sqrt{N}}(|0\rangle + e^{2\pi ix/2}|1\rangle) \otimes (|0\rangle + e^{2\pi ix/2^2}|1\rangle) \otimes ... \otimes (|0\rangle + e^{2\pi ix/2^n}|1\rangle) \tag{1}$$

Here, $N = 2^n$ is the number possible states and n is the number of qubits that make up the circuit.

The circuit that is equivalent to 1 is represented in Figure 3 [3]. H represents a Hadamard gate, and R_k are controlled rotations, which correspond to:

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}$$

To confirm this is correct, we apply an H-gate to the first qubit, to obtain:

$$\frac{1}{\sqrt{2}}(|0\rangle + \exp(\frac{2\pi i}{2}x_1)|1\rangle) \otimes |x_2x_3...x_n\rangle$$

If we now apply an R_k gate on the first qubit with the second one as the control, the state becomes:

$$\frac{1}{\sqrt{2}}(|0\rangle + \exp(\frac{2\pi i}{2^2}x_2 + \frac{2\pi i}{22}x_1)|1\rangle) \otimes |x_2x_3...x_n\rangle$$

By applying R_k gates on the first qubit with all the remaining qubits as the control, the state will become:

$$\frac{1}{\sqrt{2}}(|0\rangle + \exp(\frac{2\pi i}{2^n}x_n + ... + \frac{2\pi i}{22}x_1)|1\rangle) \otimes |x_2x_3...x_n\rangle$$

It can be seen that if this procedure is repeated with all subsequent qubits, the end state will be 1.

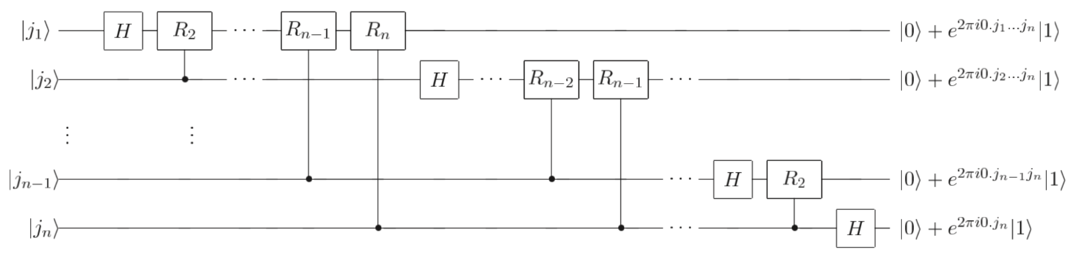
$|j_1\rangle$
$|j_2\rangle$
$|j_{n-1}\rangle$
$|j_n\rangle$
H
R_2
R_{n-1}
R_n
R_{n-2}
$|0\rangle + e^{2\pi i 0.j_1 \dots j_n}|1\rangle$
$|0\rangle + e^{2\pi i 0.j_2 \dots j_n}|1\rangle$
$|0\rangle + e^{2\pi i 0.j_{n-1} j_n}|1\rangle$
$|0\rangle + e^{2\pi i 0.j_n}|1\rangle$

Figure 3: QFT circuit [3].

3.3 Number of Gates

Quantum computers have property known as "coherence time", which indicates how long a qubit can last before losing information. Circuits with increased number of gates are more likely to surpass the coherence time and not produce usable results. [1]

The number of gates, G, of a QFT circuit increases with the number of qubits, n like:

$$G = \frac{1}{2}n(n+1)$$

If a qubit is initially in superposition, with equal probabilities for each state, we would expect the transformed state to be $|0\rangle$, independently of the number of qubits. I consider the uncertainty to be the proportion of times states other than 0 were measured.

As seen in Figure 4, the uncertainty grows linearly with the number of qubits very rapidly, to the point where it exceeds 20% after only 3 qubits. This suggests that circuits involving QFT of a significant number of qubits could currently only be done through simulations.

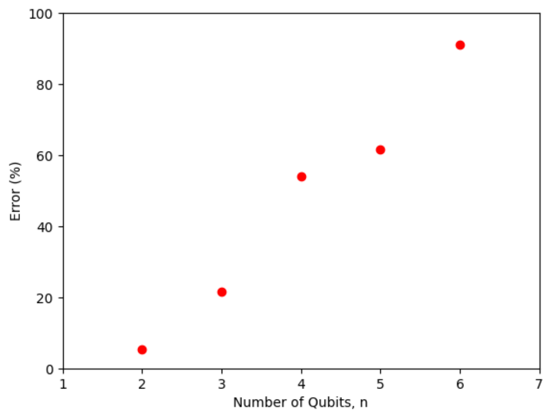

Figure 4: Uncertainty associated with a QFT of a state initially in superposition, as a function of the number of qubits in the circuit.

4 Simulating Schrodinger equation

4.1 Theoretical Introduction

The time dependent Schrodinger equation, which describes the evolution of a quantum system, is:

$$i\hbar\frac{\partial\Psi}{\partial t} = \hat{H}\Psi$$

We will consider $\hbar = 1$ and $m = 0.5$ to simplify the problem. The Hamiltonian is composed of the kinetic and potential energy operators $\hat{H} = \hat{K} + \hat{V}$, where $\hat{K} = p^2/m$.

The time evolution of the wave function is given by:

$$|\Psi(x, t + \Delta t)\rangle = e^{-i(\hat{K}+\hat{V})\Delta t} |\Psi(x, t)\rangle$$

$\hat{K}$ and $\hat{V}$ don't generally commute, but can be split using a first order approximation to obtain [4]:

$$e^{i\hat{H}\Delta t} \approx e^{i\hat{V}\Delta t} e^{i\hat{K}\Delta t}$$

We can apply step-well and double-well potentials to our quantum circuit with the use of a Z-rotation operator:

$$\mathrm{RZ}(\lambda) = \begin{pmatrix} e^{-i\frac{\lambda}{2}} & 0 \\ 0 & e^{i\frac{\lambda}{2}} \end{pmatrix}$$

The potential operator that corresponds to a step-potential can be rewritten as:

$$e^{i\hat{V}\Delta t} = e^{iv\sigma_z \Delta t}$$

Here, σ_z is the Pauli matrix and v corresponds to the height of the potential. This is equivalent to $RZ(\lambda = 2v\Delta t)$.

We can see that the σ_z operator causes 0 qubits to have a potential of 0, and 1 qubits to have a potential of $-v$. By applying the gate on the lowest order qubit creates a step-well potential, applying it to the second qubit creates a double-well potential, and applying it to both creates a single-well.

The kinetic operator is diagonal in the momentum space, so we can perform a Fourier transform, which allows there to be a correspondence between the operator and the quantum gates.

According to [2], the kinetic operator can be implemented with the following gates:

$$e^{-i\hat{K}\Delta t} = (\mathrm{QFT})\Phi_{01} Z_1 Z_0 (\mathrm{QFT})^{-1}$$

Z_i represents an $RZ(\lambda = 2\gamma c_i \Delta t)$ applied to qubit i. Φ_{01} is a controlled $RZ(\lambda = 2\gamma c_2 \Delta t)$ gate, where the first qubit is the control and the second one is the target. γ and c_i are constants which are specified in [2].

In summary, the Schrodinger equation for a time step Δt can be simulated with the circuit shown in Figure 5.

This means that it's not feasible to do this with a real quantum computer, since the number of gates required to obtain a time evolution would surpass those permitted by the qubit's coherence time, as discussed in Section 3.3. This circuit can only be implemented with simulations on classical computers.

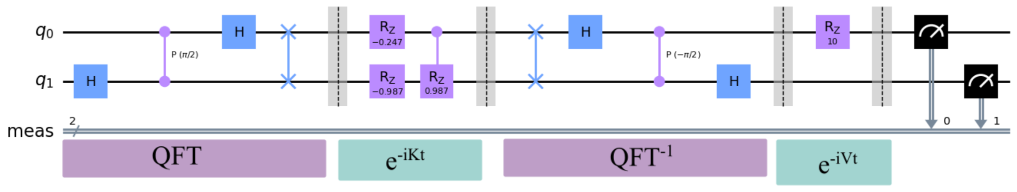

Figure 5: Circuit for simulating the Schrodinger equation with a double well potential for one time step.

4.2 Kivy Interface

To create a visual representation of my simulation of the Schrodinger equation, I used Kivy, a Python framework for developing apps.

Figure 6 shows the layout of my app. You can first set the initial state by adjusting the four sliders, which represent the probability of finding the particle at each of the four positions. The exact value of each probability is shown above the sliders.

The box in the upper right allows you to set the type and height of the potential barrier.

Once the initial conditions have been set, you must press the "Normalize" button in the box in the lower right, which normalizes amplitudes of the initial state. You can then either press "Set initial state" to change the initial probabilities again, or the "Calculate" button, which calculates the probabilities for each position for 50 time intervals, with $\Delta t = 0.1$. This may take a few seconds. After the calculation has been completed, you may press the "Play button". This will move the sliders for each position to show the shifting probabilities over time. Simultaneously, a smaller line-graph showing how the probabilities for the four positions have evolved as a function of time. After, that graph will be displayed in the place of the sliders. At this point, you can press "Play" to replay what was previously calculated, or "Set initial state" to set new initial conditions.

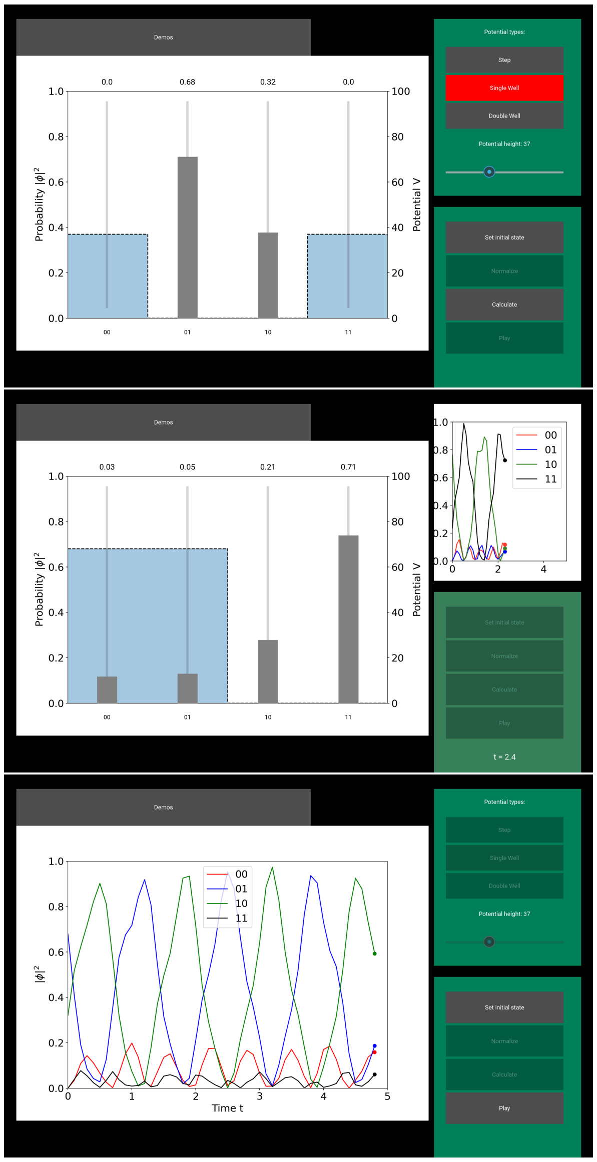

Figure 6: Interface of my app while setting the initial state, during the time the time the evolution of the states is shown, and after the animation has stopped.

I also included a series of pre-calculated demos, which can be selected through a drop-down menu located at the top left corner.

The first demo has a double well potential, with $v = 50$, and the particle is initially in 01, in the first well, as can be seen in Figure 7. We can observe that the particle oscillates between 01, and 11, with the probabilities for 00 and 10 staying constant at zero. This exhibits quantum tunneling behavior.

The second demo has a single well potential, with the particle initially at 10 (Figure 8). We can

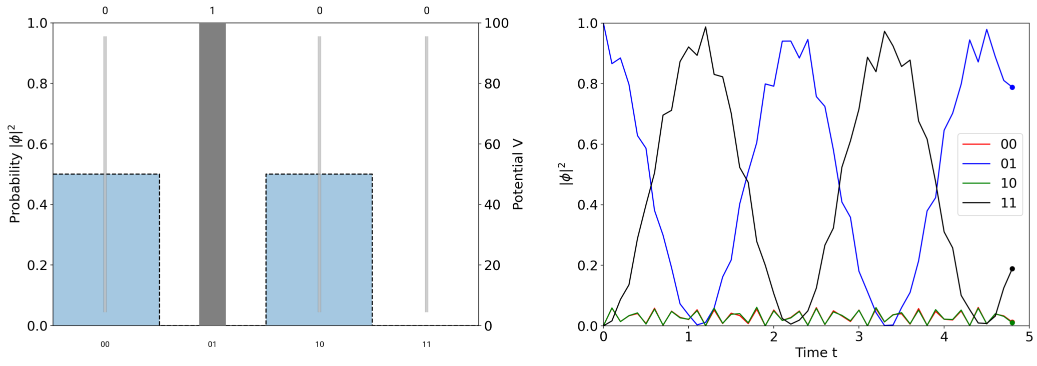

Figure 7: Initial state for Demo 1, with a double-well potential of $v = 50$ (left), and the time evolution of the probabilities of the four positions(right).

observe that the particle now oscillates between 01 and 10, since the potential barrier doesn't allow the particle to access the other two positions.

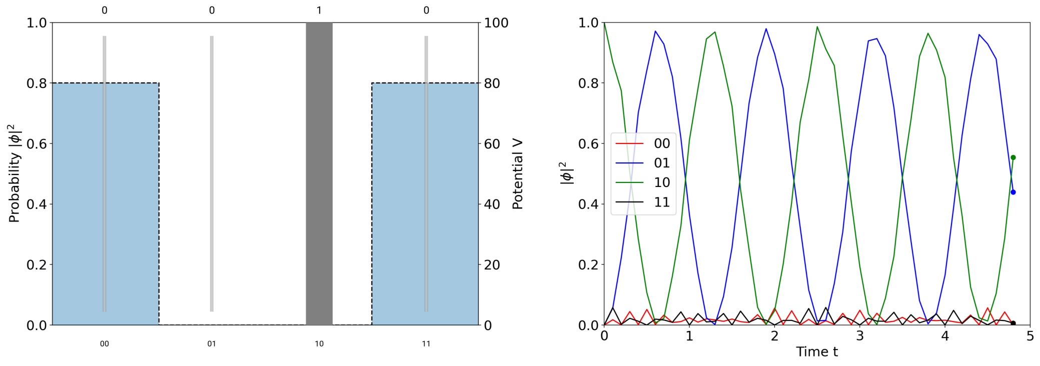

Figure 8: Initial state for Demo 2, with a single-well potential of $v = 80$ (left), and the time evolution of the probabilities of the four positions (right).

The third demo has a particle initially at 11, under a step-well potential of $v = 50$ (Figure9). Similarly to the previous case, the particle oscillates between 10 and 11, which are the positions where the potential is low.

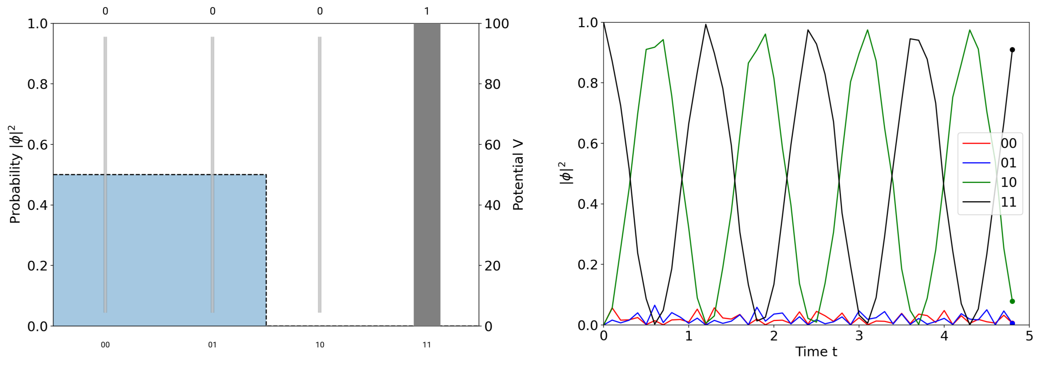

Figure 9: Initial state for Demo 3, with a step-well potential of $v = 50$ (left), and the time evolution of the probabilities of the four positions (right).

4.3 Discussion

Quantum computing shows a lot of potential for improving computational efficiency, especially in the context of physical simulations. My simulation for the Schrodinger equation were fairly similar to what is obtained by solving the equation analytically, and it reproduced quantum tunneling behavior.

The main drawback to QC is that the technology isn't there yet. The error associated with even simple circuits is significant, and while simulating a quantum computer on a classical computer, this process is more computationally costly than solving the problem through other numerical means.

The step of translating Schrodinger's equation into quantum gates adds a layer of mathematical abstraction that makes hard to tell what is happening at a physical level. For instance, the energy of the particle isn't introduced anywhere, and it's unclear whether this means the energy is zero, or it has been introduced implicitly somewhere.

While the RZ-gate introduces regions of high and low potential it's also not completely clear where the origin for the potential energy is, which could be relevant if I were trying to solve the same problem using a conventional numerical approach, for instance finite differences.

The simulation also shows some odd behaviors with regards to changing the height of the potential barrier. One would expect the amplitude of the state function at the other side of the well to decrease with the height of the well, until it reached a point at which tunneling is no longer possible. There is no value for potential where tunneling doesn't take place in my simulation. The height only seems to affect the period of oscillation between the 01 and 11 positions, which decreases as the height of the well increases until a certain value of potential.

As a whole this has been a great learning opportunity; it helped me improve my coding skills significantly and deepened my understanding of quantum mechanics.

References

[1] Amira Abbas, Stina Andersson, Abraham Asfaw, Antonio Corcoles, Luciano Bello, Yael Ben-Haim, Mehdi Bozzo-Rey, Sergey Bravyi, Nicholas Bronn, Lauren Capelluto, Almudena Carrera Vazquez, Jack Ceroni, Richard Chen, Albert Frisch, Jay Gambetta, Shelly Garion, Leron Gil, Salvador De La Puente Gonzalez, Francis Harkins, Takashi Imamichi, Pavan Jayasinha, Hwajung Kang, Amir h. Karamlou, Robert Loredo, David McKay, Alberto Maldonado, Antonio Macaluso, Antonio Mezzacapo, Zlatko Minev, Ramis Movassagh, Giacomo Nannicini, Paul Nation, Anna Phan, Marco Pistoia, Arthur Rattew, Joachim Schaefer, Javad Shabani, John Smolin, John Stenger, Kristan Temme, Madeleine Tod, Ellinor Wanzambi, Stephen Wood, and James Wootton. Learn quantum computation using qiskit, 2020.

[2] Narendra N. Hegade, Nachiket L. Kortikar, Bikramaditya Das, Bikash K. Behera, and Prasanta K. Panigrahi. Experimental demonstration of quantum tunneling in ibm quantum computer, 2017.

[3] Michael A Nielsen and Isaac L Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2010.

[4] Andrew T. Sornborger. Quantum simulation of tunneling in small systems. *Scientific Reports*, 2(597), 2012.