ADVANCED QUANTUM TECHNOLOGIES
www.advquantumtech.com

Quantum Tunneling: From Theory to Error-Mitigated Quantum Simulation

*Sorana Catrina and Alexandra Băicoianu**

Ever since the discussions about a possible quantum computer arised, quantum simulations have been at the forefront of possible utilities, with the task of quantum simulations being one that promises quantum advantage. Recently, advancements have made it feasible to simulate complex molecules using Variational Quantum Eigensolvers or study the dynamics of many-body spin Hamiltonians. These simulations have the potential to yield valuable outcomes through the application of error mitigation techniques. Simulating smaller models carries a great amount of importance as well and currently, in the Noisy Intermediate Scale Quantum era, is more feasible since it is less prone to errors. The objective of this work is to examine the theoretical background and the circuit implementation of a quantum tunneling simulation, with an emphasis on hardware considerations. This study presents the theoretical background required for such implementation and highlights the main stages of its development. By building on classic approaches of quantum tunneling simulations, this study aims at improving the result of such simulations by employing error mitigation techniques, Zero Noise Extrapolation, and Readout Error Mitigation and uses them in conjunction with multiprogramming of the quantum chip, a technique used for solving the hardware under-utilization problem that arises in such contexts.

1. Introduction

Scientists have always been fascinated by the simulation of natural phenomena and ever since Feynman's assertion that quantum mechanics cannot be efficiently simulated on a classical computer, quantum computers have been employed for such tasks.[1,2] Current simulations on quantum computers are, however, susceptible to errors. Multiple techniques have been developed to deal with these impediments[3] and, moreover, are tailored to current NISQ (Noisy Intermediate Scale Quantum) era devices. Specifically, one may attempt to mitigate the noise in the system through hardware improvements and calibrations[4] or by designing algorithms that leverage the structure of noise in order to retrieve the valuable information that is typically represented by the estimated value at zero noise level. This can be achieved either through classical post-processing techniques such as ZNE (Zero Noise Extrapolation)[5,6] or REM (Readout Error Mitigation),[7,8] or even by employing machine learning algorithms.[9] Regarding the fields that have the possibility to benefit greatly from quantum supremacy, quantum simulations for quantum chemistry in one such example.[10,11] The study of different materials through quantum simulations can also prove to be an important field that would benefit from quantum advantage.[12] Implementations of such tasks that consider current limitations of quantum computers already exist, and range from quantum simulations of small molecules through Variational Quantum Eigensolvers to open systems modeling.[13–16] The objective of the present article is to simulate quantum tunneling, a quantum phenomenon, on a quantum computer. Additionally, the circuit that has been developed also serves as an effective tool for pushing the boundaries of the capabilities of existing hardware, given the potential to increase the depth of the circuit by simulating over longer periods of time. Prior quantum simulations of tunneling have predominantly concentrated on a single approach to perform the given task.[17–19] The current study seeks to provide insight into a variety of circuit implementation alternatives by focusing on hardware run considerations for superconducting architectures. Additionally, we are of the opinion that it is imperative to highlight the compiler's importance in relation to the abstract circuit that was employed for this task. In addition, error mitigation techniques that have proved to be efficient are also discussed. Multiprogramming, a technique aimed at addressing the issue of hardware under-utilization, is also employed along with the Error Mitigation techniques chosen to complete the proposed end-to-end workflow. The results of these investigations are also presented, with an expectation value for the transmission probability of the

S. Catrina
Faculty of Mathematics and Computer Science
Transilvania University of Brasov
Eroilor, 29, Brasov 500036, Romania
A. Băicoianu
Department of Mathematics and Computer Science
Transilvania University of Brasov
Eroilor, 29, Brasov 500036, Romania
E-mail: a.baicoianu@unitbv.ro

DOI: 10.1002/qute.202400163

particle through the potential barrier that is only 0.006 off after error mitigation, about 10x smaller then the unmitigated result.

2. Experimental Section

This section provides a comprehensive overview of the theoretical framework, implementation strategies, procedural methodologies, and hardware intricacies that are relevant to our research. Concretely, this section starts by presenting the physics context necessary for comprehension of implementation, and subsequently proceeds to outline the procedures required to adapt this theory to the context of quantum computation (discretization of time and space, as well as implementation details). The final subsection of this section provides a comprehensive discussion of the details required for an efficient hardware implementation of the circuit, following the presentation of the implemented circuit.

2.1. Theoretical Background

Having delineated the primary steps to be addressed, we now proceed to the theoretical foundation required for the comprehension and, consequently, the implementation of a quantum tunneling simulation. Therefore, this section will focus on the Physics background required for understanding this effect, namely, the Schrödinger equation. The evolution of the wavefunction of a quantum mechanical system is described by the Schrödinger equation:

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

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 clear that this is 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 total energy of the quantum system. For a 1D system, the Laplace operator takes the following form:

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

It should be noted that based on the potential energy, the eigenspectrum of the wavefunction takes different forms. In this study, a square-well potential will be considered and presented in Section 2.1.1.

The formalism of quantum mechanics states that the wavefunction, previously denoted as $\psi(x,t)$ is a statevector, $|\psi\rangle$, belonging to a Hilbert Space, $\mathcal{H}$. That is, the wavefunction $\psi(x,t)$ is the representation of the vector $|\psi\rangle$ in the position basis ($\psi(x,t) = \langle x|\psi(t)\rangle$). Therefore, in Dirac notation, the vector representing the state of the system is independent on the basis used. Griffith's Introduction to Quantum Mechanics provides additional information regarding the formalism of quantum mechanics.[20] In Dirac notation, the time-dependent Schrödinger equation takes the following form:

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle \tag{3}$$

$$\hat{H} = \hat{K} + \hat{V} \tag{4}$$

where

$|\psi(t)\rangle$ = the statevector of the quantum system
$\hat{H}$ = the Hamiltonian operator
$\hat{V}$ = the potential energy operator
$\hat{K}$ = the kinetic energy 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 (5) is an eigenvalue equation, therefore $|\psi(t)\rangle$ is an eigenfunction of the *Hamiltonian operator* with corresponding eigenvalues E.

$$\hat{H}|\psi(t)\rangle = E|\psi(t)\rangle \tag{5}$$

$$|\psi(t)\rangle = \hat{U}|\psi(0)\rangle = e^{-i\hat{H}t/\hbar}|\psi(0)\rangle \tag{6}$$

where

$|\psi(t)\rangle$ = the statevector of the quantum system
$\hat{H}$ = the Hamiltonian operator
E = the set of possible energies of the system
$\hat{U}$ = the time evolution operator

In the context of quantum computation, the equations discussed in this subsection will need to be implemented. Nevertheless, the quantum tunneling phenomenon that is of interest to this study is readily apparent when the potential energy operator observed in the Schrödinger equation is of a unique form. The subsequent section addresses the usefulness of this phenomenon and the form of the potential needed.

2.1.1. Quantum Tunneling

Quantum tunneling is a unique phenomenon to quantum mechanics that defies any explanation through classical mechanics models, and has shaped our understanding of the world around. Its significance extends across various fields and applications, some examples being Scanning Tunneling Microscopy[21] and Josephson Junctions which are pivotal to the development of superconducting quantum computers.[22] Quantum dots also harness the principles of quantum tunneling and are used in other quantum computer architectures, such as spin qubit quantum

computers, first proposed in ref. [23]. This underscores the critical necessity to delve into the study of this effect, given its far-reaching implications across numerous fields. The tunneling effect occurs between two wells separated by a potential barrier. That is, the potential takes the form presented in Equation (7), where a is the width of the potential barrier. When solving the Schrödinger equation for the given potential and a kinetic energy for the particle, the wavefunction that results is one that tunnels between the wells. This phenomenon will be visible in the simulations presented beginning with Section 2.2.6.

$$V(x) = \begin{cases} 0, x < 0 \\ V_0, 0 \le x \le a \\ 0, x > a \end{cases} \quad a = \text{the width of the potential barrier} \tag{7}$$

In the subsequent section, we will concentrate on the implementation aspects that are necessary to adjust this theoretical framework to the context of quantum computing, keeping these theoretical underpinnings in mind.

2.2. Implementation Considerations

In order to effectively simulate quantum tunneling in a quantum computing environment, it is necessary to take several steps. These stages primarily assist in the implementation of the Schrödinger equation that was previously discussed. To accomplish this, it is necessary to first implement the *Hamiltonian operator* and model the initial wavefunction $|\psi(0)\rangle$. This is crucial for executing both Equations (5) and (6). To accomplish this, both space and time discretization are needed, since quantum computers operate with a fixed number of qubits, demanding the discretization of infinite space for accurate simulation.

Furthermore, as Equation (6) suggests, the system must be simulated 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 subsections, explore the implementation of the aforementioned equations are explored on a quantum computer and examine various approaches as necessary. In Section 2.2.1, the discretization process is examined in greater detail and the requisite adaptations essential for correctly and efficiently simulating quantum tunneling on a quantum computer are explored, such as the Trotter–Suzuki approximation.[24,25] Immediately following this, Section 2.2.2 will describe the implementation of the kinetic and potential energy operators and examine the numerous alternatives that are available for this purpose. The preparation of the initial state is addressed in Section 2.2.4, while QFT is the subject of Section 2.2.3. In Section 2.2.5, the final circuit overview is provided, and the final subsection of this section, Section 2.2.6, presents the results of a noiseless simulation.

2.2.1. Time and Space Discretization

The quantum tunneling simulation is implemented on a superconducting quantum computer that has a restricted number of qubits. Consequently, the Schrödinger equation necessitates discretization. Beginning with the efficient implementation of the time evolution operator and emphasizing the discretization of space, this subsection will delineate the primary stages of this process within the context of this type of simulation.

Time Discretization

Considering the time evolution of the system, in order to simulate the progression of the system, the *time evolution operator* needs to be implemented. Equation (6) can be further expanded to highlight the use of the kinetic and potential energy operators and how the state vector changes after Δt time. This is illustrated in Equation (8).

$$|\psi(t+\Delta t)\rangle = e^{-i\hat{H}\Delta t}|\psi(t)\rangle = e^{-i(\hat{K}+\hat{V})\Delta t}|\psi(t)\rangle \tag{8}$$

where

Δt = the amount of time allowed for the system to evolve after time t

It is important to notice that if the operator $e^{(\hat{K}+\hat{V})\Delta t}$ is implemented for a small enough Δt we are able to simulate the time evolution of the entire system by sequentially applying this operator. However, an issue arises when attempting to utilize Equation (8) by decomposing into terms containing the kinetic and potential energy operators, and this is highlighted in Equation (9).

$$e^{-i\hat{H}\Delta t} = e^{-i(\hat{K}+\hat{V})\Delta t} \neq e^{-i\hat{K}\Delta t}e^{-i\hat{V}\Delta t} \tag{9}$$

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, generally, commute. Therefore, we will need to use the Suzuki–Trotter approximation. Primarily, the first order formula was used given by Equation (10), but higher order approximations may also be used.[24] For example, the second order Suzuki–Trotter approximation is given by Equation (11).[25]

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

$$e^{-i\hat{H}\Delta t} \approx e^{-i\hat{V}\frac{\Delta t}{2}}e^{-i\hat{K}\Delta t}e^{-i\hat{V}\frac{\Delta t}{2}} + O((\Delta t)^3) \tag{11}$$

Space Discretization

Since the Schrödinger equation is given in continuous space, we encounter a challenge in implementation due to the finite number of qubits available. Therefore, the interval $0 \le x \le L$ could be discretized 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 (12) where k represents the particle location corresponding to binary number k.

$$|\psi(x+L,t)\rangle = \sum_{k=0}^{2^n-1} \psi(x_k,t)|k\rangle \tag{12}$$

2.2.2. Implementing the Operators

After discussing the approximation of the time evolution operator using the individual potential and kinetic energy operators, this section will delineate the methods by which we express and implement these operators. Therefore, in accordance with the Suzuki–Trotter approximation, to simulate quantum tunneling, the following operators need to be implemented:

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

The Kinetic Energy Operator

The kinetic energy operator can be represented with the help of the momentum operator as shown in Equation (13), where $\hat{p}$ is the momentum operator.

$$\hat{K} = \frac{\hat{p}^2}{2m} \tag{13}$$

where

$\hat{p}$ = the momentum operator
m = the mass of the particle

Different from coordinate space, in momentum space $\hat{p}$ is diagonal. However, the potential energy operator is dependent on $\hat{x}$, which is diagonal in the coordinate space. Consequently, it is advantageous to switch from coordinate space to momentum space to apply the kinetic energy operator and return to coordinate space to apply the potential energy operator (this is because the construction of diagonal operators is easier). In quantum mechanics, switching from coordinate space to momentum space is done by means of the Fourier Transform, as shown in Equation (14). The inverse transformation is done via Equation (15).

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

$$\psi(x,t) = \frac{1}{\sqrt{2\pi\hbar}} \int_{-\infty}^{\infty} dp \tilde{\psi}(p,t) * e^{\frac{ipx}{\hbar}} \tag{15}$$

Considering the same limitations, we will need to use the discrete formula of the Fourier Transform. Moreover, we need the quantum implementation of this algorithm, namely the Quantum Fourier Transform (QFT), shown in Equation (16). The inverse Quantum Fourier Transform (IQFT) would be used to convert back to coordinate space.

$$|j\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} |k\rangle$$
$$\omega_N^{jk} = e^{2\pi i \frac{jk}{N}} \tag{16}$$

As mentioned earlier, in momentum space, the kinetic energy operator is diagonal. Using the notation in Equation (17), the operator $e^{-i\hat{k}^2\Delta t}$ can be represented with the use of the phase gate, as shown in Equation (18) with the Pauli Z gate.[17,26]

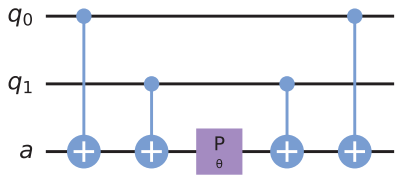


Figure 1. $Z_1 \otimes Z_2$ Hamiltonian implementation using ancilla qubits. The qubits q_0 and q_1 are the two qubits for which the Hamiltonian is implemented. Qubit a is the auxiliary qubit that is employed to facilitate computation.

$$|k\rangle = \sum_{i=0}^{n} 2^i j_{i+1} \tag{17}$$

$$e^{-i\hat{k}^2\Delta t} = exp\left(\frac{i\phi}{2^{2n-3}}(1 + \sum_{j=1}^{n} 2^{j-1}\hat{Z}_j)^2\right) \tag{18}$$

With this formulation, the one qubit operations could be easily observed. However, in order to better examine the two-qubit operations, the sum had to be opened. Focusing on the two-qubit terms, this part can be represented as in Equation (19).

$$S = \sum_i \sum_j 2\frac{2^{j-1}2^{i-1}}{2^{2n-3}}\hat{Z}_i\hat{Z}_j = \sum_i \sum_j \frac{2^{i+j-1}}{2^{2n-3}}\hat{Z}_i\hat{Z}_j$$
$$= \sum_i \sum_j \frac{1}{2^{2n-i-j-2}}\hat{Z}_i\hat{Z}_j \tag{19}$$

Since in this notation, i and j start from 1 and when implementing indexing starts from 0, we need to change the start and ending point and rewrite the sum as in Equation (20).

$$i \to i' + 1$$
$$j \to j' + 1$$
$$S = \sum_{i'} \sum_{j'} \frac{1}{2^{2n'-(i'+1)-(j'+1)-2}}\hat{Z}_i\hat{Z}_j \tag{20}$$
$$= \sum_{i'} \sum_{j'} \frac{1}{2^{2n'-i'-j'-4}}\hat{Z}_i\hat{Z}_j$$

The implementation of this operator can therefore be accomplished in various ways. One approach is to follow the schema using auxiliary qubits,[27] but one can also manage without such auxiliary qubits in simulations of two and three qubits.[18,19] An auxiliary qubit is used 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 even). This is highlighted in Algorithm 1 and illustrated for an example Hamiltonian in **Figure 1**. The first section of the algorithm concerns the single-qubit rotations, while the second is intended for the two-qubit terms in the above-mentioned equations. We mention that lines 9–10 are

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

```
for i ∈ range(n) do
    circuit.p(ϕ/2^(2n−3), q[i]) ▷ The single qubit rotations from Equation (18)
end for
for i ∈ range(n) do
    for j ∈ range(i + 1, n) do
        apply cx on q[i], aux
        apply cx on q[j], aux
        apply phase gate with angle 1/2^(2n−i−j−4) on aux ▷ The two-qubit rotations
        apply cx on q[j], aux
        apply cx on q[i], aux
    end for
end for
```

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. Further information on implementations considering hardware constraints will be provided in Section 2.3.

Potential Energy Operator

Apart from the kinetic energy operator, the Hamiltonian is also composed of the potential energy operator. In order to simulate quantum tunneling, as discussed in Section 2.1.1, one could simulate a well type potential. Implementation of the potential energy operator is then straightforward in coordinate representations: it is done by applying Z rotations to the qubit (or CZ operations to different qubits), depending on the type of potential required. It is important to note that the *qiskit* ordering will be followed, that is, little endian. For example, the controlled Pauli Z operator could be applied on the highest-order qubit to implement multiple wells, as shown in the equation below. Similarly, to have only one well, the lowest-order qubit could be applied to it.

$$e^{i\hat{V}\Delta t} = e^{-iv\sigma_z^{n-1}\Delta t} = I \otimes I \otimes I \otimes \ldots \otimes \sigma_z \tag{21}$$

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

As a summary of this section, the efficient representation and therefore implementation of the kinetic and potential energy operators are discussed. However, these operators are efficiently implemented in either coordinate space, for the potential operator, or in momentum space respectively for the kinetic energy operator. As previously mentioned, the Quantum Fourier Transform enables the transition between these two representations. Consequently, the subsequent subsection will address this topic.

2.2.3. Quantum Fourier Transform

The Quantum Fourier Transform is used to allow us to switch between coordinate space and momentum space representations. Represented mathematically, the QFT maps a quantum state $|X\rangle$ onto another quantum state $|Y\rangle$ by the formulas from Equation (22).

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

$$|Y\rangle = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} y_j |j\rangle \tag{22}$$

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

In order to implement the QFT, we can represent it as follows[27] in Equation 23.

$$\begin{aligned} QFT\,|x\rangle &= \frac{1}{\sqrt{N}} e^{2\pi i \frac{jk}{n}} x_j \\ &= \frac{1}{\sqrt{N}} (|0\rangle + e^{\frac{2\pi i}{2}x}|1\rangle) \otimes (|0\rangle + e^{\frac{2\pi i}{2^2}x}|1\rangle) \otimes \ldots \otimes (|0\rangle \\ &\quad + e^{\frac{2\pi i}{2^n}x}|1\rangle) \end{aligned} \tag{23}$$

2.2.4. Initial State

Having discussed the necessary algorithms for implementing the time evolution, now turn to prepare the initial state from which the system will have to evolve. Since this study aims for single particle simulations, an initial state that encodes the state of this initial particle was needed, such as the starting point of this particle.

In order to facilitate implementation, simulations are opted where the potential is $e^{i\hat{V}\Delta t} = e^{-iv\sigma_z^0\Delta t}$ (meaning a well is present on every other site), the initial state to be $|1\rangle$ where the state to start was needed. For every other type of potential, a Gaussian wave is used as the main form:

$$|\psi\rangle = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{24}$$

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

$$|\psi\rangle = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} e^{-ipx} \tag{25}$$

This works well for simulations with barriers since the wave is confined. A Gaussian wave is also used for free particle simulations. In quantum mechanics, due to the need for normalization,

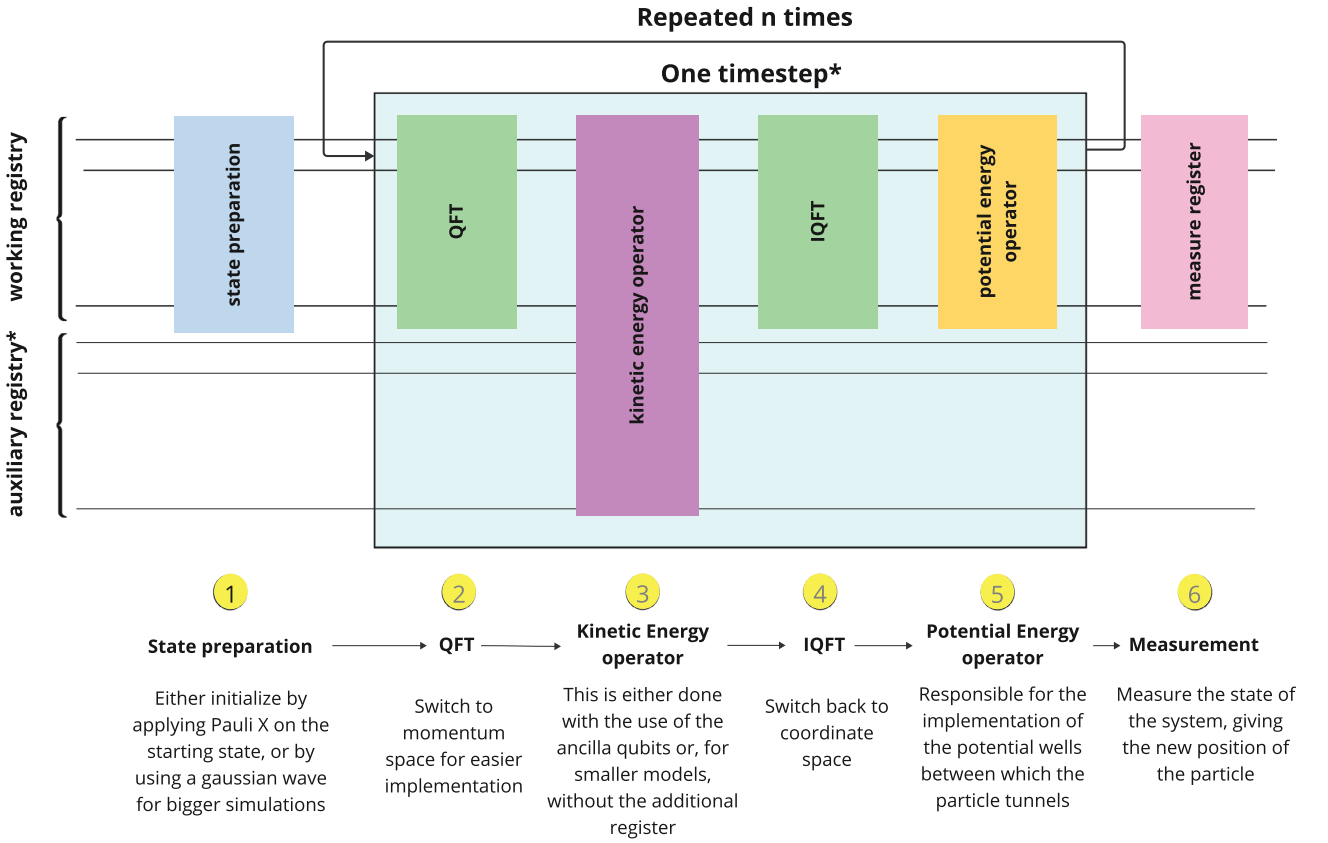


Figure 2. Full circuit overview of the quantum simulation. It starts with the state preparations of the system, after which the time evolution is implemented. In one time step, the implementation of the kinetic energy operator (by means of the QFT for switching between basis) and the potential energy operator are represented. At the end, measuring the register gives the final state of the system.

the wavepacket is represented as a superposition of such plane waves.

$$\psi(x,t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dk \tilde{\varphi}(k) e^{-\omega(k)t} e^{ikx} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dk \tilde{\varphi}(k) e^{ikx} \quad (26)$$

After a comprehensive explanation of the individual components required to simulate quantum tunneling on a quantum computer, it is now possible to integrate all of these implementations to construct the final circuit. Consequently, the subsequent section will delineate the specific applications of each algorithm that has been examined in the preceding sections.

2.2.5. Circuit Overview

After analyzing the initial state and the implementations of the kinetic and potential energy operators in the preceding sections, now able to offer a comprehensive overview of the circuit implementation. Simulation of Equation (6) is intended on the quantum computer. Upon examining the operators' implementation, an extensive overview of the critical stages required for the circuit's development could be provided, as illustrated in **Figure 2**. These steps are as follows:

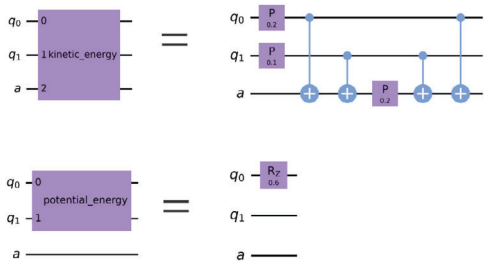
Step 1: This is represented by state preparation, which entails preparing $|\psi(x,0)\rangle$, as discussed in Section 2.2.4. This state can be a Gaussian wave for systems where the space has been discretized using more qubits. Otherwise, the wave is approximated to the state containing a flipped qubit on the position of the particle After this preparation, one can start implementing the time evolution with the use of the Suzuki–Trotter approximation. The Kinetic and Potential energy operators are implemented in steps 2–5, which represent a single time increment.

Step 2: The QFT is applied to the quantum circuit. As mentioned in Section 2.2.2, one will first switch to momentum space with the use of the QFT and then implement the kinetic operator.

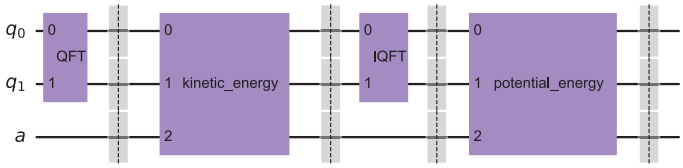
Step 3: The Kinetic Energy Operator is implemented. This can be done either with or without ancilla qubits.

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 2.2.2.



(a) The implementation of kinetic and potential energy operators for two qubit systems. An ancilla qubit is used to implement the kinetic energy operator. A phase gate is applied to the first qubit to create two wells of potential by the potential energy operator.



(b) Implementation of a one timestep for a two qubit system. Initially, the QFT is employed to transition to the momentum space. Subsequently, the kinetic energy operator is implemented. In order to return to coordinate space, the potential energy operator is implemented, and the inverse QFT is applied.

Figure 3. Overview of two-qubit operator implementations.

Step 6: The last step is measuring the working register, which provides the final state of the particle.

For implementation, *qiskit* was used, but since this is compiled to QASM, the code can be reused for other languages as well. The representation of operators for a two-qubit example, as presented in the first section, is shown in **Figure 3**.

A high-level view of the circuit for a two-qubit implementation, where an ancilla qubit is used for the kinetic energy implementation, is present in Figure 3b following the scheme presented in Section 2.2.2 with ancilla qubits.[27]

This circuit can be employed to simulate quantum tunneling in small systems after an overview of the final circuit has been discussed. The circuit developed in the preceding sections will be employed in the subsequent section to conduct a four-qubit simulation, and the results will be presented. The subsequent section will provide a comprehensive discussion of the necessary details for an efficient hardware execution following this simulation.

2.2.6. Four Qubit Quantum Tunneling Simulation

The results of the four qubit quantum tunneling simulation in a noiseless environment are presented in this section. By examining these results, we can gain a more comprehensive understanding of the quantum tunneling phenomena and emphasize the manner in which the components previously discussed converge to create 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, providing users with the option to select between noiseless and noisy simulation modes.

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

The setup of the experiments is summarized 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 the part regarding the Potential Energy operator in Section 2.2.2. Potential type x11x can be achieved through a filter-type gate, as suggested in ref. [17]. This is rather intuitive, since we need to apply a phase corresponding to this operator to the states that have a non-zero potential in our modeling. 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 4**. The results are presented in **Figure 5**, and for 5 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. As anticipated, the initial experiment does not demonstrate any tunneling, as there is a potential wall that prevents the particle from tunneling due to the size and width of the barrier. Figure 5a illustrates the results of the simulation

Table 1. Experiments Setup for four qubit quantum tunneling simulation.

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

across multiple time steps. We emphasize that the diagram visually illustrates the probability of the particle being at a specific position (represented on the x-axis) by varying the color intensity. When one sets up multiple wells, as in Figure 5b,c, 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, it can be seen "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⟩. 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 four 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.

2.3. Running on Hardware

After discussing the implementation of the quantum tunneling simulation, it is possible to begin considering the critical factors and adjustments required for the circuit to operate on actual hardware. In order to obtain precise results, it is necessary to consider the various complexities of the actual run, which will be elaborated upon in this section.

2.3.1. Aspects of Hardware Execution

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 tolerant to error as possible, which would mean including 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 seven-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 seven-qubit quantum computer Nairobi, Jakarta, and the 128-qubit quantum computer Osaka. Apart from IBM's superconducting quantum computers, others could have also been considered, such as IQM's trapped ions quantum computers. Certain advantages and disadvantages may be identified 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 that partake in different two-qubit gates are mapped to physical qubits that are not connected, this might translate into less efficiency, since one of them will need to be swapped next to the other one (therefore adding three two-qubit gates to the circuit that introduce noise). Also, since our circuit is most of the time not entirely composed of basis gates, but also of composed gates, 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 executing 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 provide the best option available for a given circuit, and even more for a given statevector. What we seek to imply by this is that two identical circuits, yielding the same statevector, may exhibit distinct implementations leading to variations in circuit depths which are not "solved" by the transpiler. To illustrate this, in Figure 8, in the abstract circuit section, we can observe two circuits that eventually produce the same statevector. However, we can also notice 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 two-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. The observation that choosing the hardware-aware implementation of a certain problem is essential when it comes to superconducting circuits must be considered when designing algorithms to be run on

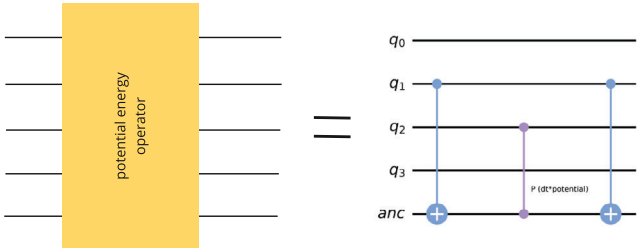
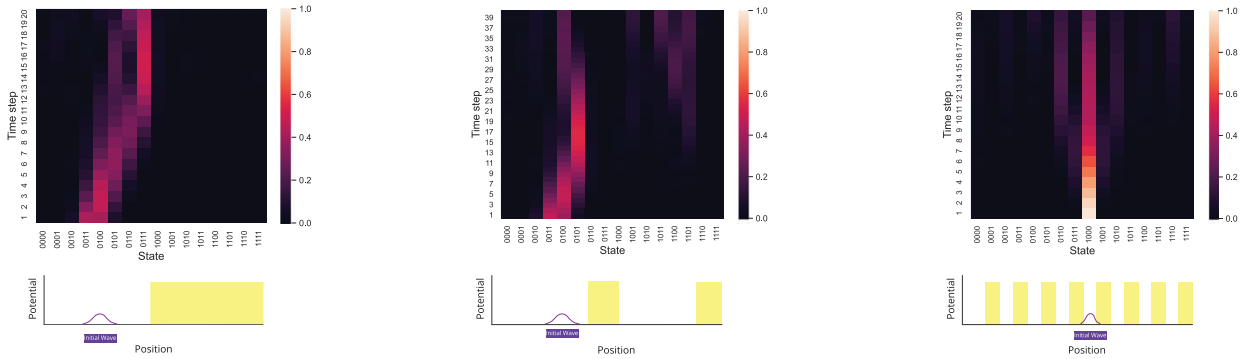


Figure 4. Implementation of x11x type potential.



(a) Simulation of wall potential. The particle starting here at $|0100\rangle$ could not penetrate the barrier.

(b) Simulation implementing two wells of potential. The particle can be observed tunneling between the two wells created.

(c) Multiple wells. The particle starting here at $|1000\rangle$ can be seen tunneling between the wells formed.

Figure 5. Each diagram represents a four qubit 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.

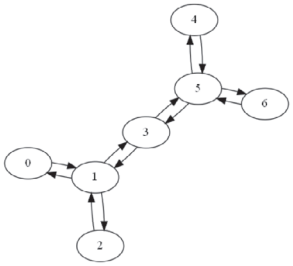
superconducting architectures. One can also notice how depth is increased by using several swap gates that were required to map the circuit to the layout, as observed in the example from Figure 7 as well. This process, responsible for adding swap gates to ensure connectivity adherence, constitutes the *routing* passes.

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 two-qubit gates regardless of which physical qubit was chosen as the control or target. However, connectivity is not symmetric on the newer chips (Eagle r3). 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.

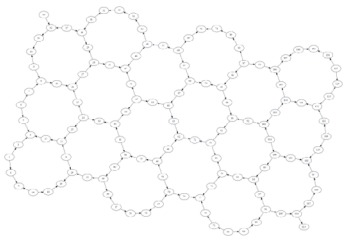
2.3.2. The Transpiler

The primary component utilized in preparing an abstract quantum circuit for running on a real quantum computer is the transpiler, as previously mentioned. This component is sometimes named compiler, depending on the programming language used, and played an important role in the optimization of the circuit analyzed in this study. Therefore, we summarize the main components, as implemented by *qiskit*, the library we chose for implementation:

1) Decomposing the custom gates into one and two-qubit gates—one step here is unfolding; for our quantum simulation circuit, this means unfolding the kinetic energy operator and the potential energy operator
2) Layout - finding the best physical qubits 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.
3) Routing—inserting swap gate in order to respect the connectivity

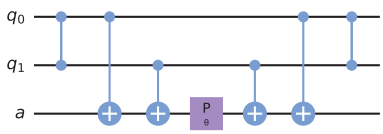


(a) Coupling map for the 7-qubit quantum computer Nairobi. It can be seen that this coupling map is symmetric.

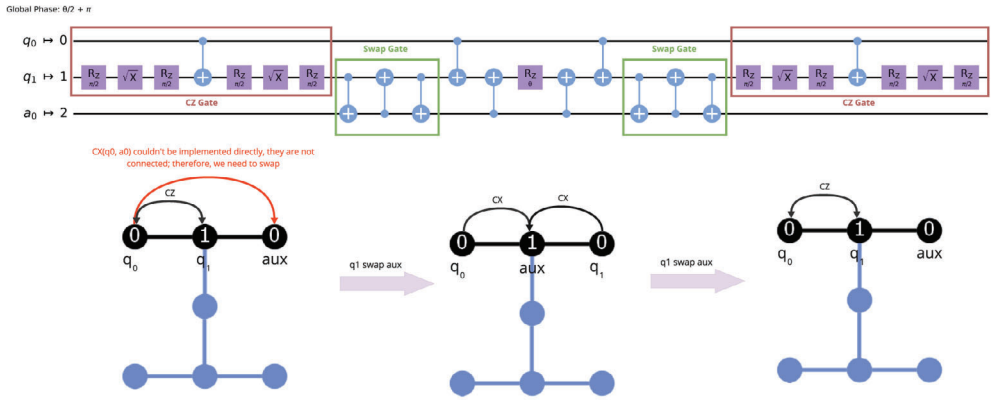


(b) Coupling map for 128-qubit quantum computer Osaka. Here, the coupling map is not symmetric.

Figure 6. Coupling maps of the IBM quantum computers used in the experiments.



(a) The diagram of the circuit containing virtual qubits. Here, the circuit is not *transpiled*. Namely, it does not adhere to specific hardware's instruction set architecture.



(b) The transpiled circuit. In this instance, the circuit is altered to be in line with the instruction set that the quantum computer's architecture supports. The mapping of the virtual qubits to the physical qubits is presented on the left-hand side of the diagram. Various aspects of this translation are emphasized, such as the routing process that is accomplished by inserting swap gates to adhere to the connectivity. The circuit diagram below illustrates the routing procedure and the chip's layout.

Figure 7. The transpilation process of a quantum circuit. Both the original circuit and the transpiled circuits are presented.

4) Translation (into basis gates—the native gates for the specific backend)
5) 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.
6) 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 (**Figure 7**). 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, would result in different final circuits of different depths. **Figure 8** presents one example that highlights this idea.

Apart from the alterations done by the transpiler in order to make the circuit adhere to the instruction set architecture of the specific hardware chosen and to optimize the circuit, another factor one might consider is the integration of error mitigation tools in the development process of the algorithm, considering NISQ era devices. Consequently, the subsequent section provides a concise overview of the function of error mitigation in quantum computation and presents a few methodologies that one might consider. Two of these were implemented in this investigation to obtain precise results.

2.3.3. Error Mitigation

In the NISQ era, the objective of various techniques is to mitigate noise rather than eliminate it. Consequently, the discipline of quantum error mitigation has emerged. Multiple techniques have been developed and they are now used in order to extract useful information in real-world applications,[28] such as the ones related to quantum chemistry.[14–16] Namely, some relevant techniques are Zero Noise Extrapolation (ZNE), Probabilistic Error Cancellation (PEC), Readout Error Mitigation (REM), Subspace expansions and Symmetry contraints[29] and Clifford data regression (CDR). Here, we will overview some of the relevant error mitigation techniques and discuss their advantages and disadvantages.

Probabilistic Error Cancellation

Probabilistic Error Cancellation (PEC) is an error mitigation technique that is based on representing the ideal operators used

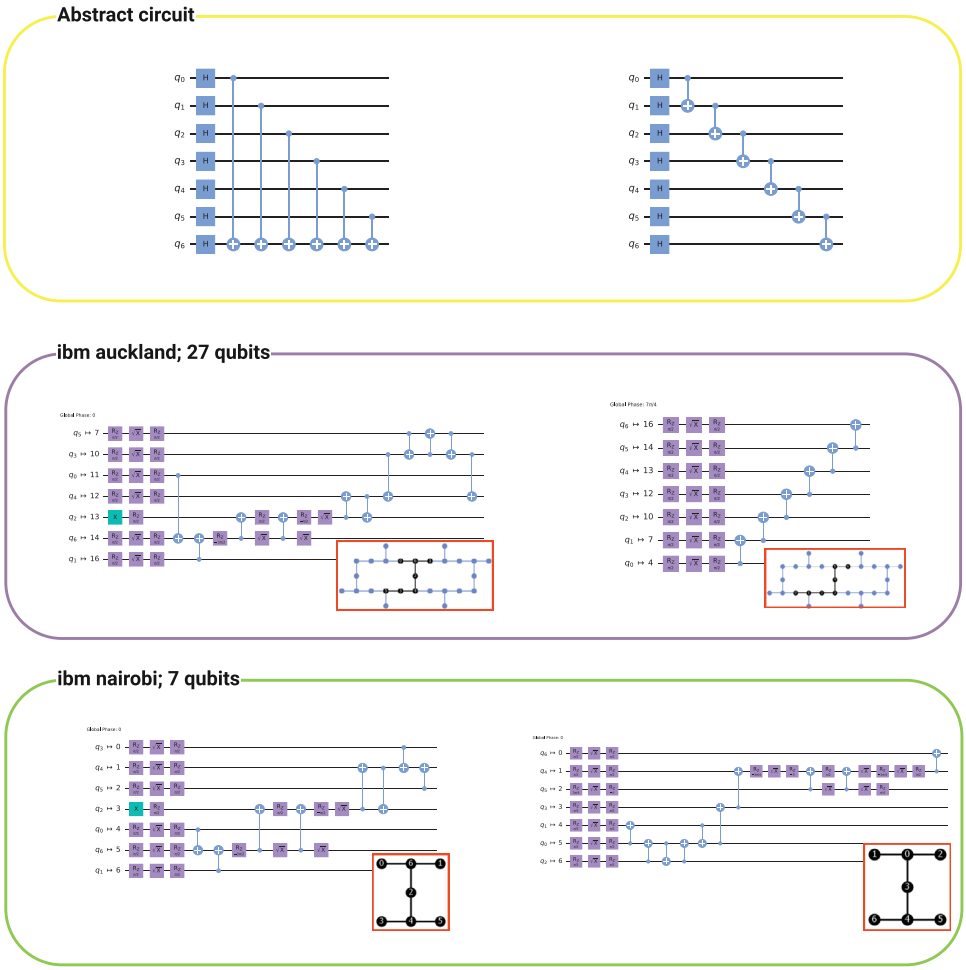


Figure 8. An example of how the depth of the circuit is influenced by the circuit design and backend. In this instance, the left and right circuits generate identical states. Various transpiled circuits are produced with varying depths, based on the quantum computer and the physical qubits employed. As implementation and hardware chosen both have an impact on the final circuits, it is necessary to implement hardware-aware design.

in the circuit by a linear combination of noisy operators.[6,30,31] Specifically, each operator $\mathcal{G}_i$ of interest to our final circuit is represented as a linear combination of noisy implementable operators $\mathcal{O}_i$, as shown in Equation 27, using the notations in ref. [32]. Here, $\eta_{i,\alpha}$ is a quasi-probability distribution representing the probabilities associated with respect to the index α. Therefore, it is assumed that the set of implementable operations $\mathcal{O}_i$ is sufficiently large to represent the ideal operators $\mathcal{G}_i$.[30]

$$\mathcal{G}_i = \sum_{\alpha} \eta_{i,\alpha} \mathcal{O}_{i,\alpha} \tag{27}$$

The aim of PEC is therefore estimating the expectation value of some observable, A, that would result in a noiseless environment. Rewriting the ideal expectation value ($\langle A \rangle_{ideal} = tr[A\mathcal{U}(\rho_0)]$, where $\mathcal{U}$ is the operator that acts on the density matrix and is composed of the ideal operators $\mathcal{G}$) in terms of noisy expectation results, it can be observed that once the quasi-probability distribution is learned, the ideal expectation value can be extracted.[30]

The formula for the ideal expectation is illustrated in Equation (28).

$$\langle A \rangle_{ideal} = \sum_{\vec{\alpha}} \eta_{\vec{\alpha}} \langle A_{\vec{\alpha}} \rangle_{noisy} \tag{28}$$

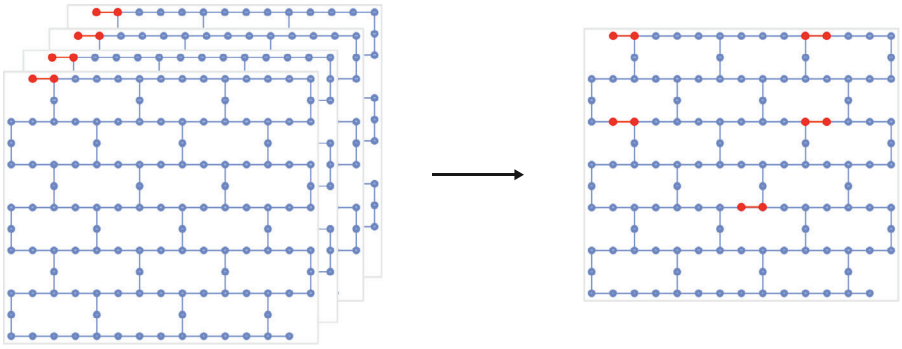
where

$$\eta_{\vec{\alpha}} = \eta_{t,\alpha_t} \dots \eta_{2,\alpha_2} \eta_{1,\alpha_1}$$

$$\langle A_{\vec{\alpha}} \rangle_{noisy} = tr[A\Phi_{\vec{\alpha}}(\rho_0)]$$

$$\Phi_{\vec{\alpha}} = \mathcal{O}_{t,\alpha_t} \circ \dots \circ \mathcal{O}_{2,\alpha_2} \circ \mathcal{O}_{1,\alpha_1}$$

However, since this sum grows exponentially with respect to the number of gates, it becomes unfeasible to compute all the corresponding expectations.[30] Therefore, one can choose to use the Monte Carlo Approximation. That is, by sampling individual gates one can effectively sample from the global quasi-probability distribution. Using this sampling, one is able to reduce the overhead by measuring only a subset of all the terms needed for approximating the noiseless value of the final result. In order to acquire the necessary information regarding



Classical approach

Constructing each circuit separetly and running on backend

Multiprogramming

Placing on the same chip the circuits which we want to run.

Figure 9. Multiprogamming allows for multiple circuits to run on the same chip.

the noisy operators, in one implementation of PEC one would have to perform gate set tomography to gain the knowledge about $\mathcal{O}$. This would be done if one would need a comprehensive representation of the operators with respect to the real noise model present in the quantum computer. Of course, simpler noise models can be considered as well along with vendor's calibration data in order to extract the representation of the ideal operators in terms of the noisy implementable ones with less overhead.

Clifford Data Regression

Clifford Data Regression (CDR) is a technique that uses data obtained from classically simulable circuits to train and therefore extract the noiseless expectation value for a given circuit. It was proposed in 2020, in ref. [33]. Simply put, a training set $\{V_i\}_{i=1}$ is constructed where each circuit is similar to the one of interest, U. A simple way to construct these circuits of interest is by replacing a subset of the gates in our original circuit, U, with Clifford gates that are close in distance to the original gates, as proposed in the initial article.[33] The need for using Clifford gates comes from the requirement that the circuits must be efficiently simulated by a classic computer (and circuits composed of Clifford gates fall into this category, due to the Gottesman–Knill theorem). After the circuits are obtained, the expectation values of a given operator X are evaluated using a classical computer. The noisy version of this expectation value is also obtained from running the circuits constructed on a quantum computer. These two sets will be merged and represent the training data set. Following the notation in ref. [34], we will denote the space of noiseless and noisy expectation values with $\epsilon_{noiseless}$ and ϵ_{noisy} respectively. The dataset obtained is then used to train a function $f : \epsilon_{noisy} \rightarrow \epsilon_{noiseless}$ that can be used to extract the approximated noiseless value for the given circuit and observable. This technique can be integrated with the development process by using mitiq.[32]

Readout Error Mitigation

Readout Error Mitigation is an umbrella term used for multiple approaches to mitigating errors, most commonly by means of a confusion matrix through applying its inverse to the outcome probability distribution, as to extract the noiseless distribution,[7,8] first proposed in ref. [35]. 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

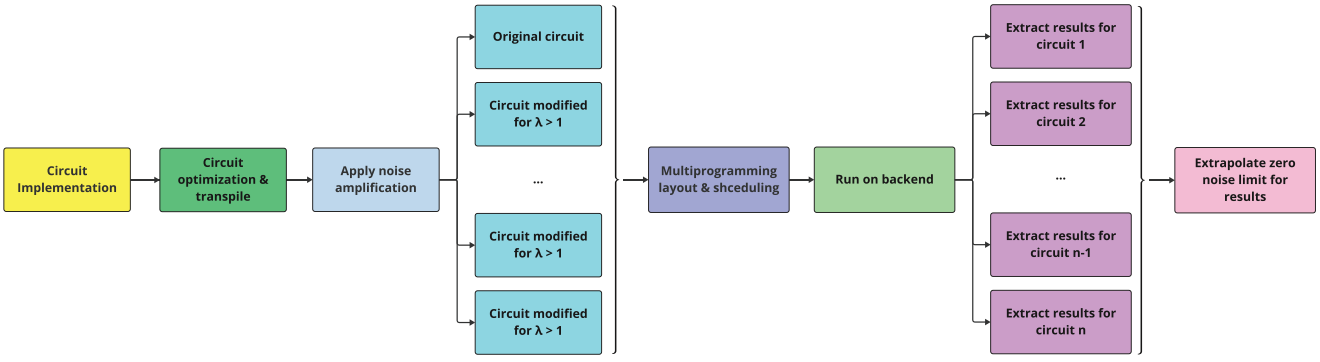


Figure 10. Final workflow of the two-qubit quantum tunneling simulation. After the circuit is implemented and optimized for a given hardware, multiple equivalent circuits with different depths are constructed (circuits modified in order to increase the noise level in the circuit). These circuits are then mapped on the same quantum chip (multiprogramming the chip) and run. The results are then extracted and the zero noise value extrapolated.

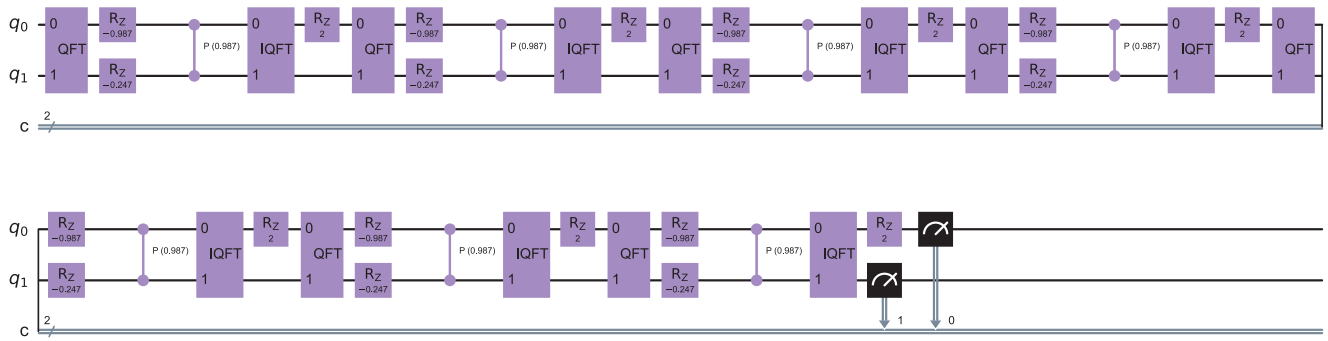


Figure 11. The circuit diagram for the two-qubit simulation. The troterized implementation of the Hamiltonian can be observed across seven timesteps. Namely, the implementation of kinetic energy operator (between the QFTs) and the potential energy operator (after each Inverse QFT) can be seen across multiple timesteps.

correlated results are considered, for n qubits, the matrix will be $2^n \times 2^n$. However, if the noise is considered local, we will have n matrices 2×2, one for each qubit. This is an important detail to take into account especially when analyzing scalability, since for a larger number of qubits, constructing and working with the whole $2^n \times 2^n$ matrix can become cumbersome. Therefore, multiple optimizations have been proposed to ensure the scalability of this technique in contexts with multiple qubits.[36,37] Under the assumption that the dominant form of errors are uncorrelated errors (or that there is minimal measurement crosstalk), both[36] and[37] use the n 2×2 calibration matrices in order to characterize the whole $2^n \times 2^n$ matrix. Moreover, matrix-free iterative techniques can also be implemented and show convergence in $\mathcal{O}(1)$ steps.[36] Ideally, regarding the type of error that is considered, these 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, when available, is calculated (the Moore–Penrose pseudoinverse) and then applied to the extracted probability distribution. When the full matrix is not available due to optimization concerns, different techniques are used in order to extract the noiseless results.[36,37]

Zero Noise Extrapolation

First introduced in 2017, Zero Noise Extrapolation is an error mitigation technique based on trying to extrapolate the noiseless result of a quantum circuit by analyzing the circuit in different noise levels.[5,6,38] 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,[32] 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:

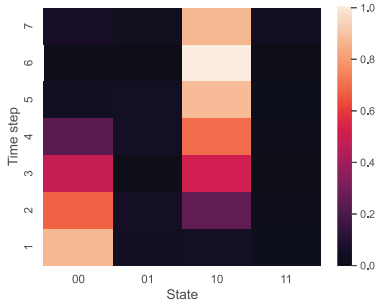
$$\langle E(\lambda)\rangle = Tr[\rho(\lambda\tau)A] \tag{29}$$

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[32] 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.[5,39] We used local folding for our circuits. Pulse level gate implementations can also be used in order to control the noise introduced.[40,41]

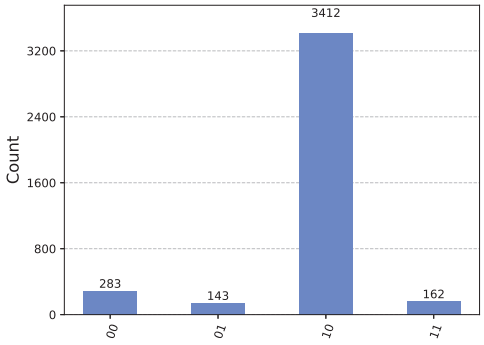
Nevertheless, a drawback of employing the ZNE technique is the additional computational burden caused by operating many circuits to extract the pertinent data. An inefficient approach would include executing these circuits one after another and then estimating the expected value at zero noise limit. Nevertheless, when running small circuits on chips with larger quantum volumes (such as the 128 qubits quantum computers offered by IBM), the quantum chip exhibits a low utilization rate for each run. Hence, it may be desirable to effectively execute various circuits while simultaneously utilizing the same quantum device. This work employed the concept of multiprogramming to address the issue of underutilization of hardware in small simulations. Hence, in the subsequent part, we delineate the specific components and considerations involved in incorporating this technique into the development workflow.

Combining Error Mitigation Techniques

Since each error mitigation technique has its own unique advantages, different workflows that incorporate more than one error mitigation strategy have been developed. Such techniques usually have the disadvantage of a bigger overhead, either due to sampling or specific requirements, but nonetheless prove to be useful in many applications. One example that conceptually



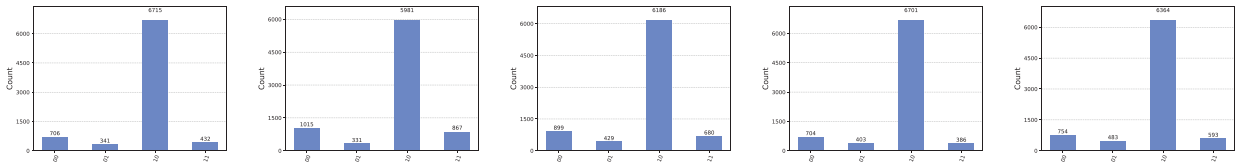
(a) Plot of the 7 timesteps. The particle starts at location $|00\rangle$ and then tunnels to location $|10\rangle$ by timestep 7.



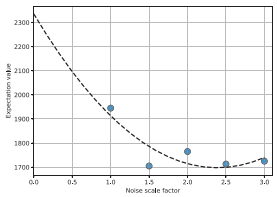
(b) Ideal distribution for 4000 shots. Quantum tunneling is visible from well $|00\rangle$ to $|10\rangle$.

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

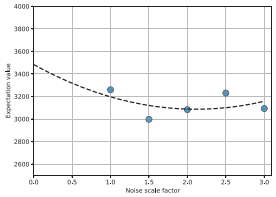
unifies ZNE and CDR is variable-noise Clifford Data Regression (vnCDR), proposed in ref. [42]. Here, the modification added to the CDR technique is the use of noise amplification after the initial alterations done to the initial circuit. We remind that the modifications introduced by CDR are done by replacing subsets of gates with Clifford gates, ensuring that the new circuits can be efficiently simulated on a classical computer. After choosing the noise levels, in the step of building the training set, these noise levels are used to alter the CDR modified circuits and to obtain the final data set, containing circuits similar to the initial circuit modified by both clifford replacements and noise amplification techniques. After the expectation values are extracted, these will be used for training the function. The efficiency of this method has been tested on multiple applications and yields results up to 33x more accurate then the unmitigated values, better by a factor of 2 compared to the constituent techniques used independently. Noise-extended Probabilistic Error Cancellation (NEPEC) proposed in ref. [30] is another example of a technique that incorporates multiple strategies. Here, both PEC and ZNE are employed in order to avoid the usage of gate set tomography and to allow the user to obtain partially mitigated results at a lower sampling cost. Using NEPEC, the expectation value is estimated from a general combination of circuits present at different noise levels (also, the noise of each gate in the circuit can be scaled differently). Another example of a technique that incorporates multiple error mitigation strategies is Unified Technique for Error Mitigation with Data (UNITED) proposed in ref. [34]. Here, Zero-noise Extrapolation, Clifford-data regression and Virtual Distillation are employed together in order to construct a framework that outperforms each method. The sampling overhead is also to be considered here since more powerful methods usually require more shots for optimal performance. Also, in order to use UNITED, training is also necessary, as one might expect from CDR-based strategies. The question then becomes what are the criteria one has to consider when choosing the error mitigation strategy for a given problem. First, a set of specifications for the best error mitigation techniques can be taken into consideration.[43] Once the requirements pertinent to the method are fulfilled, consideration of application-specific variables can begin. Moreover, in real applications one factor that must be taken into account is the overhead of computation. For example, in our problem, we might consider a two-qubit simulation. Considering the quantum volume of current hardware, the chip is mostly unused. Therefore, multiple simulations can be efficiently run simultaneously. This means that if one were to choose for example ZNE, multiple circuits with different noise levels can be executed with little overhead for the time the hardware is used (since they are run simultaneously, depending on scheduling, the time the chip is used will be the time necessary for the circuit with the most depth to run). There is, of course, an upper limit to the number of circuits that can be run on the same chip, considering the limited number of qubits and the gradual influence of crosstalk between qubits. Also, the desired accuracy is also to be considered when estimating the overhead needed. For training based solutions, one has to construct more circuits to reach desired accuracy. Also, when it comes to CDR-based techniques, one has to consider that the circuits will be run on a classical computer as well, so the final data set might require more resources to create. For our problem, we chose to start by applying the ZNE strategy, a well known technique that proved efficient for a lot of NISQ era applications, and optimize this solution in order to extract the best results. Following the criteria devised in ref. [43], we can analyze part of the benefits and drawbacks of the chosen technique. There are seven requirements proposed for an ideal error mitigation method: accurate result recovery, depth independence, the versatility of the error model, practically realizable, no necessity of additional hardware, gate-set independence and no prior knowledge of the output of the circuit being used.[43] It is clear that the majority of these conditions are satisfied. Namely, ZNE helped us recover accurate results and it is, in construction, independent of circuit depth (however, the circuit depth becomes relevant when hardware run is considered). Moreover, the method is practically realizable (indeed, the same hardware is used for running the original circuit as well as the noise amplified circuits). Also, no additional hardware is used to perform ZNE. Criteria 6 and 7 from the mentioned resource are also fulfilled, as the method is independent of the gate set and does not rely on any prior knowledge of the output for extrapolation. When considering the error model, while ZNE does in theory help mitigate all types of errors, it can underperform in cases in which the noise is not well understood



(a) Results from backend run of each modified circuit. Quantum tunneling can be observed by the presence of the particle in the well represented by the state $|10\rangle$.



(b) Inference on counts from noisy simulators.



(c) Inference on noisy results from backend.

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

and varies significantly over time. In this situations, the noise amplification along with the extrapolation can perform poorly and give vastly different results depending on the technique chosen. When these extreme cases are considered, one can choose a more targeted technique, such as Folding-Free ZNE which proves efficient in cases of extreme noise (in which noise amplification proves inefficient) and especially in cases where depolarizing and decoherence noise is present.[44] Since the noise present in IBM quantum computers is fairly stable and the size of our circuit allowed efficient noise amplification, ZNE proved to be efficient for our application. Also, the overhead of this technique was lower compared to other methods and allowed us to couple this strategy with an efficient use of the quantum chip, which will be discussed in the next session. By applying this technique, we were able to obtain results 10x more accurate than the unmitigated results, and the final error obtained is satisfactory. Increasing this accuracy would have come with a larger overhead imposed by other techniques. For more complex simulations, one could take into consideration more complex error mitigation strategies as well.

2.3.4. 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[45–48] is a technique used for tackling the problem of hardware under-utilization in the NISQ era. Because the size of the circuit is currently restricted by hardware capabilities, one might end up using only a small percentage of the qubits available in one run. For example, when we use the two-qubit experiment and we want to run it on 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**.

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.[48] Layout strategies have been developed to therefore increase the usage of a chip,[47] along with scheduling algorithms that follow the same aim of reducing crosstalk.[45,46] 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 (30).

$$\rho_Q \equiv Tr_A[\rho_{QA}] \tag{30}$$

To conclude this section, we outlined the theoretical context needed for understanding and implementing a quantum tunneling simulation as well as thoroughly explained the implementation considerations and the intricacies involved when running on hardware. Specifically, we started from the Schrödinger equation and the potential profile needed for observing the

Table 2. Results on Noisy simulator and on Real Backend. The notations are shown below. The effectiveness of the error mitigation approach can be assessed in both cases by examining the reduced error.

Type	T	T_{run}	E_1	T^{em}_{run}	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

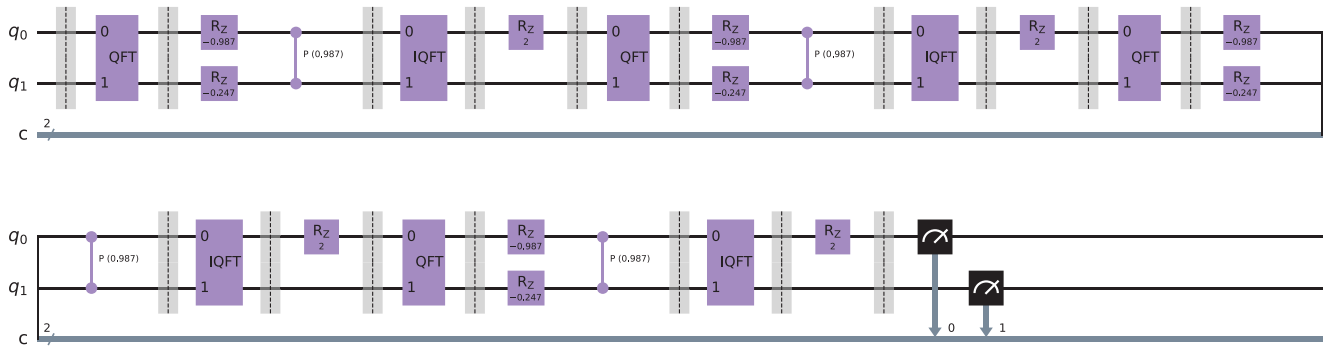


Figure 14. The diagram of the circuit used in this experiment.

phenomenon of quantum tunneling. Subsequently, Section 2.2 delved into the adaptations required for implementing the theoretical framework in the context of quantum computing. A range of possibilities for implementing the kinetic and potential energy operators were examined, and the benefits and drawbacks were carefully assessed. A four-qubit quantum simulation in a noiseless environment was also performed and the results were interpreted, showing the presence of the phenomenon studied. The last subsection detailed the complexities that must be taken into account when running the abstract circuit on physical hardware. It discussed the significance and function of the transpiler, the incorporation of error mitigation techniques, and the efficient use of the quantum chip.

The following section will describe the results of three pertinent experiments, showcasing various features of the approaches presented for a quantum tunneling simulation, after having covered the necessary algorithms and designed the workflow.

3. Results

The purpose of this section is to provide an overview of the outcomes derived from numerous iterations of Quantum Tunneling simulations. In contrast to the experiment conducted in Section 2.2.6, the initial two experiments discussed here will mostly concentrate on hardware performance enhancements and error mitigation strategies. Section 3.1 will cover a comprehensive workflow that includes implementing the circuit, preparing it for hardware, using error mitigation approaches to improve accuracy, and employing multiprogramming to efficiently utilize a quantum chip. The simulation's great accuracy is also mentioned. In Section 3.2, we will examine two additional experiments that we consider detrimental to grasping the adaptability and effectiveness of the tools used in quantum tunneling simulations.

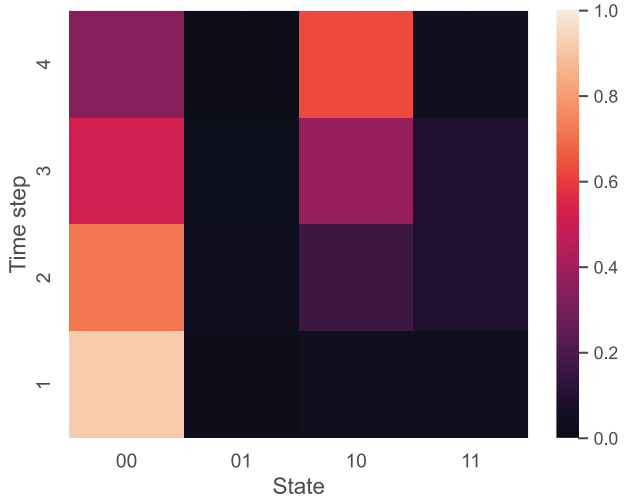


Figure 15. Plot of the probabilities of the four timesteps. Quantum tunneling is visible from well $|00\rangle$ to $|10\rangle$.

3.1. End to End Run

This experiment aims to simulate quantum tunneling on a two-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**.

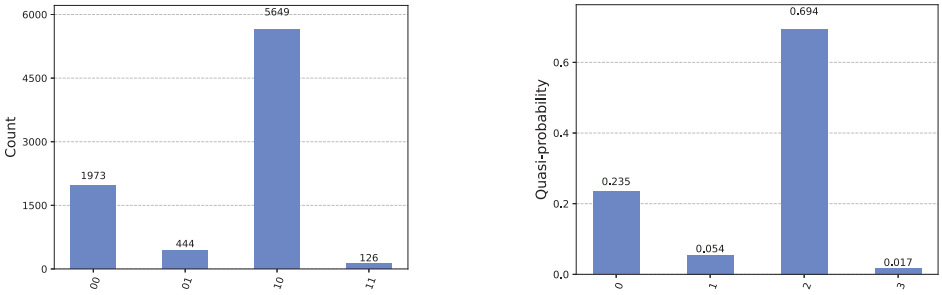
Methodology

For implementation, we used *qiskit* along with the *tket* compiler. For error mitigation techniques, we used mitiq.[32] For this experiment, the space was discretized using two qubits. We chose seven timesteps and Δt of 0.1. As hardware, the 128-qubit ibm_osaka was used.

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 part Potential Energy Operator from section 2.2.2):

$$\hat{V} = e^{-iv\sigma_z^0 \Delta t} = I \otimes e^{-iv\sigma_z \Delta t} \tag{31}$$



(a) Count distribution after the 4 timesteps. (b) Sampler output of the simulation.

Figure 16. Experimental results for the two-qubit experiment in noiseless environment.

Here, since we use *qiskit* for implementation, we opted for *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 $|11\rangle$.

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.

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. Regarding the layout technique for multiprogramming, we choose to allocate a buffer of at least one physical qubit, based on actual findings from previous investigations.[48] 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.

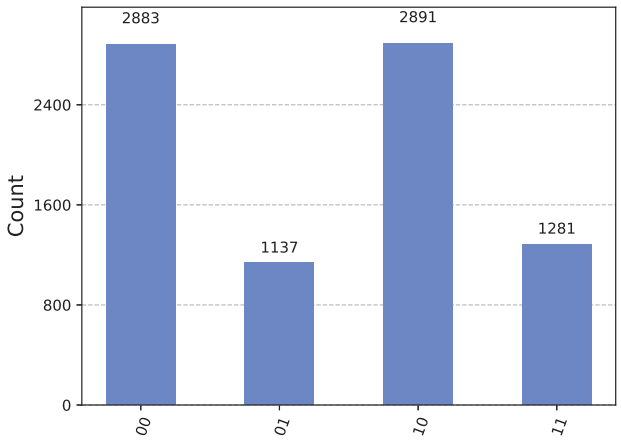


Figure 17. Results of backend run of the circuit. While tunneling can be observed between the two wells, $|00\rangle$ and $|10\rangle$, caused by the barriers $|01\rangle$ and $|11\rangle$, noise can also be observed in the erroneous counts on these states.

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**.

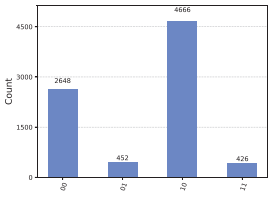
Where the following notations have been used:

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 $|T - T_{run}^{em}|$

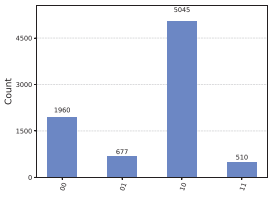
Thus, it is clear that error mitigation resulted in a substantial enhancement compared to the uncontrolled run, as we can see that the estimated expectation value of our observable is only 0.006 away from the desired value. The importance of error prevention was evident in both simulated and real scenarios. Within the simulated scenarios, the error rate decreased significantly from 0.224 to 0.066, representing a reduction of three-quarters compared to the initial error. During the actual test, the error reduced from 0.062 to 0.006, demonstrating another significant improvement. One can also acknowledge the precision of the acquired transmission probability. In addition, by employing the multiprogramming paradigm, the chip was optimally utilized. Rather than utilizing only 2 out of the 128 qubits, the same device was used for five consecutive tests.

3.2. Extended Experimental Examination

The setup described in this article showcases its versatility, and we are confident that conducting further experiments will deepen our grasp of result interpretation and the different components of the indicated procedure. Hence, we carried out two further experiments, which will be elucidated in this part. The initial

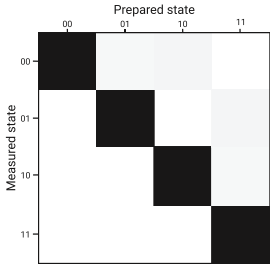


(a) Results from the backend run on the Nairobi quantum computer.

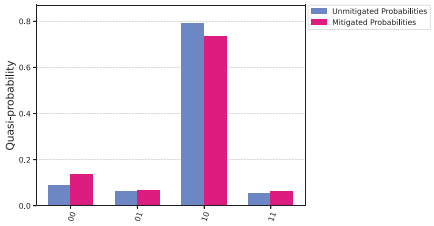


(b) Results from the backend run on the Osaka quantum computer.

Figure 18. Experimental results for two-qubit simulation without error mitigation.



(a) Confusion matrix incorporating the error observed in the systems. While the state prepared is almost always the same as the one measured (observed by the probabilities across the main diagonal), errors in measurement can also be observed.



(b) Probabilities compared of the mitigated (in blue) and unmitigated (in pink) runs.

Figure 19. Experimental results for two-qubit simulation with REM.

experiment involves simulating a system with two qubits while employing Readout Error Mitigation. The subsequent experiment entails simulating a system with six qubits in a noise-free setting, utilizing the Hadamard Test to enhance comprehension of the wavefunction during quantum tunneling. First, a diagram of probabilities step by step will be supplied for each experiment reported. A statevector simulator is used to precisely determine the probability at each time step. Once the statevector from each timestep has been obtained, the probabilities are retrieved as well.

3.2.1. Two-Qubit Simulation Using REM

The objective of the first experiment is to simulate quantum tunneling on a two-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:

1) simulates quantum tunneling on a two-qubit system
2) observe tunneling between the two wells
3) optimize and run on hardware
4) analyze the results

Theoretical Foundation: Relevant Theoretical Concepts for the Experiment

For this example, implementation of the QFT, the potential and kinetic energy operators were needed. We employed the

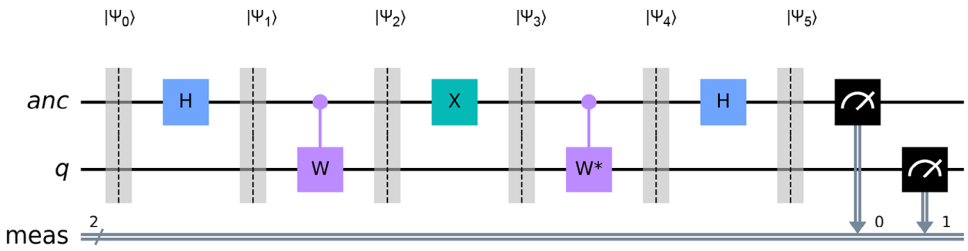


Figure 20. Hadamard test implementation.

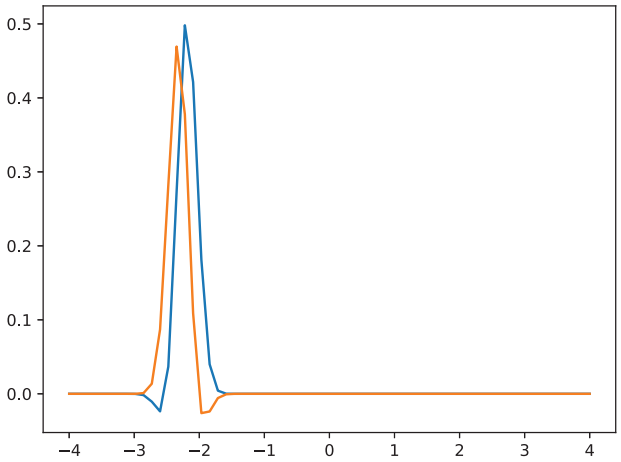


Figure 21. Plot of the initial wavepacket used to model the initial state of the system. Both imaginary and real parts are plotted. Based on the discretization of this function, the circuit will be initialized in this state.

momentum operator implementation without using an additional qubit in order to avoid increasing the circuit's depth with swap gates necessary to replicate a three-qubit all-to-all connectivity.

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 small-size experiment. 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.

Discussion and Implications of Experimental Results

First, 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 part Potential Energy Operator from section 2.2.2):

$$\hat{V} = I \otimes e^{-iv\sigma_z^0 \Delta t} \tag{32}$$

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 $|11\rangle$.

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.

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 $|10\rangle$.

First, we tried running our circuit on the *qiskit* backend without further intervention in the transpilation process. The hardware used in this experiment was the five-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 17**.

We can now notice the difference between simulations and running on hardware in Figures 16 and 17. 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 observed in the results, for instance, as erroneous counts on states $|01\rangle$ and $|11\rangle$. Errors can have multiple causes, 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 less prone to errors. As presented in section 2.3.2, 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 further optimize the circuit. Moreover, for the layouting technique, we used the *mapomatic* library and the routing from *qiskit*. By applying this optimization, the result presented in **Figure 18** is obtained.

Upon applying the same optimization for another backend, for example, for the 127-qubit Osaka quantum computer, the result in Figure 18b is obtained. It is clear from Figure 18 that the results have improved after finetuning the compilation process.

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 two-qubit state. Figure 19b shows 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.

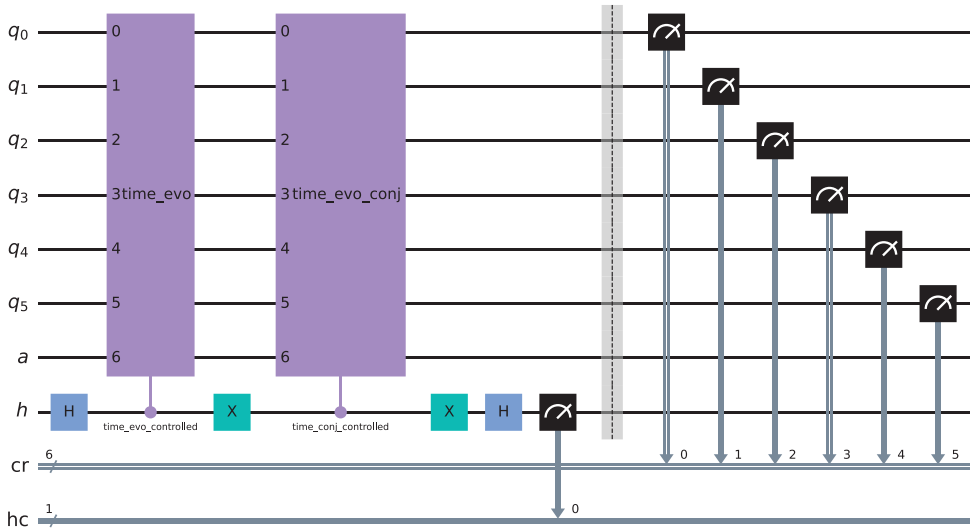


Figure 22. Implementation of Hadamard test. The working register q is of size 6. These are the qubits for which the operators were used. The auxiliary register contains one qubit a, which is used for the implementation of the momentum operator, and a qubit h needed for the Hadamard test - depending on 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 six qubits. hc is of size 1. This holds the value of the auxiliary qubit h. Based on this value, we know whether we measured the imaginary or real part of the wavefunction.

3.2.2. Six-Qubit Quantum Tunneling Simulation using the Hadamard Test

The objective of the experiment is to simulate quantum tunneling in a six-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:

1) to implement the simulation circuit in a bigger six-qubit system
2) to extract probability counts, and observe tunneling
3) to extract real and imaginary part counts, and observe how they evolve

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.

Hadamard Test

The Hadamard test is used when 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:

$$|\psi\rangle = \sum_i \alpha_i |k_i\rangle \tag{33}$$

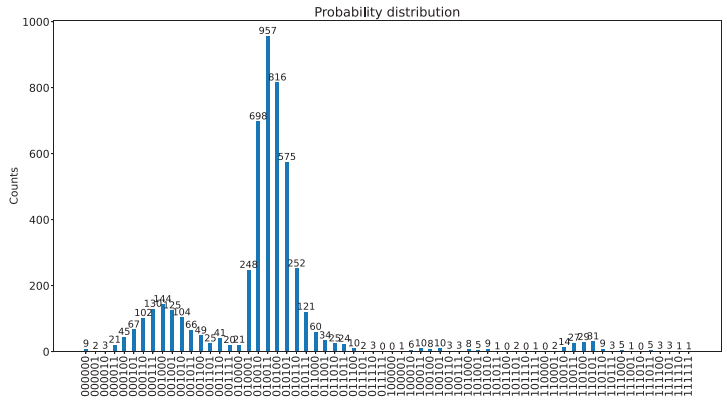
where α_i is, of course, a complex number and $|k\rangle$ represent the base states. Let W be the operator that takes the state from $|0\rangle^{\otimes n}$ to $|\psi\rangle$, that being $|\psi\rangle = W|0\rangle^{\otimes n}$. In order to extract the distribution of the real and imaginary parts, we 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.

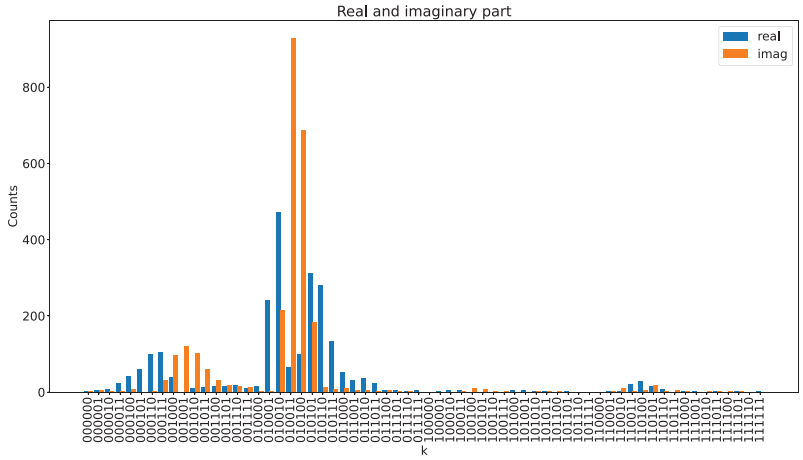
Schematically, the implementation is represented in **Figure 20**

Experiment Setup: Hardware, Software, and Key Parameters

Here we used *qiskit* for implementation. Moreover, we did not need the *tket* compiler for hardware optimisation, since the depth of the circuit would not allow for current free available quantum computers to run. Since the space here allows it, we will use the Gaussian function as the initial wavefunction. Since the depth of



(a) Counts probability distribution. This is the outcome that can be readily achieved without utilizing the Hadamard test.



(b) Real and imaginary count distributions. By utilizing the Hadamard test, it is possible to access these distributions individually, rather than being limited to simply the final probability distribution.

Figure 23. Experimental results for Hadamard test experiment.

this circuit is large, we reached no further than simulating it on qasm_simulator. The initial wavepacket is represented as:

$$|\psi\rangle = \frac{1}{\sqrt{2\pi}0.8} e^{-\frac{1}{2}\left(\frac{x+2.25}{0.032}\right)^2} e^{-300ix} \tag{34}$$

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. The 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**.

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**.

Analysis of Experimental Results

The simulation was conducted using the qasm_simulator. **Figure 23** displays the probability distribution as well as the distribution of imaginary and real counts for the evolved state. The potential barrier is placed at positions xxx11x (the implementation of this potential barrier is discussed in Section 2.2.6), 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.

4. Conclusion

This article presents the main steps required for translating the quantum mechanical challenge of simulating the Schrödinger equation into one that is compatible with a quantum computing

framework. That is, we began by examining the theoretical implications of simulating quantum tunneling and then discussed various approaches for implementing the operators. Furthermore, our focus was on identifying the changes needed to ensure the efficient and accurate running of the abstract circuit on an actual backend. In our discussion, we explored the details of the transpilation process and the continued need for hardware-aware design in the NISQ era, specifically in relation to superconducting designs. Various compilers were used to optimize the circuit for the selected backend, and they demonstrated effective optimization capabilities. Additionally, error mitigation techniques such as Zero Noise Extrapolation (ZNE) and Readout Error Mitigation (REM) have been employed to improve the accuracy of the simulation outcomes. To tackle the issue of underutilization in today's chip hardware, multiprogramming was employed to efficiently execute the circuit. This was done in conjunction with noise-amplified circuits and the aforementioned error mitigation strategies. Hadamard tests for six-qubit systems were also simulated to enrich the understanding of the topic. The outcomes of our research have various and complex implications. By effectively simulating quantum tunneling in a comprehensive workflow, we obtain accurate outcomes in a 2-qubit simulation. We suggest that this flexible methodology provides substantial value in various simulation scenarios, thereby delivering universal advantages to researchers.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

Not applicable.

Keywords

compiler/transpiler, error mitigation, multiprogramming, quantum simulation

Received: April 10, 2024
Revised: August 26, 2024
Published online: September 22, 2024

[1] R. P. Feynman, *Int. J. Theoret. Phys.* **1982**, *21*, 467.
[2] S. Lloyd, *Science* **1996**, *273*, 1073.
[3] J. Preskill, *Quantum* **2018**, *2*, 79.
[4] Y. Liu, Z. Li, A. Robertson, X. Fu, S. L. Song, *IEEE Trans. Quantum Eng.* **2023**, *4*, 1.
[5] T. Giurgica-Tiron, Y. Hindy, R. LaRose, A. Mari, W. J. Zeng, in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, **2020**.
[6] K. Temme, S. Bravyi, J. M. Gambetta, *Phys. Rev. Lett.* **2017**, *119*, 18.
[7] M. Beisel, J. Barzen, F. Leymann, F. Truger, B. Weder, V. Yussupov, *Electronics* **2022**, *11*, 19.
[8] B. Yang, R. Raymond, S. Uno, *Phys. Rev. A* **2022**, *106*, 1.
[9] H. Liao, D. S. Wang, I. Sitdikov, C. Salcedo, A. Seif, Z. K. Minev, Machine Learn. Pract. Quantum Error Mitigat., **2023**.
[10] J. I. Cirac, P. Zoller, *Nat. Phys.* **2012**, *8*, 264.
[11] A. J. Daley, I. Bloch, C. Kokail, S. Flannigan, N. Pearson, M. Troyer, P. Zoller, *Nature* **2022**, *607*, 667.
[12] Z. Zhu, S. Yu, D. Johnstone, L. Sanchez-Palencia, *Phys. Rev. A* **2024**, *109*, 013314.
[13] S. Greenaway, A. Smith, F. Mintert, D. Malz, *Quantum* **2024**, *8*, 1263.
[14] M. Kühn, S. Zanker, P. Deglmann, M. Marthaler, H. Weiß, *J. Chem. Theory Comput.* **2019**, *15*, 4764.
[15] J. Leppákangas, N. Vogt, K. R. Fratus, K. Bark, J. A. Vaitkus, P. Stadler, J.-M. Reiner, S. Zanker, M. Marthaler, *Phys. Rev. A* **2023**, *108*, 6.
[16] A. J. McCaskey, Z. P. Parks, J. Jakowski, S. V. Moore, T. D. Morris, T. S. Humble, R. C. Pooser, *npj Quantum Inf.* **2019**, *5*, 1.
[17] M. Abouelela, Bachelor's Thesis is Physics, The American University in Cairo, **2020**.
[18] N. N. Hegade, N. L. Kortikar, B. Das, B. K. Behera, P. K. Panigrahi, *Experimental Demonstration of Quantum Tunneling in IBM Quantum Computer*, **2021**.
[19] A. T. Sornborger, *Sci. Rep.* **2012**, *2*, 1.
[20] D. J. Griffiths, D. F. Schroeter, *Introduction to Quantum Mechanics*, 3rd ed., Cambridge University Press, **2018**.
[21] J. Tersoff, D. R. Hamann, *Phys. Rev. B* **1985**, *31*, 805.
[22] H.-L. Huang, D. Wu, D. Fan, X. Zhu, *Sci. China Inform. Sciences* **2020**, *63*, 8.
[23] D. Loss, D. P. DiVincenzo, *Phys. Rev. A* **1998**, *57*, 120.
[24] I. Dhand, B. C. Sanders, *J. Phys. A: Math. Theor.* **2014**, *47*, 265206.
[25] N. Hatano, M. Suzuki, *Finding Exponential Product Formulas of Higher Orders*, Springer Berlin Heidelberg, **2005**, pp. 37–68.
[26] S. Shokri, S. Rafibakhsh, R. Pooshgan, R. Faeghi, *Europ. Physic. J. Plus* **2021**, *136*, 7.
[27] M. A. Nielsen, I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press, **2011**.
[28] S. Endo, S. C. Benjamin, Y. Li, *Phys. Rev. X* **2018**, *8*, 3.
[29] Z. Cai, R. Babbush, S. C. Benjamin, S. Endo, W. J. Huggins, Y. Li, J. R McClean, T. E. O'Brien, *Rev. Mod. Phys.* **2023**, *95*, 4.
[30] A. Mari, N. Shammah, W. J. Zeng, *Phys. Rev. A* **2021**, *104*, 5.
[31] W. Shi, R. Malaney, in *GLOBECOM 2023-2023 IEEE Global Communications Conference*, IEEE, **2023**.
[32] R. LaRose, A. Mari, S. Kaiser, P. J. Karalekas, A. A. Alves, P. Czarnik, M. El Mandouh, M. H. Gordon, Y. Hindy, A. Robertson, P. Thakre, M. Wahl, D. Samuel, R. Mistri, M. Tremblay, N. Gardner, N. T. Stemen, N. Shammah, W. J. Zeng, *Quantum* **2022**, *6*, 774.
[33] P. Czarnik, A. Arrasmith, P. J. Coles, L. Cincio, *Quantum* **2021**, *5*, 592.
[34] D. Bultrini, M. H. Gordon, P. Czarnik, A. Arrasmith, M. Cerezo, P. J. Coles, L. Cincio, *Quantum* **2023**, *7*, 1034.
[35] F. B. Maciejewski, Z. Zimborás, M. Oszmaniec, *Quantum* **2020**, *4*, 257.
[36] P. D. Nation, H. Kang, N. Sundaresan, J. M. Gambetta, *PRX Quantum* **2021**, *2*, 040326.
[37] B. Pokharel, S. Srinivasan, G. Quiroz, B. Boots, *Phys. Rev. Res.* **2024**, *6*, 013187.
[38] Y. Li, S. C. Benjamin, *Phys. Rev. X* **2017**, *7*, 021050.
[39] R. Majumdar, P. Rivero, F. Metz, A. Hasan, D. S. Wang, *arXiv:2307.05203* **2023**.
[40] I. Henao, J. Santos, R. Uzdin, *npj Quantum Inf.* **2023**, *9*.
[41] J. W. O. Garmon, R. C. Pooser, E. F. Dumitrescu, *Phys. Rev. A* **2020**, *101*, 042308.
[42] A. Lowe, M. H. Gordon, P. Czarnik, A. Arrasmith, P. J. Coles, L. Cincio, *Phys. Rev. Res.* **2021**, *3*, 3.
[43] M. S. Jattana, F. Jin, H. De Raedt, K. Michielsen, *Quantum Inf. Process.* **2020**, *19*, 11.

[44] H. P. Patil, P. Li, J. Liu, H. Zhou, in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, IEEE, **2023**.
[45] P. Das, S. S. Tannu, P. J. Nair, M. Qureshi, in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '52. Association for Computing Machinery, New York, NY, USA, ISBN 9781450369381, **2019**, pp. 291–303.
[46] P. Murali, D. C. Mckay, M. Martonosi, A. Javadi-Abhari, in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20. ACM, **2020**.
[47] S. Niu, A. Todri-Sanial, *Quantum* **2023**, *7*, 925.
[48] Y. Ohkura, T. Satoh, R. Van Meter, *IEEE Trans. Quantum Eng.* **2022**, *3*, 1.

