Quantum Tunneling: From Theory to Error-Mitigated Quantum Simulation

Sorana Catrina¹ Alexandra Băicoianu²

¹Faculty of Mathematics and Computer Science, Transilvania University of Braşov, Eroilor, 29, Braşov, 500036, Romania

²Department of Mathematics and Computer Science, Transilvania University of Braşov, Eroilor, 29, Braşov, 500036, Romania, email a.baicoianu@unitbv.ro

1 Abstract

Ever since the discussions about a possible quantum computer arised, quantum simulations have been at the forefront of possible utilities and the task of quantum simulations is one that promises quantum advantage. In recent years, simulations of large molecules through VQE or dynamics of many-body spin Hamiltonians may be possible, and even able to achieve useful results with the use of error mitigation techniques. Simulating smaller models is also important, and currently, in the NISQ (Noisy intermediate-scale quantum) era, it is easier and less prone to errors. This current study encompasses the theoretical background and the hardware aware circuit implementation of a quantum tunneling simulation. Specifically, this study presents the theoretical background needed for such implementation and highlights the main steps of development. Building on classic approaches of quantum tunneling simulations, this study improves the result of such simulations by employing error mitigation techniques (ZNE and REM) and uses them in conjunction with multiprogramming of the quantum chip for solving the hardware under-utilization problem that arises in such contexts. Moreover, we highlight the need for hardware-aware circuit implementations and discuss these considerations in detail to give an end-to-end workflow overview of quantum simulations.

Keywords: quantum simulation, error mitigation, multiprogramming, compiler/transpiler

2 Introduction

Simulations of natural phenomenon have always been of great interest to scientists, and ever since Feynman proposed that quantum mechanics can not be efficiently simulated on a classical computer, quantum computers have been employed for such tasks [10, 28].

Current simulations on quantum computers are, however, prone to errors. Being in the NISQ era (Near Intermediate-Scale Quantum), there are different options to deal with these impediments [36]. Namely, one can try to reduce the noise in the system through hardware improvements and calibrations [27] or design algorithms that leverage the structure of noise in order to extract the useful information represented usually by the estimated value at zero noise level. This can be done either through classical post-processing techniques such as ZNE [13, 41] or REM [2, 43], or even by employing machine learning algorithms [26].

Regarding the fields that have the possibility to contribute greatly from quantum supremacy, quantum simulations for quantum chemistry is one such example [5, 6]. The study of different materials through quantum simulations can also prove to be an important field that would benefit from quantum advantage [44]. Implementations of such tasks that take into consideration current limitations of quantum computers already exist, and range from quantum simulations of small molecules through variational quantum eigensolvers to modelling of open systems [14, 22, 24, 31].

The current article focuses on an example of a quantum phenomenon known as quantum tunneling. It aims at simulating this on a quantum computer, the circuit of which serves as an effective tool for pushing the boundaries of existing hardware, given the potential to increase circuit depth through simulating over longer periods of time. Quantum simulations of tunneling have predominantly concentrated

on a single approach to executing the given task [1, 17, 39]. With a focus on hardware run considerations for superconducting architectures, the study at hand aims to clarify various circuit implementation alternatives. We believe it is crucial to emphasize the compiler's significance in relation to the utilized abstract circuit for this task. Moreover, error mitigation techniques that proved efficient are discussed. Multiprogramming, a technique aimed at addressing the hardware under-utilization problem, is also employed along with the EM techniques chosen to complete the proposed end-to-end workflow. Results of these experiments are also presented, showing an expectation value off only by 0.006 for the transmission probability of the particle through the potential barrier after error mitigation.

3 Theoretical Background

To simulate Quantum Tunneling effectively, one must consider the theoretical implications inherent in such an undertaking. Therefore, this section will focus on the physics background needed for understanding this effect, namely, the Schrödinger equation. The evolution of the wavefunction of a quantum mechanical system is described by the Schrödinger equation:

$$i\hbar \frac{\partial}{\partial t} \psi(x,t) = \left(-\frac{\hbar}{2m} \vec{\nabla}^2 + V(x,t)\right) \psi(x,t) \tag{1}$$

where

 \hbar = reduced Planck constant

 $\psi(x,t)$ = the wavefunction representing the system

 $\vec{\nabla}^2$ = the Laplace operator

V(x,t) = the potential that represents the environment at time t

It is evident that this constitutes a linear partial differential equation. The wavefunction, $\psi(x,t)$, assigns a complex number to each point in space, x, at each point in time, t. The potential energy term (in the equation denoted with V corresponding to the environment in which the particle exists) and the kinetic energy operator form the Hamiltonian operator, which corresponds to the quantum system's total energy. For a one-dimensional system, the Laplace operator takes the following form:

$$\vec{\nabla}^2 = \frac{\partial}{\partial x^2}$$

We should note that based on the potential energy, the eigenspectrum of the wave function takes different forms. In this study, we will consider a square-well potential, which we will present in Section 3.1

The mathematical formulation of quantum mechanics states that the wavefunction, previously denoted as $\psi(x,t)$ is a statevector, $|\psi\rangle$, belonging to a Hilbert Space, \mathcal{H} . More information on the formalism of quantum mechanics can be found in Griffith's Introduction to Quantum Mechanics [15]. In Dirac notation, the time-dependent Schrödinger equation takes the following form:

$$i\hbar \frac{\partial}{\partial t} |\psi(x,t)\rangle = \hat{H} |\psi(x,t)\rangle$$
 (2)

$$\hat{H} = \hat{V} + \hat{K} \tag{3}$$

where

 $|\psi(x,t)\rangle$ = the statevector of the quantum system

 \hat{H} = the Hamiltonian operator \hat{V} = the potential operator \hat{K} = the kinetic operator

Here, \hat{H} represents the Hamiltonian operator and as specified earlier, this is the sum between the potential energy operator and the kinetic energy operator. \hat{H} is a hermitian operator whose eigenvalues represent the system's energy levels. Following the separation of variables, the time-independent Schrödinger equation along with the time evolution of an initial state $|\psi(0)\rangle$ (which can be described by the time evolution operator, \hat{U}) are presented bellow. We can see that Equation 4 is an eigenvalue equation, therefore wavefunction is an eigenfunction of the Hamiltonian operator with corresponding eigenvalues E.

$$\hat{H} |\psi(x,t)\rangle = E |\psi(x,t)\rangle \tag{4}$$

$$|\psi(x,t)\rangle = \hat{U}|\psi(x,0)\rangle = e^{-i\hat{H}t}|\psi(x,0)\rangle \tag{5}$$

where

 $|\psi(x,t)\rangle$ = the statevector of the quantum system

 \hat{H} = the Hamiltonian operator E = the energy of the system \hat{U} = the time evolution operator

3.1 Quantum Tunneling

Quantum tunneling is a phenomenon that is unique to quantum mechanics, it defies explanation through classical mechanics models, and it shaped our understanding of the world around us. Its significance extends across various fields, such as Scanning Tunneling Microscopy [42] and Josephson Junctions, which are pivotal in the development of superconducting quantum computers [19]. Another example is quantum dots that also harness the principles of quantum tunneling and are used in other quantum computer architectures, such as spin qubit quantum computers, first proposed in [29]. This underscores the critical necessity to delve into the study of this effect, given its far-reaching implications across numerous fields.

The effect of Tunneling happens between two wells separated by a potential barrier. That is, the potential takes the form presented in Equation 6, where a is the size of the potential barrier. Solving the Schrödinger equation for the given potential and a kinetic energy gives rise to a wavefunction that tunnels between the wells. This phenomenon will be visible in the simulations presented starting from Section 4.4.

$$V(x) = \begin{cases} 0, x < 0 \\ V_0, 0 \le x \le a \end{cases} \quad a = \text{size of potential barrier}$$

$$0, x > 0$$

$$(6)$$

4 Implementation considerations

To successfully simulate quantum tunneling in a quantum computing environment, we need to implement the Hamiltonian operator and model the initial wavefunction $|\psi(x,0)\rangle$. This is crucial for executing both Equations 4 and 5. To accomplish this, we must discretize both space and time. This is necessary because quantum computers operate with a fixed number of qubits, necessitating the discretization of infinite space for accurate simulation.

Furthermore, as Equation 5 suggests, we must simulate the system across various time intervals. This entails implementing the operator $\hat{H}t$ for each time step and sequentially applying it to the initial wavefunction.

In the following section, we will explore the implementation of the aforementioned equations on a quantum computer and examine various approaches as necessary. In Section 4.1, we delve into the discretization process and explore the requisite adaptations essential for correctly and efficiently simulating quantum tunneling on a quantum computer, such as the Trotter-Suzuki approximation [8, 16].

4.1 Discretizing time and space

Discretizing time

Considering the time evolution of the system, in order to simulate how the system evolves, we need to implement the time evolution operator. Equation 5 can be further expanded to highlight the use of the kinetic and potential operators and how the wavefunction changes after Δt time as in Equation 7.

$$|\psi(x,t+\Delta t)\rangle = e^{-i\hat{H}\Delta t}|\psi(x,t)\rangle = e^{-i(\hat{K}+\hat{V})\Delta t}|\psi(x,t)\rangle$$
(7)

where

 $\Delta t =$ the amount of time after wavefunction is left to evolve after time t

It is important to notice that if we implement the operator $e^{(\hat{K}+\hat{V})\Delta t}$ for a small enough Δt we would be able to simulate the time evolution of the entire system by sequentially applying this operator. However, an issue arises when attempting to utilize Equation 5 by decomposing into terms containing the kinetic and potential energy operators, which is highlighted in Equation 8.

$$e^{-i\hat{H}\Delta t} = e^{-i(\hat{K}+\hat{V})\Delta t} \neq e^{-i\hat{K}\Delta t}e^{-i\hat{V}\Delta t}$$
(8)

This is because, in general, potential and kinetic energy operators do not commute. It is to remember that the potential operator is often dependent on \hat{x} and that $\hat{K} = \frac{\hat{p}^2}{2m}$; taking into consideration that $[\hat{x}, \hat{p}] = i\hbar$ shows that position and momentum operators do not commute. Therefore, we will need to use *Suzuki-Trotter approximation*. We will mostly use the first order formula given by Equation 9, but higher order approximations can also be used [8]. For example, the second order Suzuki-Trotter approximation is given by Equation 10 [16].

$$e^{-i\hat{H}\Delta t} \approx e^{-i\hat{K}\Delta t}e^{-i\hat{V}\Delta t} + O((\Delta t)^2)$$
(9)

$$e^{-i\hat{H}\Delta t} \approx e^{-i\hat{V}\frac{\Delta t}{2}} e^{-i\hat{K}\Delta t} e^{-i\hat{V}\frac{\Delta t}{2}} + O((\Delta t)^3)$$
(10)

Discretizing space

Since the Schrödinger equation is given in continuous space, we encounter a challenge in implementation due to the finite number of qubits available. We can therefore discretize the interval $0 \le x \le L$ with interval spacing Δl within the boundary region with a periodic boundary condition $|\psi(x+L,t)\rangle = |\psi(x,t)\rangle$. The wave can therefore be mapped to a n-qubit register as in Equation 11 where k represents the particle location corresponding to binary number k.

$$|\psi(x+L,t)\rangle = \sum_{k=0}^{2^{n}-1} \psi(x_k,t)|k\rangle \tag{11}$$

4.2 Implementing the operators

In accordance with the Suzuki-Trotter approximations, to simulate quantum tunneling, the following operators need to be implemented:

- 1. The Kinetic Energy Operator, $e^{\hat{K}}$
- 2. The Potential Energy Operator, $e^{\hat{V}}$

4.2.1 The Kinetic Energy Operator

The kinetic energy operator can be represented with the help of the momentum operator as shown in Equation 12, where \hat{p} is the momentum operator.

$$\hat{K} = \frac{\hat{p}^2}{2m}.\tag{12}$$

where

 \hat{p} = the momentum operator

m =the mass of the particle

Different from coordinate space, in momentum space \hat{p} is diagonal. The potential operator, however, is dependent on \hat{x} , which is diagonal in coordinate space. Consequently, it is advantageous to switch from coordinate space to momentum space to apply the kinetic energy operator and come back to coordinate space to apply the potential operator (this is because the construction of diagonal operators is easier). In quantum mechanics, switching from coordinate space to momentum space is done with the help of the Fourier Transform, as shown in Equation 13. The inverse transformation is done through Equation 14.

$$\tilde{\psi}(p,t) = \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} dx \psi(x,t) * e^{-\frac{-ipx}{\hbar}}$$
(13)

$$\psi(x,t) = \frac{1}{2\pi\hbar} \int_{-\infty}^{\infty} dp \tilde{\psi}(p,t) * e^{-\frac{-ipx}{\hbar}}$$
(14)

Considering the same limitations, we will need to use the discrete formula of the Fourier Transform, and moreover, we will use the Quantum Fourier Transform (QFT), shown below. We will use the inverse Quantum Fourier Transform (IQFT) to convert back to coordinate space.

$$|j\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} |k\rangle$$
$$\omega_N^{jk} = e^{2\pi i \frac{jk}{N}}$$

As mentioned earlier, in momentum space, the kinetic operator is diagonal. Using the notation in Equation 15, the operator $e^{-i\hat{k}^2\Delta t}$ can be represented with the use of the phase gate, as shown in Equation 16 with the Pauli Z gate [1, 38].

$$|k\rangle = \sum_{i=0}^{n} 2^{i} j_{i+1}$$
 (15)

$$e^{-i\hat{k}^2 \Delta t} = exp(\frac{i\phi}{2^{2n-3}} (1 + \sum_{j=1}^n 2^{j-1} \hat{Z}_j)^2)$$
(16)

With this formulation, we can easily see the one qubit operations. However, in order to better see the two-qubit operations, we have to open the sum. Focusing on the two-qubit terms, we get:

$$S = \sum_{i} \sum_{j} 2^{\frac{2^{j-1}2^{i-1}}{2^{2n-3}}} \hat{Z}_{i} \hat{Z}_{j} = \sum_{i} \sum_{j} \frac{2^{i+j-1}}{2^{2n-3}} \hat{Z}_{i} \hat{Z}_{j} = \sum_{i} \sum_{j} \frac{1}{2^{2n-i-j-2}} \hat{Z}_{i} \hat{Z}_{j}$$

Since in this notation, i and j start for 1 and when implementing indexing starts from 0, we need to change the start and ending point and rewrite the sum as follows:

$$i \to i' + 1$$

$$j \to j' + 1$$

$$S = \sum_{i'} \sum_{j'} \frac{1}{2^{2n' - (i'+1) - (j'+1) - 2}} \hat{Z}_i \hat{Z}_j$$

$$= \sum_{i'} \sum_{j'} \frac{1}{2^{2n' - i' - j' - 4}} \hat{Z}_i \hat{Z}_j$$
(17)

Algorithm 1 Kinetic Energy Operator Implementation

Input Δt representing the time step size needed by trotterization, n representing the number of qubits, circuit the quantum circuit with two registers q, and aux as auxiliary register Output The updated circuit

```
\begin{array}{c} \mathbf{for} \ i \in range(n) \ \mathbf{do} \\ circuit.p(\frac{\phi}{2^{2n-3}}, q[i]) \end{array}
                                                                             ▶ The single qubit rotations from Equation 16
 3: end for
 4: for i \in range(n) do
         for j \in range(i+1,n) do
 5:
              apply cx on q[i], aux
 6:
              apply cx on q[j], aux
 7:
             apply phase gate with angle \frac{1}{2^{2n-i-j-4}} on aux
                                                                                                        ▶ The two-qubit rotations
 8:
              apply\ cx\ on\ q[j],\ aux
 9:
              apply cx on q[i], aux
10:
11:
         end for
12: end for
```

The implementation can be accomplished in various ways. One approach is to follow the schema using auxiliary qubits [33], but one can also manage without such auxiliary qubits in simulations of 2 and 3 qubits [17, 39]. Using an auxiliary qubit is done in order to keep count of the parity of the qubits (the phase shift applied to the system is $+i\Delta t$ if the parity of the n qubits in the computational basis is

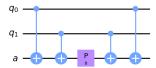


Figure 1: $Z_1 \otimes Z_2$ Hamiltonian implementation using ancilla qubits

even). This is highlighted in Algorithm 1 and illustrated for an example hamiltonian in Figure 1. The first section of the algorithm is for the single-qubit rotations, while the second is for the two-qubit terms in the above mentioned equations. We mention that lines 9-10 are meant to *undo* the parity counting in order to account for reusability.

However, we have to keep in mind that using an ancilla qubit will increase the hardware mapping depth for superconducting current architectures since the connectivity is limited. More information on implementations considering hardware constraints will be presented in Section 5.

4.2.2 Potential Energy Operator

Implementation of the potential energy operator is straightforward in coordinate representations: it is done by applying Z rotations on the qubit (or CZ operations on different qubits), depending on what type of potential we want. It is important to point out that we will follow qiskit ordering, that is, little endian. For example, we can apply the controlled Pauli Z operator on the highest-order qubit to implement multiple wells, as shown in the equation below. Similarly, to have only one well, we can apply it to the lowest-order qubit.

$$e^{i\hat{V}\Delta t} = e^{-iv\sigma_z^{n-1}\Delta t} = I \otimes I \otimes I \otimes \ldots \otimes \sigma_z$$

here, σ_z^{n-1} is the Pauli Z operator on the n-1 qubit.

4.2.3 Quantum Fourier Transform

We will need to implement the Quantum Fourier Transform to allow switching between coordinate space and momentum space representations. The Quantum Fourier Transform maps a quantum state $|X\rangle$ onto another quantum state $|Y\rangle$ by the following formulas:

$$|X\rangle = \sum_{j=0}^{n-1} x_j |j\rangle$$

$$|Y\rangle = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} y_j |j\rangle$$

$$y_k = \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} e^{2\pi i \frac{jk}{n}} x_j$$
(18)

In order to implement the QFT, we can represent it as follows [33]:

$$QFT |x\rangle = \frac{1}{\sqrt{N}} e^{2\pi i \frac{jk}{n}} x_j$$

$$= \frac{1}{\sqrt{N}} (|0\rangle + e^{\frac{2\pi i}{2}x} |1\rangle) \otimes (|0\rangle + e^{\frac{2\pi i}{2^2}x} |1\rangle) \otimes \dots \otimes (|0\rangle + e^{\frac{2\pi i}{2^n}x} |1\rangle)$$
(19)

4.2.4 Initial State

For simulations where the potential is $e^{i\hat{V}\Delta t} = e^{-iv\sigma_z^0\Delta t}$ (meaning we have a well on every other site), the initial state has $|1\rangle$ where we want our state to start. For every other type of potential, a gaussian wave is used as the main form:

$$|\psi\rangle = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

However, for moving waves, a momentum component is also added:

$$|\psi\rangle = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} e^{-ipx}$$

This works well for simulations with barriers since the wave is confined. A Gaussian wave is also used for free particle simulations. In QM, due to the need for normalization, the *wavepacket* is represented as a superposition of such plane waves.

$$\psi(x,t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dk \tilde{\varphi}(k) e^{-\omega(k)t} e^{ikx} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dk \tilde{\varphi}(k) e^{ikx}$$

4.3 Circuit Overview

Having discussed implementations of the kinetic and potential operator in previous sections as well as the initial state considered, we can now provide an overview of the circuit implementation. We want to simulate on the quantum computer Equation 5.

Upon considering the operators' implementation, we can provide an overview of the essential steps necessary for the creation of the circuit, as depicted in Figure 2 and is as follows:

- Step 1: This is represented by state preparation. This entails preparing $|\psi(x,0)\rangle$, as discussed in Section 4.2.4. This state can be a Gaussian wave for systems where the space has been discretizing using more qubits. Otherwise, the wave is approximated to the state containing a flipped qubit on the position of the particle
 - After this preparation, one can start implementing the time evolution with the use of the Suzuki-Trotter Approximation. One time step is represented by steps 2-5, these implement the Kinetic and Potential energy operators, respectively.
- Step 2: The QFT is applied to the quantum circuit. As mentioned in Section 4.2.1, one will first switch to momentum space with the use of the QFT and then implement the kinetic operator.
- Step 3: The Kinetic Energy Operator is implemented. This can be done either with or without ancilla qubits.
- Step 4: Switching back to coordinate space is represented by the Inverse QFT.
- Step 5: Implementation of the Potential Energy Operator, as described in Section 4.2.2.
- Step 6: The last step is measuring the working register, which gives us the position of the particle.

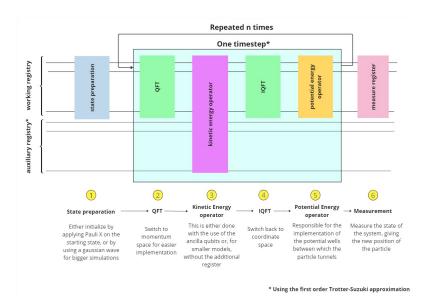
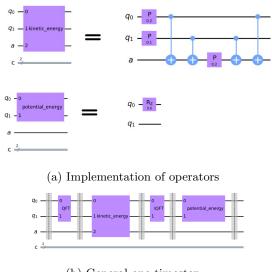


Figure 2: Full circuit overview



(b) General one timestep

Figure 3: High level overview of 2 qubit operations

For implementation, we used *qiskit*, but since it is compiled to QASM the code can be reused for other languages as well. The representation of operators for a 2 qubit example, as presented in the first section, is shown in Figure 3. A high-level view of the circuit for a 2 qubit implementation, where an ancilla qubit is used for the kinetic energy implementation, is present in Figure 3b following the scheme presented in Section 4.2.1 with ancilla qubits [33].

4.4 Four qubits Quantum Tunneling Simulation

This section presents results for 4 qubit quantum tunneling simulation in a noiseless environment. By analyzing these results, we can better understand the quantum tunneling phenomena and highlight how the parts discussed earlier come together to form a quantum tunneling simulation. Since this is a noise-free environment, the simulator employed does not account for possible errors, which are inevitable with today's quantum computers. Leveraging qiskit, this simulator is accessible via Aer, offering users the choice between noiseless and noisy simulation modes.

Regarding implementation, our approach incorporated the kinetic energy using an auxiliary qubit. The potential and QFT are implemented as discussed in earlier sections.

Table 1: Experiments Setup for 4 qubit quantum tunneling simulation

Type of potential	Initial wave		Δt	Number of time steps	Potential type
Wall Potential	Gaussian wave centered at	$0100\rangle$	0.1	20	1xxx
Two wells	Gaussian wave centered at	$0100\rangle$	0.1	40	x11x
Multiple wells	$ 1000\rangle$		0.1	20	xxx1

The setup of the experiments is summarised in Table 1. Potential type refers to positions in our simulation that have a non-null potential attributed to them in the potential landscape. The x matches for either 0 or 1. For example, for the two well simulation which has a potential type of "x11x", we can see in Figure 5b that there is non zero potential (represented by the yellow box) at positions: 0110, 0111, 1110, 1111.

The wall and multiple wells potential are discussed in Section 4.2.2. Potential type x11x can be achieved through a filter-type gate, as suggested in [1]. This should be rather intuitive, since we need to apply a phase corresponding to this operator to the states that have a non-zero potential in our modelling. This can be achieved through controlled gates using the auxiliary qubit. It can be observed that the ancilla qubit is convenient in this type of simulation since it can be used for both the potential operator and the kinetic energy operator. The implementation of the x11x barrier is presented in Figure

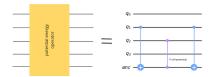


Figure 4: Implementation of x11x type potential

4. The results are presented in Figure 5, and for 5b and 5c tunneling is observed between the wells of potential. We also point out that in Figure 5b we can see that there is less tunneling between the wells in the first timesteps, since the potential barrier is also larger. Running the simulation with the circuit discussed for different timesteps, one manages to gather the results presented in Figure 5.

The first experiment, in which there is a potential wall (meaning no wells are created) reveals no tunneling happening, as expected. The results can be seen in Figure 5a. We highlight that the probability of the particle being at a certain position (represented on the x-axis) is visually illustrated through the intensity of color in the diagram. When one sets up multiple wells, as in Figure 5b and Figure 5c, tunneling can be observed. By this, we mean that the particle is able to "tunnel" from one well to the other, even though there is a potential barrier separating the two.

In Figure 5b, one can see a particle traveling to the right and after reaching the potential barrier, "tunneling" through the other well. The probability of transmission through the potential barrier is, of course, dependent on the potential. The transmission wave and the reflected wave are also observable. In Figure 5c, multiple wells have been implemented, and the simulation results are illustrated. The tunneling of the particle can be observed in this image as well. We mention that the starting position of the particle is represented by the statevector $|1000\rangle$.

In this section, we gathered a better understanding of what a noiseless quantum tunneling simulation reveals, studying different potentials and analyzing the results within the context of a 4 qubit simulation. Nonetheless, upon transitioning to actual hardware, numerous considerations arise that necessitate attention, thereby warranting adjustments before achieving satisfactory results. These considerations will be thoroughly discussed in the next section.

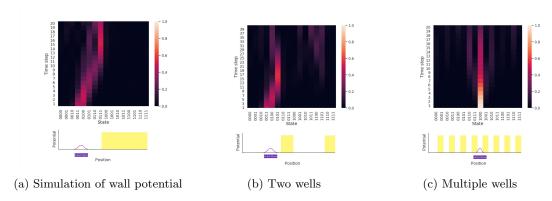


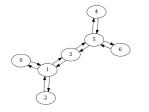
Figure 5: Each diagram reperesents a 4q noiseless simulation. The bottom image presents the profile of the potential, along with the initial position of the wave that can also be observed at timestep 0 of the diagram

5 Running on hardware

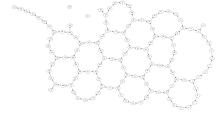
5.1 Introduction

When running on hardware, multiple factors need to be considered. We are still in the NISQ era, and therefore we need to make our circuit as prone to error as possible, this includes everything from modifying the circuit, to choosing the right layout and routing techniques.

The first thing one has to do is choose a backend. Part of the experiments were done on one of the three 7-qubit quantum computers part of the Falcon r5.11H chip and three 128-qubit quantum computers using the Eagle r3 chip. Simulations were run on the 7-qubit quantum computer Nairobi, Jakarta, and



(a) Coupling map for Nairobi



(b) Coupling map for Osaka qc

Figure 6: Coupling maps

the 128-qubit quantum computer Osaka. Apart from IBM's superconducting quantum computers, others could've also been considered, such as IQM's trapped ions quantum computers. Certain advantages and disadvantages could be analyzed for each architecture.

Once a quantum computer is chosen, we have access to the basis gates of such computer (for our case, ECR, ID, RZ, SX, X) and the chip layout (in the case of superconducting circuits).

The *layout* is important since the connectivity in superconducting quantum computers is not all-to-all. For example, when two *virtual qubits* which partake in different 2-qubit gates are mapped to *physical qubits* that are not connected, this might end up being less efficient since one of them will need to be swapped next to the other one (therefore adding 3 two-qubit gates to the circuit that introduce noise).

Also, since our circuit most of the time is not composed only of basis gates, but of composed gates as well, we need to transform our circuit to contain only allowed gates. This operation is done by the *transpiler*, which has a similar role to other languages' compiler. The transpiler works by applying multiple *passes* to the circuit, each doing different operations meant to optimize, layout, reduce depth and transform the circuit to run on the backend. However, we have to keep in mind that the transpiler does not always give the best option out there, we imply that two identical circuits, yielding the same statevector, may exhibit distinct implementations leading to variations in circuit depths.

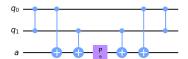
In Figure 7, in the abstract circuit section, we can observe two circuits that produce, in the end, the same statevector. However, we can observe that the implementation is different, and, therefore, depending on the backend chosen, we can see how the layout can influence circuit depth. For example, on ibm_auckland we can see that the transpiled circuit of the second implementation has a lower depth than that of the first one. Reducing 2-qubit gates plays an important role in ensuring the circuit is error-prone. However, on ibm_nairobi, the layout of the transpiled first circuit has a lower depth. This observation, that choosing the hardware-aware implementation of a certain problem is important when it comes to superconducting circuits must be considered when designing algorithms to be run on superconducting architectures. One can also see that the depth is increased by using several swap gates that were required to map the circuit to the layout. This process, responsible for adding swap gates to ensure connectivity adherence, constitutes the routing pass.

Apart from the layout, one has to consider the coupling map as well. The coupling map was symmetric for earlier chips, such as the Falcon r5.11H. This means that we could apply controlled 2-qubit gates regardless of which physical qubit was chosen as the control or target. However, on the newer chips (Eagle r3) the connectivity is not symmetric. Illustrations for coupling maps for the Falcon chip (more specifically, the Nairobi quantum computer) and the Eagle chip (the Osaka quantum computer) are presented in Figure 6. This means that, for example, the swap gate is implemented in a different way, not only because of the different basis gate (on the Eagle r3) but also due to the coupling map. We also remind that the ECR gate (part of the basis set on the Eagle r3 chip) is an echoed-cross resonance gate, equivalent to CX gate up to single qubit pre-rotations.

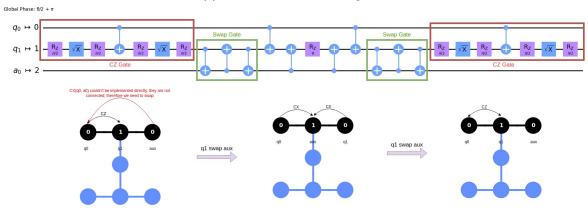
5.2 Transpiler

The transpiler is the main component that is used in order to run an abstract quantum circuit on a real quantum computer, as we've discussed earlier. This component is sometimes named compiler, depending on the programming language used. Therefore, we summarise the main components, as implemented by *qiskit*, the library we chose for implementation:

• Decomposing the custom gates into one and 2-qubit gates - one step here is unfolding; for our quantum simulation circuit, this means unfolding the kinetic energy operator and the potential energy operator



(a) The circuit with virtual qubits



(b) The transpiled circuit

Figure 7: Transpilation process

- Layout finding the best physical qubits on which our virtual qubits will be mapped to. Another tool we used for this pass was *mapomatic* for a more detailed look into the layout scoring process.
- Routing inserting swap gate in order to respect the connectivity
- Translation (into basis gates the native gates for the specific backend)
- Optimization removing redundancies in the circuit, applying circuit identities in order to decrease the circuit depth. For this pass, we also used the tket compiler.
- Scheduling this is an important pass for pulse-level control; but for this pass, we can also use different techniques (Dynamical Decoupling is a pass that needs scheduling afterward)

Qiskit's transpiler is not the only one that exists and is helpful. For example, apart from this transpiler we also used tket compiler that improved the circuit depth. This is due to different passes being implemented by tket (such as the FullPeepholeOptimise pass) that when applied to our circuit, proved efficient in making it shallower. As mentioned earlier, one should also consider that two circuits which in the end produce the same statevector, depending on the implementation and layout of the circuit, different final circuits with different depths would result. Figure 8 presents one example that highlights this idea.

5.3 Error Mitigation

Since we are in the NISQ era, rather then eliminating all the noise, the goal of different techniques is to rather mitigate them, therefore the field of quantum error mitigation was born. Multiple techniques have been evolved and are now used in order to extract useful information in real-world applications [9], such as the ones related to quantum chemistry [22, 24, 31]. Namely, some relevant techniques are zero noise extrapolation, probabilistic error cancellation, and error measurement mitigation [4]. Here, we will discuss Readout error mitigation and zero noise extrapolation. Dynamical Decoupling has also been tested, but the results were inconclusive as to whether they were more accurate.

5.3.1 Readout Error Mitigation

Readout Error Mitigation is an umbrella term used for multiple approaches to mitigating errors, usually with the help of a confusion matrix by applying its inverse to the outcome probability distribution to

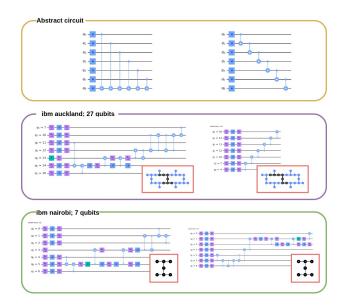


Figure 8: Illustration of why circuit design and backend influence the depth of the circuit. This calls for hardware-aware design

extract the noiseless distribution [2, 43]. This confusion matrix is a matrix that encodes the probabilities of measuring a prepared state $|u\rangle$ as another state, $|v\rangle$. One can consider whether the qubits' results are correlated between qubits or not. When correlated results are considered, for n qubits, the matrix will be $2^n \times 2^n$. If the noise is, however, considered, local, we will have n matrices 2×2 , one for each qubit.

Ideally, regarding the type of error that is considered, this matrices will have one on the main diagonal, and 0 everywhere else, meaning that, the state prepared is always the same as the state measured. However, in a noisy environment this is not the case, therefore this matrix will encode the effects of noise on the states of the system. One simple way to construct this matrix is to prepare all the states $|u\rangle$ and then use the measurement distribution to construct its probabilities. This technique is, however, circuit independent. This means the circuit that will be run is not taken into consideration when performing these measurements.

To extract the noiseless probability distribution, usually the inverse of the confusion matrix is calculated (the Moore-Penrose pseudoinverse) and then applied to the extracted probability distribution.

5.3.2 Zero Noise Extrapolation

Firstly introduced in 2017, Zero Noise Extrapolation is an error mitigation technique based on trying to extrapolate the noiseless result of a quantum circuit by analysing the circuit in different noise levels [13, 41, 25]. More technically, we need to estimate the expectation value of some observable with respect to an evolved state that is subject to noise. As the aforementioned article explains, they used Richardson extrapolation in order to extrapolate the zero-noise limit in short depth quantum circuits.

With this being said, let λ be the factor by which we increase the noise level. Following the notation from [23], we let τ quantify the noise level in the circuit and let $\tau' = \lambda \tau$ the scaled noise level. We can, therefore, see that for $\lambda = 1$, the input circuit remains unchanged. We let $\rho(\tau)$ be the density matrix representing the state prepared by the noise scaled quantum circuit. The expectation value of an observable A can therefore be represented as:

$$\langle E(\lambda) \rangle = Tr[\rho(\lambda \tau)A] \tag{20}$$

Therefore, if we measure for different values of $\lambda \geq 1$ we can try to extrapolate the zero-noise limit. Different functions can be used to extrapolate the result, the ones provided by [23] and that were used in the experiments are Richardson Extrapolation based and Polynomial extrapolation. Noise amplification, that means modifying the initial circuit for increased levels of λ can be done through gate folding on a global or local scale [13, 30]. We used local folding for our circuits. Pulse level gate implementations can also be used in order to control the noise introduced [18, 12].

5.4 Multiprogramming - An Efficient Use of the Quantum Chip

Whenever we want to run a certain experiment, one can take into consideration the efficient use of the quantum chip. Multiprogramming [7, 32, 34, 35] is a technique used for tackling the problem of hardware under-utilization in the NISQ era. Due to the circuits that are able to not be as affected by noise on the backend being restricted to smaller sizes and the hardware currently available, one would end up using only a small percentage of the qubits available in one run. For example, if we use the 2 qubit experiment, we want to run, and the chip of 128 qubits available through IBM, sending the experiments sequentially would result in the utilization of hardware of just 1.25%. Therefore, in our experiments, we chose to use multiprogramming for this reason. The main idea of this technique is highlighted in Figure 9.

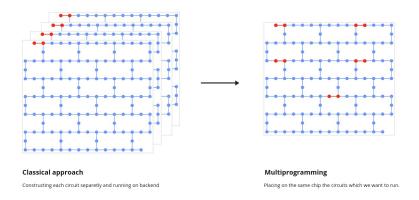


Figure 9: Multiprogramming allows for multiple circuits to run on the same chip

Despite the advantages, there are also some disadvantages to multiprogramming, one of which is interference. These interferences may occur due to the crosstalk introduced by additional operations and qubit measurement operations. Different layout strategies have been tested in order to limit this crosstalk. For example, there is evidence that in practice, for concurrent runs, crosstalk does not affect shorter circuits and for most runs, a physical buffer of one qubit between circuits is enough to reduce the effects of interference drastically if the circuit depth doesn't increase above 30 CX [35]. Layout strategies have been developed to therefore increase the usage of a chip [34], along with scheduling algorithms that follow the same aim of reducing crosstalk [7, 32]. For count-experiments, extracting the result for each circuit is straightforward. For statevector simulations, one can trace out the qubits that are not needed with the use of a partial trace over the density matrix of the system. If Q is the register of interest and A are the qubits for the other circuits, the density matrix of the system Q is extracted as in Equation 21.

$$\rho_Q \equiv Tr_A[\rho_{QA}] \tag{21}$$

6 End to End Run

This experiment aims to simulate quantum tunneling on a 2-qubit system. The barriers are placed on the highest order qubit, and the desired result is to see quantum tunneling between the two wells created. Moreover, we aim to optimize the circuit for running on hardware and use error mitigation techniques (ZNE in this case) to extract the best result possible of tunneling through the barrier. The multiprogramming paradigm was employed to optimize hardware chip utilization. The steps of this experiment are illustrated in Figure 10.

6.0.1 Methodology

For implementation, we used qiskit along with the tket compiler. For error mitigation techniques, we used mitiq [23]. For this experiment, the space was discretized using 2 qubits. We chose 7 timesteps and Δt of 0.1. As hardware, the 128 qubit ibm_osaka was used.

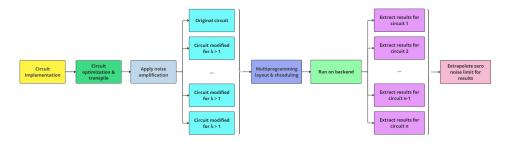


Figure 10: Workflow of the circuit

6.0.2 Experimental Process

Figure 11 presents the abstract circuit diagram. Here the Quantum Fourier Transform operations were grouped in the gate QFT, the same for the inverse Fourier Transform. The single Rz operator in between barriers represents the potential operator, and it implements the potential barrier. Mathematically, this operator is represented by (presented in section 4.2.2):

$$\hat{V} = e^{-iv\sigma_z^0 \Delta t} = I \otimes e^{-iv\sigma_z \Delta t}$$

Here, since we use qiskit for implementation we used qiskit ordering. This means that qubit 0 is the least-important qubit (uses little-endian convention). Therefore, this implements a potential barrier at $|01\rangle$ and $|11\rangle$.

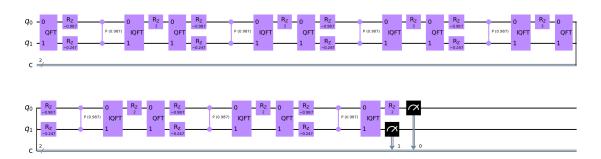


Figure 11: The circuit diagram

We can see the implemented evolution of the system in Figure 12. Tunneling is visible from well $|00\rangle$ to $|10\rangle$ (the particle starts at $|00\rangle$). This diagram shows the expected ideal output - it was calculated based on the density operators obtained by the statevectors taken at each timestep (based on ideal calculations, not noisy simulation or count distribution). Our wavefunction can now be written as $|\psi\rangle = \sum_{i=0}^{3} \alpha_i |k_i\rangle$, where k_i is the statevector representing the binary representation of index i and therefore the probability of being at a given k_i is α_i^2

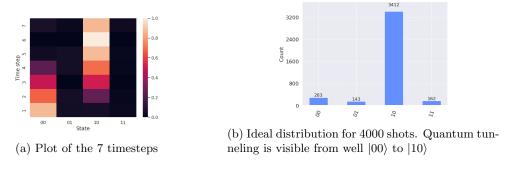


Figure 12: Experimental results for end-to-end process

After the implementation of the original circuit, we needed to apply the noise amplification technique. In this experiment, we opted for *local folding* and after conducting experiments with various factors, we determined scaling factors of [1.0, 1.5, 2.0, 2.5, 3.0] to be most effective.

As for the layout technique for multiprogramming, we chose to leave a buffer of at least 1 physical qubit, following empirical evidence of other studies [35]. We will also mention that the circuit has been compiled and prepared for backend run with the use of *qiskit* as well as *tket* compiler.

6.0.3 Analysis of Experimental Results

Following the run on ibm_osaka, the results for each noise amplified circuit are extracted and then used for inference of the zero noise limit. Results on noisy simulators are presented in Figure 13, along with the results from the circuits. Moreover, the results are concisely presented in Table 2

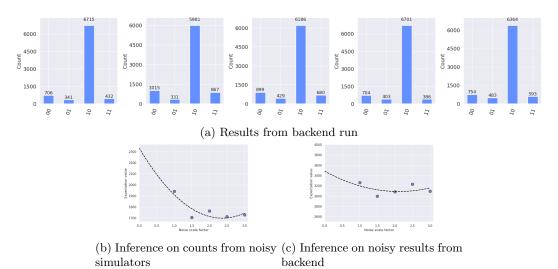


Figure 13: Experimental Results for End-to-End Process with Noise

Table 2: Results on Noisy simulator and on Real Backend

Type	T	T_{run}	E_1	T_{run}^{em}	E_2
simulation	0.864	0.64	0.224	0.798	0.066
real run	0.864	0.802	0.062	0.870	0.006

Where the following notations have been used:

T = the ideal probability of transmission; this entails finding the particle at location $|10\rangle$

 T_{run} = The result for the run (either on a noisy simulator or on the backend)

 E_1 = unmitigated error, this is $T - T_{run}$

 T_{run}^{em} = the transmission probability resulted from applying the error mitigation technique

 E_2 = The final error resulting after applying error mitigation, this is $|T - T_{run}^{em}|$

It is, therefore, evident that error mitigation had a crucial improvement over the simple, unmitigated run since we can observe that the inferred expectation value of our observable is only 0.006 off. We can see that error mitigation played a crucial role in both simulated and real environments. In the simulated environments, error was reduced from 0.224 to 0.066, only a quarter of the original error. On the real run, the error was reduced from 0.062 to 0.006, observing yet again a substantial decrease. One can also appreciate the accuracy of the transmission probability achieved. Moreover, using the multiprogramming paradigm, the chip was also efficiently used, since instead of using only 2 qubits out of the 128 qubits, five times in a row, all experiments were run on the same chip.

7 Extended Experimental Examination

The setup outlined in this article demonstrates versatility, and we believe that conducting additional experiments will further enhance understanding regarding result interpretation and the various components of the workflow outlined. Therefore, we conducted 2 additional experiments, which will be presented in this section. The first one is a 2 qubit simulation using Readout Error Mitigation and the second one is a 6 qubit simulation in a noise-free environment that employs the Hadamard Test in order to get a better understanding of the wavefunction during quantum tunneling.

For each experiment presented a timeline of probability will be firstly provided. A statevector simulator is used in order to get the probabilities at each timestep accurately. After the statevector from each timestep has been extracted, the probabilities are also extracted.

7.1 2 qubit simulation using REM

7.1.1 The objective(s) of the experiment

The objective of the first experiment is to simulate quantum tunneling on a 2-qubit system. The barriers are placed on the highest order qubit, and the desired result is to see quantum tunneling between the two wells created. Moreover, we aim to optimize the circuit for running on hardware and compare the different optimization procedures for this experiment. Concisely, this experiment follows:

- simulates quantum tunneling on a two-qubit system
- observe tunneling between the two wells
- optimize and run on hardware
- analyze the results

7.1.2 Theoretical foundation: relevant theoretical concepts for the experiment

For this example, implementation of the QFT, the potential and kinetic energy operators were needed. We used the implementation of the momentum operator without the use of an ancillary qubit since we didn't want to increase the depth of the circuit with swap gates required to mimic 3 qubit all-to-all connectivity.

7.1.3 Experiment Setup: Hardware, Software, and Key Parameters

As hardware, we used the ibm_belem, ibm_nairobi and ibm_osaka quantum computers, which have 5, 7 and 127 qubits, respectively. As of the 29th of November, the first two systems are retired, but we believe the results are still relevant due to the size of the experiment not being large. For the implementation, we used qiskit as the main SDK for implementing the operators discussed, along with the *tket* compiler for further optimization and mapomatic for a more detailed result of the layout and mapping scores of the circuit.

7.1.4 Discussion and Implications of Experimental Results

Firstly, we implemented the circuit with the help of qiskit. Figure 14 presents the abstract circuit diagram. Here the Quantum Fourier Transform operations were grouped in the gate QFT, the same for the inverse Fourier Transform. The single Rz operator in between barriers represents the potential operator, and it implements the potential barrier. Mathematically, this operator is represented by (presented in section 4.2.2):

$$\hat{V} = I \otimes e^{-iv\sigma_z^0 \Delta t}$$

Here, since we use qiskit for implementation we used qiskit ordering. This means that qubit 0 is the least-important qubit (uses little-endian convention). Therefore, this implements a potential barrier at $|01\rangle$ and $|11\rangle$.

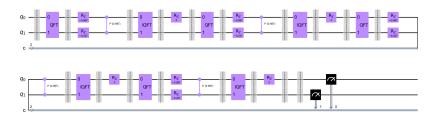


Figure 14: The circuit diagram

The evolution of the implemented system is illustrated in Figure 15. Tunneling is visible from well $|00\rangle$ to $|10\rangle$ (the particle starts at $|00\rangle$). This diagram shows the expected ideal output - it was calculated based on the density operators obtained by the statevectors taken at each timestep (based on ideal calculations, not noisy simulation or count distribution). Our wavefunction can now be written as $|\psi\rangle = \sum_{i=0}^{3} \alpha_i |k_i\rangle$, where k_i is the statevector representing the binary representation of index i and therefore the probability of being at a given k_i is α_i^2 .

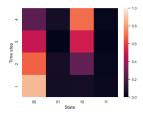


Figure 15: Plot of the 4 timesteps. Quantum tunneling is visible from well $|00\rangle$ to $|10\rangle$

7.1.5 Analysis of Experimental Results

Count distribution and probabilities from running on the qasm_simulator and Sampler primitive are presented in Figure 16. Measurements were made only after all steps had been executed, as seen in the circuit diagram in Figure 14, so represented below are the probabilities of the last step. Therefore, we see that the particle has tunneled from well $|00\rangle$ to $|10\rangle$ by the probability given for state $|10\rangle$.



Figure 16: Experimental results for the 2q experiment

Firstly, we tried running our circuit on the qiskit backend without further intervention in the transpilation process. The hardware used in this experiment was the 5-qubit quantum computer named ibm_belem (which has been retired) and the 127 quantum computer ibm_osaka. The results of this run are presented in Figure 18.



Figure 17: Backend run

We can now notice the difference between simulations and running on hardware with the use of Figures 16 and 18. Running the circuit on quantum computers has the disadvantage of being affected by error, since we are now in the NISQ era of quantum computing. The error can be seen in the results, for example, as erroneous counts on states $|01\rangle$ and $|11\rangle$. Errors can have multiple factors, ranging from

external noise affecting the states of the qubit to the routing and layout chosen by the transpiler not being the optimal solution for the given circuit. Because of this, we can try to optimise this circuit in order to make it more prone to errors. As presented in section 5.2, we can further optimize the passes presented. For this experiment, we chose to use the *tket* compiler as an add-on to the *qiskit* optimization passes in order to optimize the circuit further. Moreover, for the layouting technique, we used the *mapomatic* library and the routing from qiskit. By applying this optimization, the following result is obtained.



Figure 18: Experimental Results for 2 qubit simulation without error mitigation

Applying the same optimization for another backend, for example, for the 127-qubit Osaka quantum computer, the result in Figure 18b

7.1.6 Readout Error Mitigation

An error mitigation technique that can be used in practice is Readout Error Mitigation. It is based on constructing an assignment matrix similar to a confusion matrix, meaning it is constructed in order to show how often a result is measured wrongly when, in reality, the true outcome should be different. Local and Correlated readout error mitigators can be used, but here we chose to use Correlated readout error mitigators. Both the confusion matrix and the comparison are presented in Figure 19. The confusion matrix observed in Figure 19a has been constructed with the use of simple circuits (by this, we mean minimal in-depth, by using only additional X gates when needed) preparing each possible 2-qubit state. We can see in Figure 19b the results of the mitigated run. While the difference is not substantial, improvements can still be observed, for example, in positions where barriers should be.

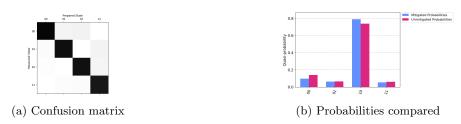


Figure 19: Experimental Results for 2 qubit simulation with REM

7.2 6 qubit Quantum Tunneling Simulation using the Hadamard Test

7.2.1 The objective(s) of the experiment

The objective of the experiment is to simulate quantum tunneling in a 6-qubit system. Here, the Hadamard test was used in order to "extract" (estimate) the real and imaginary part of the wavefunction rather than the probability of said wavefunction. Concisely, the objectives of the experiment were:

- implements the simulation circuit in a bigger 6-qubit system
- extract probability counts, observe tunneling
- extract real and imaginary part counts, observe how they evolve

7.2.2 Theoretical foundation: relevant theoretical concepts for the experiment

Apart from the QFT, potential, and kinetic energy operators, the Hadamard test is also needed for this experiment. Therefore, in order to simulate in one run and interpret the result, familiarity with how the Hadamard test works is needed.

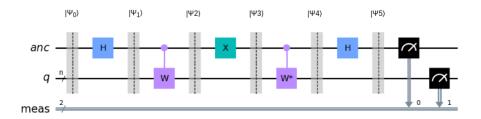


Figure 20: Hadamard test implementation

Hadamard test

The Hadamard test is used in case we want to extract the distribution of $Re\{\psi(x,t)\}^2$ and $Im\{\psi(x,t)\}^2$ separately. This is done with the help of controlled operations. Suppose we have a state of superposition:

$$|\psi\rangle = \sum_{i} \alpha_i |k_i\rangle$$

where α_i is, of course, a complex number and $|k\rangle$ represent the base states. Let W be the operator that takes the state from $|0\rangle^{\otimes n}$ to $|\psi\rangle$, that being $|\psi\rangle = W|0\rangle^{\otimes n}$. In order to extract the distribution of the real and imaginary parts, we will also use the complex conjugate of the operator W, W^* . An auxiliary qubit is also added. The Hadamard test is therefore represented:

- 1. Initialize the system in the state $|\psi_0\rangle = |0\rangle |0\rangle^{\otimes n}$
- 2. Apply Hadamard to ancillary qubit: $|\psi_1\rangle = |+\rangle |0\rangle^{\otimes n}$
- 3. Apply Controlled W operator: $|\psi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle W |0\rangle^{\otimes n} + |1\rangle |0\rangle^{\otimes n}) = \frac{1}{\sqrt{2}}(|0\rangle |\psi\rangle + |1\rangle |0\rangle^{\otimes n})$
- 4. Apply Not gate on ancillary qubit $|\psi_3\rangle = \frac{1}{\sqrt{2}}(|1\rangle |\psi\rangle + |0\rangle |0\rangle^{\otimes n})$
- 5. Apply Controlled W^* operator: $|\psi_4\rangle = \frac{1}{\sqrt{2}}(|1\rangle |\psi\rangle + |0\rangle W^* |0\rangle^{\otimes n}) = \frac{1}{\sqrt{2}}(|1\rangle |\psi\rangle + |0\rangle |\psi^*\rangle)$, where $W^* |0\rangle^{\otimes n} = |\psi^*\rangle$
- 6. Apply Hadamrd operator again $|\psi_5\rangle = \frac{1}{2}(|0\rangle |\psi\rangle |1\rangle |\psi\rangle + |0\rangle |\psi^*\rangle + |1\rangle |\psi^*\rangle) = \frac{1}{2}(|0\rangle (|\psi\rangle + |\psi^*\rangle) + |1\rangle (|\psi^*\rangle |\psi\rangle))$
- 7. Measuring the qubits, it can be observed that if the ancilla qubit is measured as 0, then the result from the second register defines $Re\{\psi(x,t)\}^2$, while if the ancilla qubit is 1, the $Im\{\psi(x,t)\}^2$ distribution can be extracted.

Schematically, the implementation is represented in Figure 20

7.2.3 Experiment Setup: Hardware, Software, and Key Parameters

Here we used qiskit for implementation. Moreover, we didn't need the tket compiler for hardware optimisation, since the depth of the circuit wouldn't allow for current free available quantum computers to run. Since the space here allows it, we will use as the initial wavefunction the Gaussian function. Since the depth of this circuit is large, we restrained to only simulating it on qasm_simulator. The initial wavepacket is represented as:

$$|\psi\rangle = \frac{1}{\sqrt{2\pi}0.8}e^{-\frac{1}{2}(\frac{x+2.25}{0.032})^2}e^{-300ix}$$

That is to say, the potential of the wave is 300. This number was chosen because we wanted a wave that traveled to the right. Mean, and standard deviations were chosen for the wavefunction to start before the barrier and not extend beyond it. The initial wavepacket is presented in Figure 21.

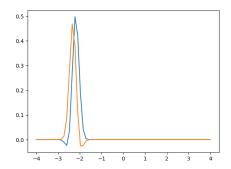


Figure 21: Initial wavepacket

7.2.4 Discussion and Implications of Experimental Results

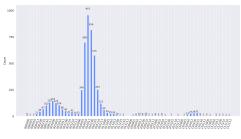
As presented in the Hadamard test section, the implementation of this type of function requires controlled operations and also an auxiliary qubit as the control qubit. The implementation of the state preparation, QFT, IQFT, potential and kinetic energy operators were grouped into the gate $time_evo$. We should mention that this includes all the timesteps from the troterrized form. The implementation of the momentum operator followed the schema with the auxiliary qubit, here noted as a. The circuit is presented in Figure 22.

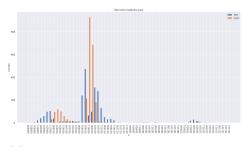


Figure 22: Implementation of Hadamard test. The working register q is of size 6. These are the qubits for which the operators were used. The auxiliary register contains one qubit a, which is used for the implementation of the momentum operator, and a qubit h needed for the Hadamard test - depending on this value, the operator or the conjugate operator was applied in order to extract the real/imaginary part. cr are the classical bits that hold the measurements of the 6 qubits. hc is of size 1. This holds the value of the auxiliary qubit h. Depending on this value, we know that we measured the imaginary or real part of the wavefunction.

7.2.5 Analysis of Experimental Results

The simulation was run on qasm_simulator and the Figure 23 shows probability distribution and imaginary and real counts distribution of the evolved state. The potential barrier is placed at positions xxx11x (the implementation of this potential barrier is discussed in Section 4.4), and we can clearly see tunneling of the wavepacket after the potential barrier. Moreover, we can observe the real and imaginary distributions in Figure 23b. The transmission wave is clearly illustrated in this image, and through these larger simulations in terms of qubits used, which means a more fine discretization of space, one can try to approach real simulations more accurately.





(a) Counts probability distribution

(b) Real and imaginary count distributions

Figure 23: Experimental Results for Hadamard Test Experiment

8 Conclusions

In this article, we have outlined the primary procedures necessary for translating the quantum mechanical challenge of simulating the Schrödinger equation into a quantum computing framework. That is, we began by examining the theoretical implications of simulating quantum tunneling and subsequently deliberated on diverse methodologies for operator implementation.

Moreover, we focused on the changes required in order to efficiently and accurately run the abstract circuit on a real backend. We discussed what the transpilation procedure entails and why a call for hardware-aware design is still necessary in the NISQ era in the context of superconducting architectures. Multiple compilers have been employed in order to optimize the circuit for the chosen backend, and they proved capable of efficient optimization. Error mitigation has also been discussed, and ZNE and REM are used to improve the simulation's results. For today's chip's hardware underutilization problem, multiprogramming was harnessed to optimally run the circuit along with the noise-amplified circuits alongside the aforementioned error mitigation techniques. Hadamard tests for 6-qubit systems were also simulated to enrich the understanding of the topic.

Our research yields multifaceted implications. Through the successful simulation of quantum tunneling within a comprehensive workflow, we achieve precise results in a two-qubit simulation. We assert that this adaptable workflow offers significant value across diverse simulation contexts, thereby providing universal benefits to researchers.

References

- [1] Mohamed Abouelela. Quantum simulation of the schrodinger equation using ibm's quantum computers. Bachelor's thesis is physics, 2020.
- [2] Martin Beisel, Johanna Barzen, Frank Leymann, Felix Truger, Benjamin Weder, and Vladimir Yussupov. Configurable readout error mitigation in quantum workflows. *Electronics*, 11(19), 2022.
- [3] Giuliano Benenti and Giuliano Strini. Quantum simulation of the single-particle schrödinger equation. American Journal of Physics, 76(7):657–662, jul 2008.
- [4] Zhenyu Cai, Ryan Babbush, Simon C. Benjamin, Suguru Endo, William J. Huggins, Ying Li, Jarrod R. McClean, and Thomas E. O'Brien. Quantum error mitigation. *Reviews of Modern Physics*, 95(4), December 2023.
- [5] J. Ignacio Cirac and Peter Zoller. Goals and opportunities in quantum simulation. *Nature Physics*, 8:264 266, 2012.
- [6] Andrew J. Daley, Immanuel Bloch, C. Kokail, Stuart Flannigan, N Pearson, Matthias Troyer, and Peter Zoller. Practical quantum advantage in quantum simulation. *Nature*, 607:667 676, 2022.
- [7] Poulami Das, Swamit S. Tannu, Prashant J. Nair, and Moinuddin Qureshi. A case for multi-programming quantum computers. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '52, page 291–303, New York, NY, USA, 2019. Association for Computing Machinery.

- [8] Ish Dhand and Barry C Sanders. Stability of the trotter–suzuki decomposition. *Journal of Physics A: Mathematical and Theoretical*, 47(26):265206, June 2014.
- [9] Suguru Endo, Simon C. Benjamin, and Ying Li. Practical quantum error mitigation for near-future applications. *Physical Review X*, 8(3), July 2018.
- [10] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6–7):467–488, Jun 1982.
- [11] Daniel Stilck França, Liubov A. Markovich, V. V. Dobrovitski, Albert H. Werner, and Johannes Borregaard. Efficient and robust estimation of many-qubit hamiltonians, 2022.
- [12] J. W. O. Garmon, R. C. Pooser, and E. F. Dumitrescu. Benchmarking noise extrapolation with the openpulse control framework. *Phys. Rev. A*, 101:042308, Apr 2020.
- [13] Tudor Giurgica-Tiron, Yousef Hindy, Ryan LaRose, Andrea Mari, and William J. Zeng. Digital zero noise extrapolation for quantum error mitigation. In 2020 IEEE International Conference on Quantum Computing and Engineering (QCE). IEEE, October 2020.
- [14] Sean Greenaway, Adam Smith, Florian Mintert, and Daniel Malz. Analogue quantum simulation with fixed-frequency transmon qubits. *Quantum*, 8:1263, February 2024.
- [15] David J. Griffiths and Darrell F. Schroeter. *Introduction to Quantum Mechanics*. Cambridge University Press, 3 edition, 2018.
- [16] Naomichi Hatano and Masuo Suzuki. Finding Exponential Product Formulas of Higher Orders, page 37–68. Springer Berlin Heidelberg, November 2005.
- [17] Narendra N. Hegade, Nachiket L. Kortikar, Bikramaditya Das, Bikash K. Behera, and Prasanta K. Panigrahi. Experimental demonstration of quantum tunneling in ibm quantum computer, 2021.
- [18] Ivan Henao, Jader Santos, and Raam Uzdin. Adaptive quantum error mitigation using pulse-based inverse evolutions. npj Quantum Information, 9, 11 2023.
- [19] He-Liang Huang, Dachao Wu, Daojin Fan, and Xiaobo Zhu. Superconducting quantum computing: a review. *Science China Information Sciences*, 63(8), July 2020.
- [20] Valay Jain, Bikash Behera, and Prasanta Panigrahi. Quantum simulation of discretized harmonic oscillator on ibm quantum computer. 07 2019.
- [21] Shubham Kumar, Rahul Pratap Singh, Bikash K. Behera, and Prasanta K. Panigrahi. Quantum simulation of negative hydrogen ion using variational quantum eigensolver on ibm quantum computer, 2019.
- [22] Michael Kühn, Sebastian Zanker, Peter Deglmann, Michael Marthaler, and Horst Weiß. Accuracy and resource estimations for quantum chemistry on a near-term quantum computer. *Journal of Chemical Theory and Computation*, 15(9):4764–4780, August 2019.
- [23] Ryan LaRose, Andrea Mari, Sarah Kaiser, Peter J. Karalekas, Andre A. Alves, Piotr Czarnik, Mohamed El Mandouh, Max H. Gordon, Yousef Hindy, Aaron Robertson, Purva Thakre, Misty Wahl, Danny Samuel, Rahul Mistri, Maxime Tremblay, Nick Gardner, Nathaniel T. Stemen, Nathan Shammah, and William J. Zeng. Mitiq: A software package for error mitigation on noisy quantum computers. Quantum, 6:774, August 2022.
- [24] Juha Leppäkangas, Nicolas Vogt, Keith R. Fratus, Kirsten Bark, Jesse A. Vaitkus, Pascal Stadler, Jan-Michael Reiner, Sebastian Zanker, and Michael Marthaler. Quantum algorithm for solving open-system dynamics on quantum computers using noise. *Physical Review A*, 108(6), December 2023.
- [25] Ying Li and Simon C. Benjamin. Efficient variational quantum simulator incorporating active error minimization. *Phys. Rev. X*, 7:021050, Jun 2017.
- [26] Haoran Liao, Derek S. Wang, Iskandar Sitdikov, Ciro Salcedo, Alireza Seif, and Zlatko K. Minev. Machine learning for practical quantum error mitigation, 2023.

- [27] Yiding Liu, Zedong Li, Alan Robertson, Xin Fu, and Shuaiwen Leon Song. Enabling efficient real-time calibration on cloud quantum machines. *IEEE Transactions on Quantum Engineering*, 4:1–17, 2023.
- [28] Seth Lloyd. Universal quantum simulators. Science, 273:1073 1078, 1996.
- [29] Daniel Loss and David P. DiVincenzo. Quantum computation with quantum dots. *Phys. Rev. A*, 57:120–126, Jan 1998.
- [30] Ritajit Majumdar, Pedro Rivero, Friederike Metz, Areeq Hasan, and Derek S Wang. Best practices for quantum error mitigation with digital zero-noise extrapolation, 2023.
- [31] Alexander J. McCaskey, Zachary P. Parks, Jacek Jakowski, Shirley V. Moore, Titus D. Morris, Travis S. Humble, and Raphael C. Pooser. Quantum chemistry as a benchmark for near-term quantum computers. *npj Quantum Information*, 5(1), Nov 2019.
- [32] Prakash Murali, David C. Mckay, Margaret Martonosi, and Ali Javadi-Abhari. Software mitigation of crosstalk on noisy intermediate-scale quantum computers. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20. ACM, March 2020.
- [33] Michael A. Nielsen and Isaac L. Chuang. Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press, 2011.
- [34] Siyuan Niu and Aida Todri-Sanial. Enabling multi-programming mechanism for quantum computing in the nisq era. *Quantum*, 7:925, February 2023.
- [35] Yasuhiro Ohkura, Takahiko Satoh, and Rodney Van Meter. Simultaneous execution of quantum circuits on current and near-future nisq systems. *IEEE Transactions on Quantum Engineering*, 3:1–10, 2022.
- [36] John Preskill. Quantum computing in the nisq era and beyond. Quantum, 2:79, August 2018.
- [37] Afonso Rodrigues. Validation of quantum simulations: Assessing efficiency and reliability in experimental implementations. PhD thesis, Universidade do Minho, 2018.
- [38] Sina Shokri, Shahnoosh Rafibakhsh, Roghayeh Pooshgan, and Rita Faeghi. Implementation of a quantum algorithm to estimate the energy of a particle in a finite square well potential on IBM quantum computer. The European Physical Journal Plus, 136(7), jul 2021.
- [39] Andrew T. Sornborger. Quantum simulation of tunneling in small systems. *Scientific Reports*, 2(1), aug 2012.
- [40] Francesco Tacchino, Alessandro Chiesa, Stefano Carretta, and Dario Gerace. Quantum computers as universal quantum simulators: State-of-the-art and perspectives. *Advanced Quantum Technologies*, 3(3), dec 2019.
- [41] Kristan Temme, Sergey Bravyi, and Jay M. Gambetta. Error mitigation for short-depth quantum circuits. *Physical Review Letters*, 119(18), November 2017.
- [42] J. Tersoff and D. R. Hamann. Theory of the scanning tunneling microscope. *Phys. Rev. B*, 31:805–813, Jan 1985.
- [43] Bo Yang, Rudy Raymond, and Shumpei Uno. Efficient quantum readout-error mitigation for sparse measurement outcomes of near-term quantum devices. *Physical Review A*, 106(1), July 2022.
- [44] Zhaoxuan Zhu, Shengjie Yu, Dean Johnstone, and Laurent Sanchez-Palencia. Localization and spectral structure in two-dimensional quasicrystal potentials, 2024.