One-dimensional Hadamard Quantum Walk on a Cycle with Rotational Implementation

Konstantinos Georgopoulos and Paolo Zuliani

School of Computing, Newcastle University, Newcastle-upon-Tyne, United Kingdom

(Dated: January 20, 2020)

Quantum walks have been extensively studied recently, mainly due to their vast difference in behavior to classical random walks. This paper is concerned with discrete time and space quantum walks of particles that propagate through a one-dimensional line. This line can be either a lattice or a graph or any other form of mathematical structure that can be viewed as a one-dimensional line. First is defined a concrete way to describe the unitary evolution of a quantum walk through a balanced coin operator and a shift operator. Then follows the implementation of the quantum walk on an 8-cycle, i.e a cycle graph with 8 nodes, which is then run locally as a simulation and on IBM’s quantum computer. The paper explores two implementations of the quantum walk as a quantum circuit: the first one consists of generalised controlled inversions, as introduced in [9], whereas the second one tries to replace them with rotation operators around the basis states. The main aim is to find a way around the caveat resulting from the large amount of ancilla qubits required to carry out the computation in the case where only generalised inverters are used. Next, another three experiments are computed, involving cycles with a larger state space, more specifically 16, 32 and 64 possible positions. In order to measure the magnitude of the error of the circuit we use the cross entropy benchmarking method, calculated through the Hellinger distance. Finally, a derivation of the variance of the quantum walk is provided along with a calculation of the variance for our experiment.

I. INTRODUCTION

Classical random walks are a very common and well studied scientific area and consist one of the most fundamental processes in physics and computer science. Usually, in the simplest discrete case, a walker is considered (walker being a particle, for example) with the ability to move through the adjacent discrete positions of a one-dimensional, bidirectional lattice. In each discrete time step a fair coin is flipped and, depending on the outcome, the walker moves one position to the right or to the left. Such discrete time and space random walks can be generalised to any form of finite or infinite lattices and graphs, which can be viewed as random walks on the line.

This paper considers the quantum mechanical analogue of a random walk on the line, or otherwise referred to as a quantum walk. These processes differ from the classical random walks in terms of the dynamics of the particle’s propagation through the state space. The state space consists of all the possible discrete positions that the particle can be in. They can be represented as points on a line or nodes on a lattice or a graph. In quantum walks, the particle has an extra degree of freedom that assists in its motion, called chirality. This property results to the vast difference between classical and quantum walks.

Chirality can take two values, Left and Right. If we imagine a quantum particle that moves freely between adjacent discrete points on a one-dimensional line, then at each time step, the balanced coin is flipped and the chirality state undergoes a unitary transformation, otherwise called shift. Then the particle progresses according to its new chirality state, thus evolving the walk. This allows us to consider the quantum walk as a translation invariant local unitary process.

This type of quantum walk is called a Hadamard walk, earning its name from the use of a Hadamard operator as the balanced coin. Figure 1 depicts the dynamics of the quantum walk on a one-dimensional line.

![FIG. 1: Dynamics of the Hadamarad quantum walk.](image)

(a) The walk begins at time $t$ with chirality Left or Right. (b) The result of the Hadamard transform where the particles are in equal superposition of their two chirality states. (c) The particles move accordingly to generate the state in time $t+1$.

The behaviour of the quantum Hadamard walk is very different to that of the classical random walk. The reason is quantum interference. Whereas there cannot be constructive or destructive interference in a classical random walk, in the quantum walk two separate paths leading to the same point may be massively out of phase and cancel one another.
Quantum walks have the potential of offering the means to speed up classical algorithms that are based on random walks. There have been many systematic studies on this subject area and many of them can lead to further in-depth analysis of more advanced quantum algorithms, such as quantum Metropolis, quantum Markov chains or quantum Monte Carlo methods [7, 13, 19, 18].

Throughout this paper we make use of IBM’s quantum computer. We try to simulate a particle’s one-dimensional quantum walk on a finite cycle [3]. This walk can be treated as a discrete state and discrete time quantum walk on a finite cycle [3]. The discrete time quantum walk can be described by a cycle with 8 nodes, where the particle can discretely exist in 8 possible positions. In order to represent the states become $|0\rangle$, |1\rangle, ..., |7\rangle. After the state |111\rangle the cycle goes back to |000\rangle.

**FIG. 2:** Graph representing an $N$ cycle, for $N = 8$ nodes (or possible states for the walk).

For our experiment we will use an 8-cycle, or otherwise, a cycle with 8 nodes, where the particle can discretely exist in 8 possible positions. In order to represent the positions, $n = \log N = 3$ qubits are needed. This way, the states become $|000\rangle, |001\rangle, ..., |111\rangle$ or, equivalently, $|0\rangle, |1\rangle, ..., |7\rangle$. After the state $|111\rangle$ the cycle goes back to $|000\rangle$.

### II. QUANTUM WALKS

The discrete time quantum walk can be described by the repeated application of a unitary evolution operator, $U$. This operator acts on a Hilbert space $\mathcal{H}^C \otimes \mathcal{H}^S$, where $\mathcal{H}^C$ is the Hilbert space associated with a quantum coin and $\mathcal{H}^S$ is the Hilbert space associated with the state space (positions, nodes on the graph) of the walk. In order to describe the quantum walk we use the unitary operator, $U$, given as

$$U = S \cdot (C \otimes \mathbb{I})$$

where $S$ is the shift operator, or the operator that describes how the particle moves to the next state, and $C$ is the quantum coin, or the operator that randomly “chooses” which path the particle takes. The quantum walk can take one of two directions: either go right, or increase its position by a step of 1, or go left and decrease its position by a step of 1.

Since our experiment is about quantum walks of particles, it is important to talk a bit about what this means for our research. All elementary particles have a fundamental property called spin. The spin of a particle exists as a binary value and can be either spin-up, $|\uparrow\rangle$, or spin-down, $|\downarrow\rangle$. The initial state of a particle (spin-up or -down) before the quantum walk affects the behavior of the walk, introducing direction bias through an additional degree of freedom called chirality. The chirality of the particle, as mentioned in the previous section, can be either Left ($|L\rangle$) or Right ($|R\rangle$). One way to remove the bias introduced by the chirality state is to use a balanced quantum coin. One such coin is the Hadamard coin.

For our experiment, the quantum coin, $C$, is simply a Hadamard operator, or the aforementioned Hadamard coin, with matrix representation:

$$C = H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

The shift operator $S$ defines the direction the particle will take after each coin flip and can be described as a permutation matrix. These permutations can be achieved via increment and decrement functions, as demonstrated by [9]. Thus, since there is only two directions the particle can take, we can describe the $S$ operator in terms of progressing the state by adding 1 to the current position or subtracting 1 from the current position. Thus, we can describe the shift operator as

$$S = S^- \otimes |0\rangle \langle 0| + S^+ \otimes |1\rangle \langle 1|$$

where $S^+$ is an operator for increasing our position by one, or otherwise $S^+ |x\rangle \rightarrow |x + 1\rangle$ as moving one step to the right and $S^-$ for decreasing the position, or otherwise $S^- |x\rangle \rightarrow |x - 1\rangle$ as moving one step to the left.

In terms of a quantum circuit, we can describe the quantum walk on an $N$ cycle as follows: first we apply a Hadamard gate $H$ (which represents the coin operator $C$) on a single qubit quantum register initialised at state $|0\rangle$. This puts the qubit in a superposition of states described by $\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)$. Then, we propagate through the graph according to the shift operator, $S$. When $S$ acts on the coin state $|0\rangle$, then the particle moves one position to the left, decreasing its current state by 1. When it acts on $|1\rangle$ it moves one position to the right, increasing its current state by 1. We allow the quantum walk to evolve...
for a discrete number of steps, \( t \), and then we measure. Each time we measure, the particle can be in one of \( N/2 \) different states with equal probability. In other words, if we observe the position of the particle a large number of times (i.e., 1,000,000 runs of the quantum walk for \( t \) steps each time), we will find the particle in each position approximately equal times (i.e., 250,000 times in each of the four possible states).

It is important here to explain the reason why only four states can be observed each time. This happens due to the modularity of the quantum walk \([17]\). Assuming the walk is initialised in an odd position, if we let the particle move for an odd/even number of steps then, after measurement, we can only observe the particle in even/odd position. If the particle starts on an even position (including \(|0\rangle\)) then, after an odd/even number of steps the particle’s position will be an odd/even state.

As mentioned earlier, if we measure the position of a particle an infinite number of times, then with equal probability we will observe the particle in all possible states, in accordance with the modularity property. This type of quantum walk is called a symmetric quantum walk. In our case however, the quantum walk is asymmetric. This means that, unlike the above description, after the evolution of the particle’s position, the probability of each state to be measured is not the same.

The reason for the above phenomenon is quantum interference. Interference in quantum mechanics occurs mainly due to the mathematical properties of the amplitudes. To be more precise, the amplitudes are complex numbers and can, thus, be positive and negative. When the wave function (partially) collapses the probabilities can be calculated as the modulus squared of the amplitudes in the superposition. For example, if we encounter a superposition \( \alpha |0\rangle + \beta |1\rangle \), where the amplitudes are \( \alpha, \beta \in \mathbb{C} \), then the probabilities are \( |\alpha|^2 \) and \( |\beta|^2 \), where of course \( |\alpha|^2 + |\beta|^2 = 1 \). Thus, we can easily see that any negative or complex amplitude vanishes and the probabilities remain real numbers in [0, 1].

On the other hand, the fact that the amplitudes are complex numbers leads to the effect of interference. Interference can be either constructive or destructive. This can affect the quantum walk experiment when we evolve for a number of steps, i.e., many iterations of the shift operator, \( S \). Precisely, the leftwards path \( (S^-) \) interferes more destructively, as it is multiplied by \(-1\), whereas the rightwards path undergoes more constructive interference and the system tends to take steps towards the left. This results to the asymmetry of the quantum walk \([11]\). In other words, the asymmetry is the result of the Hadamard coin treating the states \(|↑\rangle\) and \(|↓\rangle\) asymmetrically.

To reach symmetric results we can use one of two solutions. The first is an initialisation solution, where the particle starts in a balanced superposition of the form \( \frac{1}{\sqrt{2}}(|↑\rangle + i|↓\rangle) \). The second solution makes use of a different coin operator, \( C \). Instead of the Hadamard operator, shown in equation \((2)\), we could use an operator which doesn’t bias the coin towards a certain base vector. This operator can be expressed, for example, as

\[
Y = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}.
\]

A visualisation of the asymmetry of the quantum walk is given in Figure 3.

### A. Algorithmic Speedups via Quantum Walks

One very important property of quantum walks is their ability to provide quantum advantage over various classical algorithms or, in other words, give algorithmic speedup. There is various examples of quantum speedups via quantum walks, especially since the early 2000s.

One first example is the use of quantum walks on graphs. In their 2001 paper, Dorit Aharonov et al. \([1]\) generalise random walks on graphs on the quantum world via quantum walks. These walks do not converge to any stationary distribution, as they are unitary and reversible. Aharonov et al. prove that, in the context of propagation of the quantum walk through the graph, the quantum walk is almost quadratically faster than the classical random walk. This result is analogous to the quadratic increase in efficiency presented in this paper.

In another research, Andris Ambainis in 2007 \([2]\) uses quantum walks to construct a quantum algorithm for element distinctness, along with its generalization. Element distinctness can be defined as the problem of finding if there is two identical items within a set of \( N \) given objects. Ambainis proves a query algorithm of complexity \( O(N^{2/3}) \) for a pair of identical items, along with an algorithm of complexity \( O(N^{k/(k+1)}) \) for the generalization of finding \( k \) equal items in the set. Classically, the element distinctness problem can be solved in \( O(N \log N) \).

One of the most impressive pieces of work done in this area comes by Childs et al. in 2003 \([3]\). The authors construct an oracular graph traversing problem that can
be solved by a quantum computer exponentially faster than any classical computer. As opposed to the problems above, as well as the research in this paper, this work uses a continuous time quantum walk. Finally, it is proven by Childs et al. that this problem cannot be solved by any classical algorithm in sub-exponential time.

A similar result found by Julia Kempe in 2005 \[10\]. The author shows that the hitting time of a discrete time quantum walk on an \(N\)-bit hypercube from one corner to its opposite is polynomial in \(N\), giving an exponential speedup over the classical hitting time of random walks.

Another work done by Childs and Goldstone in 2004 \[8\] refers to using quantum walks to perform spatial search. Grover’s quantum search algorithm provides a quantum speedup over classical algorithms on the quadratic scale. Childs and Goldstone \[8\] base their work on continuous time quantum walks to prove that they can achieve a speedup of \(O(\sqrt{N})\) on a \(d\)-dimensional periodic lattice, where \(d > 4\).

Additionally on the context of quantum search, Magniez et al. \[12\] show a method for designing quantum algorithms for finding marked elements using quantum walks. In this case, the quantum walk is based on Szegedy’s \[18\] quantum Markov chain constructs. Their main idea evolves around applying quantum phase estimation on the quantum walk to implement an approximate reflection operator, which can then be used for amplitude amplification.

Finally, another work worth mentioning here is the quantum speedup of backtracking algorithms by Ashley Montanaro in 2018 \[14\]. Backtracking algorithms explore a tree whose vertices are partial solutions to a constraint satisfaction problem in an attempt to find a complete solution. Assuming that the problem is to find a solution to a constraint satisfaction problem of \(N\) variables (or output that there is no solution), whose tree has \(T\) vertices. Montanaro shows that this can be done in \(O(\sqrt{T}N^{3/2}\log N)\) queries via the use of quantum walks.

The next step is to talk about how we can quantum mechanically increment and decrement the state of the particle.

III. IMPLEMENTING THE INCREMENT AND DECREMENT FUNCTIONS

One way to implement the increment and decrement functions that evolve the state of the quantum walk is to use generalised CNOT gates (NOT gates with more than two control qubits), as demonstrated by \[9\]. There is also a different way to do addition via Quantum Fourier Transform (QFT). This method allows the addition of two arbitrary numbers, something that results to a larger and more complicated circuit. For our case though, a simple counter is enough and there is no need to accommodate for a larger scale addition.

The implementation of the increment and decrement functions is discussed in section [III A] below. Section [III B] analyses our approach that uses rotations around the basis states in order to increase and decrease the state of the system. Both methods are used for the experiment, something that also allows us to compare their results.

A. Using generalised Inverter Gates

First we discuss the generalised CNOT, or otherwise generalised inverter method used by \[9\]. This method relies exclusively on the use of inverter gates, or Pauli-X or NOT gates. The matrix that describes the inversion operator is the Pauli-X matrix, given by the representation

\[
\sigma_x = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}
\]

This matrix can be expanded to represent a controlled-NOT (or CNOT) operation by adding an additional control qubit. In this case, the inversion will occur on the target qubit only if the control qubit is in the state \(\vert 1 \rangle\). The matrix representation of this gate is

\[
X_c = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}
\]

When we talk about generalised CNOT, or in extent, any generalised controlled unitary, we refer to operations that are controlled by more than one control qubit. In this case, the operation (be it inversion or any other unitary) will occur if and only if all the control qubits are in state \(\vert 1 \rangle\). The most common example and one that is used extensively here is the Toffoli gate, or controlled-controlled-NOT (or CCNOT) gate. This gate has two control qubits and one target qubit. The target qubit will be inverted only if the control qubits are both in the quantum state \(\vert 1 \rangle\). In other words, it will be inverted only if the addition modulo 2 of the two control qubits results to 1. The truth tables of the NOT, CNOT and CCNOT (Toffoli) gates are shown in Figure 4.

Of course, generalised CNOT gates along with the Toffoli gate include inverters with more than two control qubits. Alas, one important drawback is that, unlike the Toffoli gate, there is no hardware implementation of a generalised CNOT gate with more than two control qubits. The solution for this limitation is to use Toffoli gates in order to store in external ancilla qubits the intermediate CNOT operations between the control qubits. In other words, the addition modulo 2 operation of specific pairs of control qubits and ancilla qubits is computed and stored in a new ancilla qubit. This process builds up and the last stored ancilla qubit will control the overall operation by acting as a single control qubit on the target. A more detailed description of this process is given in Appendix A.
FIG. 4: Truth table for NOT, controlled-NOT and Toffoli gates. In this, the target qubit $t$ is the leftmost qubit in each table, and the control qubit(s) $c$ is(are) the rest.

FIG. 5: Implementation of one step for the quantum walk of a particle.

Now we have the foundations to use the generalised CNOT gates, as discussed above, to construct a quantum circuit that implements the increment and decrement functions. These functions are used to evolve the quantum walk, or otherwise move the particle around, according to the shift operator, $S$, as expressed in equation (3). It is apparent that there is a need for two individual quantum circuits, one for the increment and one for the decrement function. As shown by [9], this can break down to just one quantum circuit, used with opposite control logic. Figure 5 shows the higher level circuit for one iteration of the quantum walk. The implementation of the increment and decrement circuits is shown in Figure 6a. The truth table for this is shown in Figure 6b.

As we can see, the increment and decrement operations are modulo $N$, or rather modulo 8 for our 8 state space experiment. This means that, when the particle reaches the quantum state $|111\rangle$, in the case of increment it will progress to state $|000\rangle$. In the case of decrement, when it reaches the quantum state $|000\rangle$ it will move on to state $|111\rangle$.

Fortunately, IBM’s development kit offers a very nice way of producing circuit schematics. The quantum circuits that visualise the implementation of the increment and decrement functions are shown in Figure 7a and 7b, respectively.

FIG. 6: (a) Quantum circuits for increment and decrement operations. A filled control circle means that the control qubits have to be in state $|1\rangle$ in order for the operation to occur. An empty control circle means they have to be in state $|0\rangle$. (b) Truth tables for increment and decrement functions.

B. Using Rotations

The method presented in the previous section is a simple way to increase and decrease the position of the particle during the quantum walk. The main limitation of the approach is that, in order for the generalised CNOT gates to be implemented, there is a need for additional ancilla qubits. This leads to a significant increase of the workspace (i.e. the number of qubits needed for the computation). To be more precise, a generalised CNOT gate with $n$ control qubits requires additional $n-1$ ancilla qubits for the implementation. Thus, considering the quantum computers available to us today, the largest one is the 16 qubits IBMQ Melbourne device, of which only 14 are active. This means that, at the very best, we can implement a quantum walk on a cycle with at most $2^7 = 128$ states. The reason is that, a quantum walk with a state space of 7 qubits has at most 6 control qubits (for the leftmost generalised CNOT) and thus requires an ancilla register with 5 qubits in order for the generalised CNOTs to be implemented. Considering the single qubit coin register as well, it adds up to the maximum workspace capacity of 13 qubits. Trying to have
an additional qubit for the state space will not work, as it will need 14 qubits for the position and ancilla registers, and there is no space left for the single qubit coin register.

In this work we managed to prove and implement a solution that uses no ancilla qubits at all, resulting to the ability to perform a quantum walk using the entire capacity of the quantum computer’s workspace. This means that we can use all the 14 qubits of IBMQ’s quantum computer for the state space of the walk, resulting to a cycle with $2^{13} = 8192$ positions. This is a significant rise, compared to the generalised CNOT approach. It is noteworthy that the reason why 14 qubits lead to $2^{13}$ possible positions is because, again, 1 qubit is used in the coin register.

This solution uses rotations around the basis states in order to implement the quantum walk. There is a number of gates within IBMQ’s backends that can implement those rotations.

Before we showcase our quantum circuit for the rotational approach, we need to start by expressing three very important lemmas, first introduced and proven by [4]. The first two essentially allows us to turn any generalised unitary operator to a sequence of rotations that lead to the same result as the unitary itself. These are expressed in Lemmas [III.1] and [III.2]

**Lemma III.1** Every unitary $2 \times 2$ matrix with unity determinant can be expressed by the following sequence of matrix multiplications

$$
\begin{pmatrix}
0 & e^{i\delta} \\
e^{i\delta} & 0
\end{pmatrix}
\begin{pmatrix}
e^{i\alpha/2} & 0 \\
0 & e^{-i\alpha/2}
\end{pmatrix}
\begin{pmatrix}
cos \theta/2 & \sin \theta/2 \\
\sin \theta/2 & \cos \theta/2
\end{pmatrix}
\begin{pmatrix}
0 & e^{i\beta/2} \\
e^{-i\beta/2} & 0
\end{pmatrix},
\]

where $\delta, \alpha, \theta$ and $\beta$ are real valued.

**Lemma III.2** For any special unitary operator $W$, where $W \in SU(2)$, there exist operators $A$, $B$ and $C \in SU(2)$ such that $ABC = I$ and $A \sigma_x B \sigma_x C = W$.

**Proof** By Lemma [III.1] we can show that there exist $\alpha, \theta$ and $\beta$ such that $W = R_z(\alpha)R_y(\theta)R_z(\beta)$, where $R_z, R_y$ are rotation functions expressed in equations [7] and [8] respectively. Thus, we can set $A = R_z(\alpha)R_y(\theta/2), B = R_y(-\theta/2)R_z(-\alpha + \beta)/2$ and $C = R_z((\beta - \alpha)/2)$. Then

$$ABC = R_z(\alpha)R_y\left(\frac{\theta}{2}\right)R_y\left(-\frac{\theta}{2}\right)R_z\left(-\frac{\alpha + \beta}{2}\right)$$

$$\times R_z\left(\frac{\beta - \alpha}{2}\right)$$

$$= R_z(\alpha)R_z(-\alpha) = I$$

and thus

$$A \sigma_x B \sigma_x C = R_z(\alpha)R_y\left(\frac{\theta}{2}\right)\sigma_z R_y\left(-\frac{\theta}{2}\right)R_z\left(-\frac{\alpha + \beta}{2}\right)$$

$$\sigma_z R_z\left(\frac{\beta - \alpha}{2}\right)$$

$$= R_z(\alpha)R_y\left(\frac{\theta}{2}\right)\sigma_z R_y\left(-\frac{\theta}{2}\right)\sigma_z$$

$$R_z\left(-\frac{\alpha + \beta}{2}\right)$$

$$= R_z(\alpha)R_y\left(\frac{\theta}{2}\right)R_y\left(\frac{\theta}{2}\right)R_z\left(\frac{\alpha + \beta}{2}\right)$$

$$R_z\left(\frac{\beta - \alpha}{2}\right)$$

$$= R_z(\alpha)R_y(\theta)R_z(\beta) = W.
$$

What we can learn from Lemma [III.2] is that, we can express any unitary operator and more precisely, for our case, a generalised CNOT gate, as a sequence of matrices $A \sigma_x B \sigma_x C$. Thus, we can narrow our efforts down to finding the appropriate $A, B, C$ operators that suit our specific needs (in this case, implement a generalised CNOT).

We have managed to prove and design a quantum circuit that implements a Toffoli gate via a number of controlled rotations. Figure 8 showcases the resulting circuit that implements a Toffoli gate using rotational gates. More information on the quantum mechanical analysis of the quantum circuit is given in Appendix B.

For more convenience, we showcase here the basic gate operators used within the quantum circuit. The specific gates needed can be defined through the unitary rotation matrix, $U_3$, expressed as

$$U_3(\theta, \phi, \lambda) = \begin{pmatrix}
\cos \theta/2 & -e^{i\phi} \sin \theta/2 \\
e^{i\phi} \sin \theta/2 & e^{i(\lambda+\phi)} \cos \theta/2
\end{pmatrix}. \quad (5)$$

Thus, applying the controlled rotation $R_y(\theta/2)$ for $\theta = \pi, \phi = 0$ and $\lambda = 0$ (operation $U_3(\pi/2, 0, 0)$) results in applying the unitary matrix

$$R_y(\theta) = U_3(\theta, 0, 0) = \begin{pmatrix}
\cos \theta/2 & -\sin \theta/2 \\
\sin \theta/2 & \cos \theta/2
\end{pmatrix} \quad \text{for } \theta = \pi/2
$$

and the controlled rotation $R_z(\phi)$ (operation $R_z(\pi/2)$), which results to applying the unitary matrix

$$R_z(\phi) = \begin{pmatrix}
e^{i\phi/2} & 0 \\
0 & e^{-i\phi/2}
\end{pmatrix} \quad \text{for } \phi = \pi.
$$

The same unitaries apply for $\theta = \phi = -\pi$, which are the operations $R_z(-\pi)$ and $U_3(-\pi/2, 0, 0)$ respectively. This quantum circuit has a truth table identical to the Toffoli gate from Figure 4.

The next step is to generalise this quantum circuit so it can accommodate more than two control qubits. In other words, we want to create a generalised CNOT circuit.
FIG. 7: (a) Increment circuit. The three qubit $q_n$ register is the state space of the quantum walk, the coin register $coin_0$ represents the Hadamard coin. The ancilla qubits used for the computation are $anc_0$ and $anc_1$. (b) Decrement circuit. Important here is the need to invert all the control qubits (including the coin) at the start of the computation and uncompute them at the end.

FIG. 8: Quantum circuit implementing a Toffoli gate using conditioned rotations. Using rotations. In order to do this we use another lemma from [9]. Finally, since the inverter gate is not a special unitary (i.e. it doesn’t have determinant 1, there is the need to use an additional phase gate, $\Phi(\delta)$ for $\delta = \pi/2$, where

$$
\Phi(\delta) = \begin{pmatrix} e^{i\delta} & 0 \\ 0 & e^{i\delta} \end{pmatrix}.
$$

Lemma III.3 (Generalised rotational network)

For any $SU(2)$ matrix $W$, a $\wedge_{n-1}(W)$ gate can be simulated by a network of rotational operators, as shown in Figure 7, where $A, B$ and $C \in SU(2)$.

In this context, we need to introduce the notation $\wedge_{n-1}(W)$ as used by [4]. For any unitary matrix $U = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}$ and $m \in \{0, 1, 2, \ldots \}$ we define the $(m+1)$-bit ($2^m+1$-dimensional) operator $\wedge_m(U)$ as

$$
\wedge_m(U)(|x_1, \ldots, x_m, y\rangle) = \begin{cases} 
u_{y0} |x_1, \ldots, x_m, 0\rangle + u_{y1} |x_1, \ldots, x_m, 1\rangle & \text{if } \wedge_k x_k = 1 \\ |x_1, \ldots, x_m, y\rangle & \text{if } \wedge_k x_k = 0 \end{cases}
$$

Lemma III.3 describes a way to expand any generalised unitary with an arbitrary number $m$ of control qubits to a network of controlled rotation gates and generalised CNOTs of the form $A\sigma_x B\sigma_x C$. It is easy to see that, if $W = X = \sigma_x$, we can iteratively expand each one of the generalised CNOT gates to a new network of the form $\Phi A\sigma_x B\sigma_x C$. The expansion will stop when the generalised CNOT gates end up being regular Toffoli gates, i.e. have two control qubits. The unitary $2 \times 2$ operator $W$ that results from this approach is

$$
W = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},
$$

For more information regarding the mathematical derivation of the operator see Appendix B.

Thus, we have reached a quantum circuit that implements a generalised CNOT gate with an arbitrary number of control qubits without depending on the use of any ancilla qubits. This logic can be applied not only to generalised CNOT gates, but to any unitary operator that is controlled by an arbitrary number of qubits.

We can now integrate this implementation in the increment and decrement circuits, as those were described in the previous section. We will substitute the Toffoli gates that require ancillas with the network described in Lemma III.3. Any Toffoli, CNOT or inverter $X$ gate that is not an expansion of a generalised CNOT operator will remain the same. A visualisation of this quantum circuit is shown in Figure 10. The decrement circuit will follow similar logic with the difference that all the control qubits have to be inverted at the start of the computation, so that the quantum operation can be controlled by $|0\rangle$ qubits instead of $|1\rangle$. 
IV. IMPLEMENTING THE QUANTUM WALK

Following the above analysis, we can easily implement and study the quantum walk of a particle. Following exactly the unitary that describes the evolution of the particle’s quantum walk, as it was expressed in equation (1), we can express each step of the quantum walk as shown in Figure 5.

For our experiment we implement a quantum walk on an 8 cycle, or otherwise a cycle where the particle can be in 8 different states, as discussed in previous sections (see Figure 2). The implementation of the quantum walk, for a number of steps, t, can be described as shown in Algorithm 1.

Algorithm 1: Quantum Walk on a Cycle

1 Initialisation. Initialise the quantum registers in all |0⟩ (ground) states.
2 State preparation. Prepare the workspace in the state that represents the initial position of the particle.
3 Coin flip. Flip the quantum coin. The coin operator, C, is implemented as a simple Hadamard coin. This can be done by using a Hadamard gate, H, applied on the coin register.
4 Evolution. The position of the particle evolves following the unitary shift operator, S. This operator is implemented using the increment and decrement functions, as described in section II.
5 Repeat for t. If the coin has been flipped t times, then go to step 6. Otherwise, repeat from step 3.
6 Measure. Measure the quantum state of the position register and observe the position of the particle. This will reveal the state of the quantum system, thus collapsing the superpositions and destroying the quantum mechanical properties of the walk.

It is important to state here that, the position of the particle after a number of steps t depends on t itself, due to the aforementioned property of modularity. This means that, for an experiment that is initialised on an odd position, if t is odd, when we measure, we will find the particle in an even position. The opposite applies if t is even.

This property stands when the quantum walk is implemented on a simulator, but, as we will see in the following section, it gets partially violated when the experiment is run on the quantum computer. The reason for that is lack of quantum error correction.

The quantum walk produces a very long circuit with a large number of gates, which is why it is not included as a figure in this paper. To give a sense of the difference between the circuits produced by the two implementations, we provide the number of gates of each circuit. For N = 8 nodes on the cycle and t = 1 coin flips, the generalised CNOT circuit contains 23 quantum gates, more specifically, generalised CNOTs, Toffoli and inverter gates. The rotational circuit, on the other hand, contains 25 gates. Although, opposed to the generalised CNOT implementation, the rotational implementation contain U5 and Rz gates, which are more complex than the CNOTs. Additionally, due to the complexity of lemma III.3, the difference in number of gates between the two approaches increases with the state space. For example, for N = 16 and t = 1, the generalised CNOT implementation has 39 gates whereas the rotational has 59.

In the next section we state the results from running the experiment on a local simulator and on IBM’s quantum computer.

V. STATEMENT AND DISCUSSION OF RESULTS

In this section we discuss the results from running our quantum walk experiment locally as a simulation, using IBM’s quantum development kit (Qiskit), and on IBM’s quantum computer. The initial state of the particle is the position |3⟩ or |011⟩.

A. Simulation

First of all, we talk about the results provided from the local simulation of the particle’s quantum walk experiment. For this we use IBM’s Qiskit, which provides an excellent range of tools that allow us to implement the quantum circuit and simulate it on a classical computer. The quantum walk experiment is run for a total of 15 steps (iterations) before the final quantum state is produced. The implementation follows the steps of Algorithm 1.

For the simulation we use both the generalised CNOT and the rotational implementation for the shift operator, as given by equation (3). The results of the experiment are given in the histograms of Figure 11. These histograms show the probability distribution of the possible positions of the particle after 15 steps of the quantum walk. The probability distribution over the states is read off the simulated final state (which is in a superposition) in Qiskit’s simulator.

There is two important points to observe here: the asymmetry and the modularity of the quantum walk. As explained in section II, quantum walks show asymmetry through the evolution of states. In other words, when we measure the position of the particle after our t = 15 steps, we do not find it to each possible position with equal probability. The reason for that is, mainly, the effects of interference, either constructive or destructive. A more detailed analysis on this was presented in section II.

The second point involves the modularity of the quantum walk. As has been explained earlier, this specific property means that if the quantum walk evolves for an odd/even number of steps, then the particle can only be in even/odd posi-
FIG. 11: Probability distribution of output states for the simulated quantum walk on an 8-cycle. Tiny differences among the probabilities of the two approaches appears around the $8^{th}$ decimal place.

B. Quantum Computer

Following the above, we discuss the results we get from running the experiment on the quantum computer. For this we use IBM’s 16 qubits quantum computer (IBM Q 14 Melbourne). Note here that, for the rotational implementation on a 3 qubit state space, we could also use the 5 qubits device (IBM Q 5 Tenerife), since there is no need for additional ancilla qubits for the implementation of the generalised CNOT gates. Evidently, the required resources regarding the number of qubits needed for the experiment drop in the case of rotational implementation.

It is important to notice here that the above claim holds when considering the direct implementation in the fundamental operations for a quantum device. In general, error corrected or fault tolerant quantum information processors may require substantial overhead in ancillary qubit count to implement rotations about arbitrary angles, due to the use of $T$-operations to implement said rotations. Thus, for the scope of this research, we are interested only in application to current and near-term (that is, not fault-tolerant) devices.

The results of running the experiment on the quantum computer are shown in Figure 12. The quantum walk is allowed to run for 10,000 iterations and the histograms shown represent the probabilities of the quantum states to occur as the sum of all the occurrences of said states over the total number of 10,000 iterations. These histograms look very different from the simulation, as we encounter some states that we don’t expect to, as they violate the modularity property of the quantum walk. The main reason for that is quantum noise in the machine.

Quantum computers, unlike classical computers that we are used to, are not fault tolerant. In other words, quantum machines produce many errors during computations. The reason for that is quantum noise. A quantum computer works with quantum particles, or otherwise called, qubits. A qubit, additional to superposition, can have another quantum property which is to lose or gain energy levels. To be more precise, a qubit that is on the ground $\ket{0}$ state can gain an energy level and jump to an excited $\ket{1}$ state, or the other way around. This introduces noise in the computation.
VI. QUANTUM WALKS ON LARGER CYCLES

This section is concerned with the implementation of one-dimensional Hadamard quantum walks on cycles with a large number of nodes. This means that the state space of the walk will be bigger, i.e., a larger number of possible positions for the particle. Three different quantum walk experiments are performed, on a 16-cycle, a 32-cycle, and a 64-cycle.

A. Quantum Walk on a 16-Cycle

The first quantum walk is done on a 16-cycle, or otherwise a cycle with 16 nodes. For \( N = 16 \) different positions, \( n = \log N = 4 \) qubits are needed, hence the state space for the particle is \(|0000\), \ldots, |1111\). For this experiment, the particle is initialised in position \(|0000\) and it evolves using the coin and shift operators, as described in equations (2) and (3). The walk evolves for \( t = 5 \) time steps (i.e., coin flips).

The results of this walk are shown in Figure 13a. It is shown that the properties of the quantum walk are preserved in this case. More precisely, the modularity is preserved. Note here that, taking into account the initial position \(|0\) and letting the walk evolve for \( t = 5 \) (odd number of steps), the modularity results to the observed odd positions, as shown in the figure.

The quantum circuit for the 16-cycle has 195 quantum gates. These quantum gates consist entirely of generalised CNOT, Toffoli, and inverter gates.

The walk is then run on IBM’s quantum computer. The experiment returns the results shown in Figure 13b. Similarly to the 8-state space walk implemented earlier, the quantum computer evolves the particle to all the possible states, violating modularity due to quantum error.

B. Quantum Walk on a 32-Cycle

The second quantum walk is done on a 32-cycle, or otherwise a cycle with 32 nodes. For \( N = 32 \) different positions, \( n = 5 \) qubits are needed, hence the state space for the particle is \(|00000\), \ldots, |11111\). Again, the particle is initialised in position \(|00000\) and it evolves using the coin and shift operators, as described in equations (2) and (3). The walk evolves for \( t = 12 \) time steps (i.e., coin flips).

The results of this walk are shown in Figure 14a. It is shown that the properties of the quantum walk are preserved as well. It is again noteworthy that, taking into account the initial position \(|0\) and the progression of the walk for \( t = 12 \) coin flips (even number), the particle is encountered in even positions.

The generated quantum circuit contains 708 quantum gates in total. This is a big leap compared to the previous experiments. The reason for that is the need to use more Toffoli gates, as there are more control qubits on the leftmost generalised CNOT gate. Also, there is a greater number of repetitions of the increment and decrement circuits, as the coin is flipped more times.

Once again, the walk is run on the quantum computer. The results of the experiment are shown in Figure 14b. Similarly to the previous 16-cycle experiment, the modularity property is violated as all possible positions of the state space appear on the results.

C. Quantum Walk on a 64-Cycle

The last quantum walk is done on a 64-cycle, or otherwise a cycle with 64 nodes. For \( N = 64 \) different positions, \( n = 6 \) qubits are needed, hence the state space for the particle is \(|000000\), \ldots, |111111\). Again, the particle is initialised in position \(|000000\) and it evolves using the coin and shift operators, as described in equations (2) and (3). The walk evolves for \( t = 32 \) time steps (i.e., coin flips).

The results of this walk are shown in Figure 15. It is apparent that, similarly to the previous cases, the properties of the quantum walk are preserved and more precisely, the modularity property. Again, taking into account the initial position \(|0\) and that the walk evolves for an even number of coin flips, we observe the particle in even positions.

As expected, the complexity of the circuit has increased massively with the quantum circuit made of 2656 quantum gates (generalised CNOT gates).
Finally, the experiment is attempted to run on the quantum computer. Alas, this endeavour has been unsuccessful. For $t = 32$ coin flips the quantum computer could not handle the request, returning an error message “Payload Too Large”. The next attempt was done for $t = 16$, $t = 10$ and $t = 5$ coin flips, but again the quantum computer failed to handle the request. The experiment was attempted on the 16 qubit IBM Melbourne device which is the largest one available publicly. Perhaps IBM’s 20 qubit machine could succeed, but alas it is only available to IBM clients.

VII. SIMULATION AND QUANTUM COMPUTER VARIANCE AND MIXING TIME OF THE QUANTUM WALK

The variance of a quantum walk is a very important parameter, as it shows how far the particle will travel in the specific number of coin flips. It is shown in the next section [VII] that the variance of a quantum walk increases near-quadratically with the number of coin flips: $\sigma^2 \sim t^2$.

We can create a method to calculate the variance of the quantum walk. As proven in [VIII], the variance can be calculated exactly from the equation

$$\sigma^2 = t^2 \left( \sqrt{2} - \frac{1}{\sqrt{2}} \right) \left( 1 - \frac{\sqrt{2} - 1}{\sqrt{2}} \right) \right)$$

where $t$ is the number of steps or coin flips done in the quantum walk.

It is easy to see that the standard deviation is simply the square root of the variance, and it increases almost linearly with the number of steps, $\sigma \sim t$.

It is important to remember that, due to the initial state being the $|0\rangle$ state and the particle initialised in chirality $|0\rangle$ (Left), the particle will progress towards the left with higher probability, or anticlockwise around the cycle. For example, in a 16-cycle starting at state $|0001\rangle$, the particle is biased towards state $|1111\rangle$.

We can see that the results of the quantum walk experiments (simulations, as the quantum computer violates the modularity of the quantum walk) correspond with the standard deviation and, in extent, the variance, proving that the variance increases linearly with the number of coin flips $\sigma^2 \sim t^2$.

A comparison between the simulation variance of the quantum walk and the theoretical variance is visualised in figure 16.

On the contrary to the simulated experiments, the variance of the quantum walk the real quantum machine is much more complicated to obtain. As mentioned before, the quantum computers available to us are subjects to large amounts of noise. The result of the noisy evolution of the walk is the violation of the modularity. In other words, the property

![FIG. 14: (a) Quantum walk simulation on a 32-cycle for $t = 12$ coin flips using the generalised CNOT implementation. (b) Quantum walk run on the quantum computer on a 32-cycle for $t = 12$ coin flips using the generalised CNOT implementation.](image1)

![FIG. 15: Quantum walk simulation on a 64-cycle for $t = 32$ coin flips using the generalised CNOT implementation.](image2)
that defines the positions of the particle after each step no longer stands allowing the particle to be found in erroneous positions. For example, in the quantum walk on a 16-cycle the particle will be found in all possible 16 states after a number of coin flips that should only have allowed it to traverse a much smaller portion of the state space (i.e visual comparison in figures 14a and 14b).

Unfortunately, the aforementioned behavioral violation of the quantum walk results to the inability to calculate the variance, as the particle will evolve erroneously and exist in positions it should not be found in. Similar observation can be done for the mixing time of the quantum walk. In short, the mixing time of the quantum walk expresses how many steps will it take for the walk to converge to the stationary distribution. Erroneous positions of the particle during the quantum walk limits the ability to calculate the mixing time, same as with the variance.

Additionally, the above error influences how the variance and the mixing time scale with the size of the walk. During the simulations we have calculated that the variance increases analogous to the square of the coin flips \( \sigma^2 \sim t^2 \) (i.e quadratically faster than the classical walk). This result is also theoretically proven in section VIII. In other words, if we increase the size of the quantum circuit allowing for double the amount coin flips, then we will see increase in the variance that follows the equation \( \sigma^2 \sim t^2 \). Alas, as expected, this does not happen in the quantum computer. The erroneous evolution of the quantum walk leads the particle to erroneous position, leading the variance to erroneous values.

On the other hand, the variance of the quantum walk does not change with the size of the state space (or the system size) of the experiment. Adding more qubits or increasing the size of the cycle does not alter the variance or the mixing time of the system. This can be realised from the simulated experiments as well when comparing the variance on two different cycles (i.e the 16- and the 32-cycles).

A way to fix the issues that arise due to quantum errors is to use quantum error correcting techniques in our circuits. A brief overview of our goals for the future of this research can be seen in section XI.

### A. Quantum States Analysis

This section attempts to retrieve the final state of the system before measurement, or in other words, the quantum state before partially collapsing the superpositions. The purpose of this is to be able to look at the amplitudes of the states before collapse and determine the quantum state of the system. For this purpose we are gonna use the statevector simulator included in Qiskit.

For this analysis the 8-cycle is used and it can be easily extended to the other quantum walks. One drawback in this endeavour is that the statevector simulator also includes the ancilla register within the circuit, as well as the coin register. Thus, the quantum state contains the entire workspace and there is a need to separate the state space from the workspace. For example in the 8-cycle case a superposition of 64 states is produced that includes the coin and the ancilla registers.

It is important to state here that the complex numbers in each quantum state are the amplitudes, \( a \). This means that the probability of the particle to be in each state is the amplitude’s modulus squared. For example, if we run the quantum walk for \( t = 4 \) iterations, then we can get the quantum state before measurement as

\[
|\psi_{N=8}^{t=5}\rangle = -0.25 \times |000000\rangle + 0.25 \times |000101\rangle + 0.25 \times |001010\rangle + 0.25 \times |001101\rangle + 0.25 \times |010010\rangle + 0.25 \times |010101\rangle + 0.25 \times |011001\rangle + 0.25 \times |011100\rangle
\]

and the probabilities for every state are the amplitudes modulus of each state squared, i.e \( p_0 = 0.0625 = \left(\frac{1}{8}\right)^2 \), \( p_{000000} = 0.0625 \), \( p_{000100} = 0.0625 \), \( p_{001000} = 0.0625 \), \( p_{001100} = 0.0625 \), \( p_{010010} = 0.0625 \), \( p_{010101} = 0.0625 \), \( p_{011001} = 0.0625 \), \( p_{011100} = 0.0625 \). A simple addition shows that the sum of all the probabilities is \( \sum_{i=0}^{63} p_i = 1 \).

Now we can see that, in the case of probabilities, we can disregard the coin and ancilla operators. Thus, the probability of each positions of the state space can be found according to the equation

\[
p_i = \sum_{j \in \{0, 1\}^{N \mod \text{mod } 8 = i}} p_j
\]

where \( N \) is the number of positions in the cycle (otherwise the size of the state space), \( n = \log N \) is the number of qubits used to represent the state space and \( q \) is the number of qubits used in the work space (including ancilla and coin registers). If we want to simplify the above equation we can use the decimal representation to avoid confusion as

\[
p_i = \sum_{j \in \{0, 2^n-1\} \mod \text{mod } N = i} p_j
\]

As an example, we can use the above equation to find the probability of the particle to be in the position |000\rangle after \( t = 4 \) iterations of the walk. From the above results we can see that the amplitude, \( a_{000000} \), of the quantum circuit’s workspace state to be |000000\rangle is \( a_{000000} = -1/4 \) and the probability for the workspace state is \( p_{000000} = 1/16 = 0.0625 \). But for the probability of the state space position |000\rangle we have to take into account all the workspace states that will lead to a collapse on position |000\rangle. Thus, using the above equation, we can compute

\[
p_{000} = \sum_{j \in \{0, 1\}^{6 \mod \text{mod } 8 = 0}} p_j
\]

or in simplified decimal notation

\[
p_{0} = \sum_{j \in \{0, 64\} \mod \text{mod } 8 = 0} p_j
\]

As an example, we can use the above equation to find the probability of the particle to be in the position |000\rangle after \( t = 4 \) iterations of the walk. From the above results we can see that the amplitude, \( a_{000000} \), of the quantum circuit’s workspace state to be |000000\rangle is \( a_{000000} = -1/4 \) and the probability for the workspace state is \( p_{000000} = 1/16 = 0.0625 \). But for the probability of the state space position |000\rangle we have to take into account all the workspace states that will lead to a collapse on position |000\rangle. Thus, using the above equation, we can compute

\[
p_{000} = \sum_{j \in \{0, 64\} \mod \text{mod } 8 = 0} p_j
\]

or in simplified decimal notation

\[
p_{0} = \sum_{j \in \{0, 64\} \mod \text{mod } 8 = 0} p_j
\]

As an example, we can use the above equation to find the probability of the particle to be in the position |000\rangle after \( t = 4 \) iterations of the walk. From the above results we can see that the amplitude, \( a_{000000} \), of the quantum circuit’s workspace state to be |000000\rangle is \( a_{000000} = -1/4 \) and the probability for the workspace state is \( p_{000000} = 1/16 = 0.0625 \). But for the probability of the state space position |000\rangle we have to take into account all the workspace states that will lead to a collapse on position |000\rangle. Thus, using the above equation, we can compute

\[
p_{000} = \sum_{j \in \{0, 64\} \mod \text{mod } 8 = 0} p_j
\]

or in simplified decimal notation

\[
p_{0} = \sum_{j \in \{0, 64\} \mod \text{mod } 8 = 0} p_j
\]
We can easily see that this probability is approximately the one we get from running the experiment, \( p_{(0)} = 0.125 \), after a large number of shots (preferably greater than 1000 shots). Thus, we can calculate the probabilities for each state space as

\[
\begin{align*}
p_{(0)} &= 0.125 \\
p_{(2)} &= 0.125 \\
p_{(4)} &= 0.125 \\
p_{(6)} &= 0.025,
\end{align*}
\]

which of course adds up to 1. The probabilities of the rest of the positions is 0, due to the modularity properly.

The intuition behind this is that every probability of the state space of the walk, i.e the quantum states that represent the nodes of the walk (\([000], \ldots, [111]\) in this case), each have probability equal to the sum of the probabilities of the states of the work space (including ancillas and coin) whose binary representation modulo \( N \) equals the specific state space.

VIII. THEORETICAL VARIANCE OF THE QUANTUM WALK

In this section, we will calculate the variance of the quantum Hadamard walk on a cycle, as it was run on the simulator and the quantum computer. For convenience, the work will be done on an \( N \)-cycle with \( N = 8 \), using \( n = 3 \) qubit states, namely \([000], \ldots, [111]\). The circuit for this quantum walk is presented in the above analysis. Our theoretical analysis follows the steps used by [15].

First we will consider a theoretical approach for finding the variance of the quantum walk. The main characteristic of a quantum walk is its asymmetry, compared to the classical walk. This asymmetry derives from the effects of interference during the steps of the walk (interference arises from the fact that the amplitudes of the quantum states can be negative and/or imaginary numbers). The analysis of the quantum walk provided this far is not concerned with the symmetry of its states, but it is concerned with the bias that arises from the initial state of the walk.

A quantum walk on a particle will show different behavior depending on whether the initial spin of the particle is up (|↑⟩) or down (|↓⟩). In order to remove this bias, the quantum coin is initialized in an equal superposition of the two states using a Hadamard gate. Thus, depending on the initial state of the particle, the walk will be initialized as

\[
\begin{align*}H|L\rangle &= \frac{1}{\sqrt{2}}(|L\rangle + |R\rangle) \\
H|R\rangle &= \frac{1}{\sqrt{2}}(|L\rangle - |R\rangle)
\end{align*}
\]

Note here how the amplitudes depend on each initial state, i.e for |↑⟩ we have addition and for |↓⟩ we have subtraction. This difference in the initial state amplitudes comprises what we call chirality, which describes the predisposition of the quantum walk to interference.

The two component vector of the amplitudes of the particle being at point \( n \) in time \( t \) is

\[
\Psi(n, t) = \begin{pmatrix} \psi_L(n, t) \\ \psi_R(n, t) \end{pmatrix}
\]

with the upper component \( \psi_L(n, t) \) having chirality Left and the lower component \( \psi_R(n, t) \) having chirality Right. The dynamics of the wave function \( \Psi \) are given by the transformation

\[
\begin{align*}
\Psi(n, t + 1) &= \begin{bmatrix} 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \Psi(n - 1, t) + \\
&\begin{bmatrix} 0 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \Psi(n + 1, t)
\end{align*}
\]

\[
M_+\Psi(n - 1, t) + M_-\Psi(n + 1, t)
\]

This transformation is unitary, since it is a composition of a unitary operator (namely \( H \)) and a reversible move/step to the left or to the right. We assume that the particle starts at the origin with chirality left and so the initial conditions are

\[
\Psi(0, 0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}
\]

which of course implies

\[
\Psi(n, 0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \text{ if } n \neq 0.
\]

A. Fourier Analysis of the Hadamard Walk

One of the better ways to map a continuous function to a discrete space is to use Fourier Transforms (F.T). The spatial F.T \( \tilde{\Psi}(k, t) \) for \( k \in [-\pi, \pi] \) of the wave function \( \Psi(n, t) \) over \( Z \) is given by

\[
\tilde{\Psi}(k, t) = \sum_n \Psi(n, t)e^{ikn}
\]

where \( \Psi(n, t) : Z \mapsto C \) maps from continuous space \( Z \) to discrete space \( C \) and is a complex valued function over the integers, with its F.T being \( \Psi(k, t) : [-\pi, \pi] \mapsto C \) with \([-\pi, \pi]\) being continuous.

From the dynamics of \( \Psi \) we may deduce the following about \( \tilde{\Psi} \):

\[
\tilde{\Psi}(k, t + 1) = \sum_n \left( M_+\Psi(n - 1, t) + M_-\Psi(n + 1, t) \right)e^{ikn}
\]

\[
= e^{ik}M_+\sum_n \Psi(n - 1, t)e^{ik(n-1)} + e^{-ik}M_-\sum_n \Psi(n + 1, t)e^{ik(n+1)}
\]

\[
= \left( e^{ik}M_+ + e^{-ik}M_- \right)\tilde{\Psi}(k, t)
\]

where it is important to prove the following

\[
\tilde{\Psi}(k - 1, t) = \sum_n \left( M_+\Psi(n - 1, t) + M_-\Psi(n + 1, t) \right)e^{ik(n-1)}
\]

\[
= \sum_n \Psi(n - 1, t)e^{ikn}e^{-ik}
\]

\[
= e^{-ik}\sum_n \Psi(n - 1, t)e^{ikn}
\]

\[
= e^{-ik}e^{ik}\sum_n \Psi(n - 1, t)e^{ik(n-1)}
\]

\[
= \sum_n \Psi(n - 1, t)e^{ikn} = \bar{\tilde{\Psi}}(k, t)
\]
for the transition between the last two equalities. Thus we have that
\[ \tilde{\Psi}(k, t + 1) = M_k \tilde{\Psi}(k, t) \]  
(14)
where
\[ M_k = e^{ikM_+} + e^{-ik}M_- \]
\[ = \frac{1}{\sqrt{2}} \begin{bmatrix} e^{-ik} & e^{-ik} \\ e^{ik} & -e^{ik} \end{bmatrix} \]  
(15)
Note that we can also write \( M_k = A_k U^\dagger \), where \( A_k \) is a diagonal matrix and \( U^\dagger \) is the transpose of the unitary that acts on the chirality state (in our case \( U = H \)). Thus, we know that \( M_k \) is unitary and the recurrence in the Fourier space takes the form
\[ \tilde{\Psi}(k, t) = M_k \tilde{\Psi}(k, t) = M_k^t \tilde{\Psi}(k, 0) \]  
(16)
We can calculate \( M_k^t \) by diagonalising the matrix \( M_k \) which is readily done, since it is a \( 2 \times 2 \) matrix.

Next, we will talk about the eigendecomposition of \( M_k \). If \( M_k \) has eigenvectors \( (|\Phi_1^k \rangle, |\Phi_2^k \rangle) \) and eigenvalues \( (\lambda_1^k, \lambda_2^k) \), we can write
\[ M_k = \lambda_1^k |\Phi_1^k \rangle \langle \Phi_1^k | + \lambda_2^k |\Phi_2^k \rangle \langle \Phi_2^k |. \]
and then obtain the evolution matrix as
\[ M_k^t = (\lambda_1^k)^t |\Phi_1^k \rangle \langle \Phi_1^k | + (\lambda_2^k)^t |\Phi_2^k \rangle \langle \Phi_2^k |. \]  
(17)
We can find the eigenvalues of \( M_k \) as \( \lambda_1^k = e^{-i\omega_k} \) and \( \lambda_2^k = e^{i(\pi - \omega_k)} \), where \( \omega_k \) is the angle in \([-\pi/2, \pi/2]\) such that \( \sin(\omega_k) = \frac{1}{\sqrt{2}} \). The corresponding eigenvectors are:
\[ \Phi_1^k = \frac{1}{\sqrt{2N(k)}} \begin{bmatrix} e^{-ik} \\ e^{ik} \end{bmatrix} \]
\[ \Phi_2^k = \frac{1}{\sqrt{2N(\pi - k)}} \begin{bmatrix} -\sqrt{2}e^{-i\omega_k} + e^{ik} \\ e^{ik} \end{bmatrix} \]  
(18)
where \( N(k) = (1 + \cos^2 k) + \cos k \sqrt{1 + \cos^2 k} \).

Considering the evolution of a particle that starts with chirality \( \text{Left} \) and in the Fourier basis \( \tilde{\Psi}(k, 0) = |0\rangle |k\rangle \), we can find the two components of the wave function of equation \( \text{[12]} \) at time \( t \), given by
\[ \tilde{\psi}_L(k, t) = \frac{1}{2} \left( 1 + \frac{\cos k}{\sqrt{1 + \cos^2 k}} \right) e^{-i\omega_k t} + \frac{(-1)^t}{2} \left( 1 - \frac{\cos k}{\sqrt{1 + \cos^2 k}} \right) e^{i\omega_k t} \]  
(19)
\[ \tilde{\psi}_R(k, t) = \frac{ie^{ik}}{2 \sqrt{1 + \cos^2 k}} \left( e^{-i\omega_k t} - (-1)^t e^{i\omega_k t} \right) \]

Now, we can invert the F.T to return to the real basis. This inversion is given by
\[ \Psi(n, t) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \tilde{\psi}(k, t) e^{-i kn} dk \]  
(20)
Thus, we can now describe the components of the wave function as:
\[ \psi_L(n, t) = \frac{1}{2} \left( 1 + \frac{\cos k}{\sqrt{1 + \cos^2 k}} \right) e^{-i(\omega_k t + kn)} \]
\[ \psi_R(n, t) = \frac{1}{2} \left( 1 - \frac{\cos k}{\sqrt{1 + \cos^2 k}} \right) e^{-i(\omega_k t + kn)} \]  
(21)
The above theory is derived for a particle that starts in chirality left or \( |\downarrow\rangle \). Now we want to account for chirality right or \( |\uparrow\rangle \). The operator \( M_k^t \) acting on the chirality state \( |c\rangle = \{\uparrow, \downarrow\} \) will now be written as
\[ M_k^t |c\rangle = (|\lambda_1^t\rangle \langle \phi^t_L | |\phi^t_R \rangle + (|\lambda_2^t\rangle \langle \phi^t_L | |\phi^t_R \rangle) |\phi^t_R \rangle \]

The basis in F.T can be expressed as
\[ |\Psi_{n,t}^c \rangle = U^t |\psi_0^c \rangle = U^t(|0\rangle \otimes |c\rangle) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \tilde{\psi}(k, t) \otimes M_k |c\rangle dk \]
and we can get
\[ |\psi_t^c \rangle = \sum_n |n\rangle \otimes [A^c_t(n) |\uparrow\rangle + B^c_t(n) |\downarrow\rangle] \]
with the following equalities:
\[ A^c_t(n) = \frac{1 + (-1)^{t+n}}{2} \left[ \alpha^c(n) + \beta^c(n) \right] \]
\[ A^c_t(n) = \frac{1 + (-1)^{t+n}}{2} \left[ \beta^c(n) - \gamma^c(n) \right] \]
\[ B^c_t(n) = \frac{1 + (-1)^{t+n}}{2} \left[ \beta^c(n) + \gamma^c(n) \right] \]
and
\[ \alpha^c(n) = \int_{-\pi}^{\pi} \frac{dk}{2\pi} \frac{e^{i(kn - \omega_k)}}{\sqrt{1 + \cos^2 k}} \]
\[ \beta^c(n) = \int_{-\pi}^{\pi} \frac{dk}{2\pi} \frac{\cos k}{\sqrt{1 + \cos^2 k}} e^{i(kn - \omega_k)} \]
\[ \gamma^c(n) = \int_{-\pi}^{\pi} \frac{dk}{2\pi} \frac{\sin k}{\sqrt{1 + \cos^2 k}} e^{i(kn - \omega_k)} \]  
(23)

B. Derivation of the Variance

To find the variance or the standard deviation of the walk, we want to generalise the initial state in a similar fashion to \( \text{[17]} \). Thus, we can start with the state:
\[ |\psi_0 \rangle = |0\rangle \otimes \left[ \sqrt{q} |\uparrow\rangle + \sqrt{1-q} e^{i\sigma} |\downarrow\rangle \right] \]  
(24)
with \( q \in [0, 1] \) and \( \sigma \in \mathbb{R} \), which is a completely general state. The state after \( t \) steps will be:
\[ |\psi_t \rangle = U^t |\psi_0 \rangle = \sum_n |n\rangle \otimes \left\{ \begin{array}{l} \sqrt{q} A^t(n) + \sqrt{1-q} e^{i\sigma} A^t(n) |\uparrow\rangle \\
\sqrt{q} B^t(n) + \sqrt{1-q} e^{i\sigma} B^t(n) |\downarrow\rangle \end{array} \right. \]
The above can be further simplified if we take into account the modularity property of the walk.
The probability to be at position \( n \) after \( t \) steps is:
\[ p^t(n) = \left\{ \begin{array}{l} \sqrt{q} A^t(n) + \sqrt{1-q} e^{i\sigma} A^t(n) \\
\sqrt{q} B^t(n) + \sqrt{1-q} e^{i\sigma} B^t(n) \end{array} \right)^2 \]  
(25)
The integral parts of the matrices \( A_n^c \) etc are mostly concentrated in the interval \([-t/\sqrt{2}, t/\sqrt{2}]\) and they quickly decrease beyond the bounding values of this interval. This can be shown by the method of stationary phase (described in Appendix\(^B\)). Thus, doing the correct approximations we can find the probability \( P^t(n) \) of equation \((27)\) as an oscillation around the function

\[
P^t(n) = \frac{2t}{\pi(t-n)/\sqrt{t^2-2n^2}}
\]

where we dropped the vanishing part coming from the modularity property (for even number of steps, \( t \), the walk cannot be in even position, \( n \)).

The function \( P^t(n) \) allows us to approximately evaluate the averages of position \( n \)-dependent functions in the \( t \)-th step of the walk as

\[
\langle f^t(n) \rangle \simeq \frac{1}{2} \int_{-\sqrt{2}/2}^{\sqrt{2}/2} f^t(n) P^t(n) dn
\]

Thus, using equation \((26)\), we can find

\[
\langle n \rangle \simeq \frac{1}{2} \int_{-\sqrt{2}/2}^{\sqrt{2}/2} n \frac{2t}{\pi(t-n)/\sqrt{t^2-2n^2}} dn = -t \sqrt{2} - 1
\]

\[
\langle n^2 \rangle \simeq \int_{-\sqrt{2}/2}^{\sqrt{2}/2} n^2 \frac{2t}{\pi(t-n)/\sqrt{t^2-2n^2}} dn = t^2 \sqrt{2} - 1
\]

and finally, using equations \((27)\), we can find the variance of the quantum walk as

\[
\sigma^2 = \langle n^2 \rangle - \langle n \rangle^2 = t^2 \left( \sqrt{2} - 1 \right) - t^2 \left( \sqrt{2} - 1 \right)^2
\]

\[
= t^2 \left( \sqrt{2} - 1 \right) \left[ 1 - \frac{\sqrt{2} - 1}{\sqrt{2}} \right]
\]

Thus, we have proved that the variance is \( \sigma^2 \sim t^2 \) or otherwise the standard deviation is \( \sigma \sim t \). This means that the deviation of the quantum walk grows near-linearly with time and thus the quantum walk is quadratically faster than the classical random walk (where the variance grows as \( \sigma \sim \sqrt{t} \)).

Using the above analysis, we can compute the variance of the Hadamard quantum walk on an \( N \)-cycle using \((28)\). For example, in our 8-cycle experiment (\( N = 8 \)) the variance after a number of steps \( t = 15 \) and following equation \((28)\) will be

\[
\sigma^2 \simeq 15^2 \left( \frac{\sqrt{2} - 1}{\sqrt{2}} \right) \left[ 1 - \left( \frac{\sqrt{2} - 1}{\sqrt{2}} \right) \right] \approx 46.575.
\]

and standard deviation

\[
\sigma \simeq \sqrt{46.575} = 6.824
\]

C. Simulation vs Theoretical Variance

Thus, we can use the implementation of the quantum walk to calculate the variance, as it derives from the simulated walk. For this we use the quantum state that results from the simulation of the quantum walk on an \( N \)-cycle after arbitrary number of coin flips, \( t \). Also, this simulation variance, annotated \( \sigma^2_n \), can be compared to the theoretical variance, \( \sigma^2 \) and the variance of a classical random walk, \( \sigma^2_c \).

A visual representation of the three aforementioned variances for the quantum and classical walks is shown in Figure \(16\). We can easily see that the simulation variance follows a quadratic tendency, which proves that it is faster than the classical variance. Additionally, we can see that the theoretical variance is very close to the simulation variance and, also, both remain quadratically faster than the classical random walk.

![Figure 16: The various variances of the quantum walk.](image)

**FIG. 16:** The various variances of the quantum walk.

**IX. BENCHMARKING QUANTUM ERROR USING HELLINGER DISTANCE**

In information theory, the cross entropy between two probability distributions \( A = (a_1, \ldots, a_k) \) and \( B = (b_1, \ldots, b_k) \) over the same underlying set of events measures the average number of bits needed to identify an event drawn from the set if a coding scheme used for the set is optimized for an estimated probability distribution \( B \), rather than the true distribution \( A \).

The cross entropy for the distributions \( A \) and \( B \) over a given set is defined as follows

\[
H(A, B) = \mathbb{E}_A [-\log B].
\]

The definition may be formulated using the Kullback-Leibler divergence \( D_{KL}(A||B) \) of \( B \) from \( A \) as

\[
H(A, B) = H(A) + D_{KL}(A||B)
\]

where \( H(A) \) is the entropy of \( A \) defined as \( H(A) = -\sum_{i=1}^k P(a_i) \log P(a_i) \), where \( P \) is the probability mass function of \( A \).

The Kullback-Leibler divergence is a measure of how one probability distribution is different from a second, reference probability distribution. For discrete probability distributions

\[
D_{KL}(A||B) = \sum_i P_i \log \frac{P_i}{Q_i}
\]
A and $B$ with the same support, $\mathcal{X}$, the Kullback-Leibler divergence is defined as

$$D_{KL}(A||B) = -\sum_{x \in \mathcal{X}} A(x) \log \left( \frac{B(x)}{A(x)} \right)$$

For discrete probability distributions $A$ and $B$ with the same support, $\mathcal{X}$, the cross entropy can be calculated as

$$H(A, B) = -\sum_{x \in \mathcal{X}} a(x) \log b(x). \quad (29)$$

In other words, it is the expectation of the logarithmic difference between the probabilities $A$ and $B$.

Essentially what this means is that we can use cross entropy to quantify the difference between two distributions. Our quantum circuits run on IBMQ’s quantum computer (with errors) and on the quantum simulator. Thus, we can generate probability distributions for any quantum walk after it is run on the quantum computer and then benchmark our circuit by comparing it to the theoretical probability distribution as produced by the simulator.

The main problem we come across here is that the two probability distributions do not have the same support, as that is required by the cross entropy method. More precisely, since the quantum computer violates the modularity property, we encounter some states that have probability 0 to occur on the simulator. Mathematically this means that the logarithm in equation (29) will become undefined.

One way to combat this, as will be explained later, is to use a different way to calculate the difference between the two distributions. For the purposes of this we use the Hellinger distance. But first, we briefly discuss about the distributions we use for the benchmarking.

A. The Distributions

The Hellinger distance is calculated for the 8-cycle experiment described in the above sections. In that experiment we have a 3 qubit state space and the quantum walk is run for 15 coin flips and is initialized in state $|3\rangle$. The resulting probability distributions are shown in figure (11) for the simulation and (12) for the quantum computer.

In addition to these two distributions, we will use a discrete uniform probability distribution, $R = (r_0, \ldots, r_7)$, of states on the 8-cycle. This is the distribution where all the discrete values (i.e the quantum states of the particle or the nodes of the cycle) have equal probability to be observed. The probability for each state is found simply as $r_i(x) = 1/8$, $\forall i \in [0, 7]$. Figure (17) shows the uniform distribution $R$.

For convenience, we define as $P = (p_0, \ldots, p_7)$ the discrete probability distribution of the theoretical (or simulation) output of the quantum walk. This is taken from the output of the simulations done on IBMQ’s quantum simulator. Finally, we define as $Q = (q_0, \ldots, q_7)$, the probability distribution of the output of the quantum computer.

B. Hellinger Distance

The Hellinger distance is used to quantify the difference between two distributions without the need for them to have the same support. For our discrete quantum walk on a one-dimensional line we are interested in the comparison between discrete distributions. Thus, for two generic probability distributions $A = (a_1, \ldots, a_k)$ and $B = (b_1, \ldots, b_k)$, the Hellinger distance $\eta_H(A, B)$ can be defined as:

$$\eta_H(A, B) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{k} (\sqrt{a_i} - \sqrt{b_i})^2} \quad (30)$$

which is closely related to the Euclidean norm of the difference of the square root vectors, i.e

$$\eta_H(A, B) = \frac{1}{\sqrt{2}} \| \sqrt{A} - \sqrt{B} \|
$$

and also

$$1 - \eta_H^2(A, B) = \sum_{i=1}^{k} \sqrt{a_i b_i}.$$

With the theoretical part out of the way we can now use the above equations to calculate the Hellinger distance between the uniform probability distribution for our system, $R$, the theoretical (or simulation) probability distribution, $P$, and the quantum computer’s probability distribution, $Q$, as they were defined in the above section IX A.

We calculate the Hellinger distance for every one of the above pairs of distributions. The results show as follows

$$\eta_H(R, P) = 0.582$$

$$\eta_H(R, Q) = 0.109$$

$$\eta_H(P, Q) = 0.559$$

To translate these quantities we need to consider that the Hellinger distance takes values between 0 and 1. A distance of 0 means that the two probability distributions are identical (or one and the same). Any value of $0 < \eta_H \leq 1$ quantifies the difference between the two distributions. Thus, with that in mind, we can see that the uniform probability distribution is quite close to the output of the quantum computer, as their distance is of the measure of 0.109. On the other hand, the
uniform and the simulation distributions differ much more. The same thing goes for the simulation and quantum computer distributions. This, essentially, means that the quantum computer’s output differ a lot from the theoretical output that we expect to see and, in the end, proves and quantifies the error of the quantum machine.

X. CONCLUSIONS

We carry out an experiment where we implement the quantum walk of a particle. In order to cancel the bias of the walk, we use a balanced coin, or otherwise, a Hadamard coin, which, in every coin flip, puts the register in an equal superposition of its possible one-qubit states. According to the outcome of each coin flip, the particle progresses to the next possible position by either increasing or decreasing its position state. This can be done using increment and decrement function, as described in [9].

The coin operator that implements the coin flip is a simple Hadamard operator and the corresponding quantum gate. Thus, the main focus of the work is the increment and decrement function. There is two approaches used throughout this paper: the generalised CNOT, as used by [9], and our rotational approach.

Regarding the former, using generalised CNOTs keeps the implementation very simple and the circuit smaller. But it introduces the caveat that, in order to implement the gates with more than 2 control qubits, we need to use an ancilla register. Furthermore, the number of ancilla qubits (otherwise the size of the ancilla register) increases linearly with the number of control qubits, i.e for n control qubits we need n − 1 ancilla qubits. This approach quickly leads to a very large workspace which, in turn, limits the capabilities of our experiment. For example, local simulations struggle to simulate a workspace of more than 20 qubits. Additionally, IBM’s quantum computer that is available to us is a 16 qubit machine, with 15 working qubits.

On the other hand, the rotational approach deals with the aforementioned limitation by reducing the number of ancilla qubits needed. Using rotation gates, like U3 and Rz already implemented on IBMQ’s backends, allows us to rotate the state space around the basis states, eliminating the need for intermediate computations and rendering the quantum ancilla register obsolete. This allows us to experiment with a much larger state space for our quantum walk. The disadvantage of the rotational approach is the size of the resulting circuit. The rotational gates used are not fundamental gates, like the inverters and Toffoli gates used in the first approach. This renders the quantum circuit slightly slower.

Further experiments done with quantum walks of particles include walks on 16, 32 and 64 cycles. As expected, the resulted circuits are much larger every time the state space increases, rendering the simulations and the experiments slower. Regarding the simulation of the quantum walk, the results support the theoretical properties, i.e the asymmetry and modularity of the quantum walk. Opposite to that, the experiments run on the quantum computer for the 16 and 32 state space cases (4 and 5 qubits) seem to violate this property and measure the particle in all possible positions. The 6 qubit experiment fails to run on the quantum computer for a range of coin flips, t.

Furthermore, in the context of quantum walks, we use the Hellinger distance to quantify the difference between the output of the quantum simulator and the real quantum machine. This is a very convenient way of showing the scale of the error when the walks are run on the quantum computer.

Additionally, we follow the Fourier analysis approach of the quantum walk in order to find the variance of the quantum walk. As shown in section VIII the quantum propagates approximately as ς = t, whereas the classical random walk propagates with ς = \sqrt{t}. Thus, we get a quadratic speedup to the particles walk. The near-quadratic tendency of the variance found mathematically agrees with the variance found in the simulations.

Finally we can use the method of cross entropy benchmarking via Hellinger distance in order to quantify the difference between the probability distributions that occur as output of the simulator and the quantum computer. According to this method, we see that the Hellinger distance is quite significant, showing that, when the circuit runs on the quantum machine, it produces a large number of errors. The main reason for this is the violation of the modularity property.

XI. FUTURE WORK

The main research area we are going to focus in following work mainly aims to address the erroneous evolution of the quantum walk on the real machine. As explained in section VII the collapse of the modularity property causes a number of issues with performing quantum walks and calculating the variance and the mixing time of the quantum walk. This will allow us to collect more data on how the rotational quantum walks behave on real machines.

Another idea evolves around modeling the error within the simulation and then, using, for example, Hellinger distance, calculating the difference between the erroneous simulation distribution and the quantum machine’s distribution. This way, quantum walks can also be used to benchmark the quantum error rates of various machines, as we expect similar behavior (in terms of probability distributions) of a single quantum walk.

[1] Dorit Aharonov, Andris Ambainis, Julia Kempe, and Umesh Vazirani. Quantum walks on graphs. In Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing, STOC ’01, pages 50–59, New York, NY, USA, 2001. ACM.
[2] Andris. Ambainis. Quantum walk algorithm for element distinctness. SIAM Journal on Computing, 37(1):210–239, 2007.
[3] Andris Ambainis, Eric Bach, Ashwin Nayak, Ashvin Vishwanath, and John Watrous. One-dimensional quantum walks. In Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing, STOC ’01, pages 37–49, New York, NY, USA, 2001. ACM.
[4] Adriano Barenco, Charles H. Bennett, Richard Cleve,
David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52:3457–3467, Nov 1995.

[5] Andrew M. Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, and Daniel A. Spielman. Exponential algorithmic speedup by a quantum walk. *Proceedings of the thirty-fifth ACM symposium on Theory of Computing - STOC '03*, 2003.

[6] Andrew M. Childs and Jeffrey Goldstone. Spatial search by quantum walk. *Physical Review A*, 70(2), Aug 2004.

[7] Anirban Narayan Chowdhury and Rolando D. Somma. Quantum algorithms for gibbs sampling and hitting-time estimation. *Quantum Info. Comput.*, 17(1-2):41–64, February 2017.

[8] David A. Cieslak, T. Ryan Hoens, Nitesh V. Chawla, and W. Philip Keegelmeier. Hellinger distance decision trees are robust and skew-insensitive. *Data Min. Knowl. Discov.*, 24(1):136–158, January 2012.

[9] B. L. Douglas and J. B. Wang. Efficient quantum circuit implementation of quantum walks, 2007.

[10] Julia Kempe. Discrete quantum walks hit exponentially faster. *Probability Theory and Related Fields*, 133(2):215–235, Oct 2005.

[11] Hari Krovi. Symmetry in quantum walks. 2007.

[12] Frederic Magniez, Ashwin Nayak, Jeremie Roland, and Miklos Santha. Search via quantum walk. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 575–584, New York, NY, USA, 2007. ACM.

[13] Ashley Montanaro. Quantum speedup of monte carlo methods. 2015.

[14] Ashley Montanaro. Quantum-walk speedup of backtracking algorithms. *Theory of Computing*, 14(15):1–24, 2018.

[15] Ashwin Nayak and Ashvin Vishwanath. Quantum walk on the line, 2000.

[16] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011.

[17] Daniel Reitzner, Daniel Nagaj, and Vladimr Buek. Quantum walks. *Acta Physica Slovaca Reviews and Tutorials*, 61(6):603–725, 2011.

[18] Mario Szegedy. Quantum speed-up of markov chain based algorithms. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '04, pages 32–41, Washington, DC, USA, 2004. IEEE Computer Society.

[19] K. Temme, T. J. Osborne, K. G. Vollbrecht, D. Poulin, and F. Verstraete. Quantum metropolis sampling. *Nature*, 471(736):87–90, 2011.

**Appendix A: Generalised Toffoli Gate for n Control Qubits**

A Toffoli gate, or otherwise a Controlled-Controlled-NOT (CNOT) gate, is the inverter gate with two control qubits. The Toffoli gate represents the inverter with the maximum number of control qubits that is implemented on IBMQ’s quantum simulator and machines. This is very problematic as, in many circumstances, we need to control an inversion, or in extent any unitary operation, with more than two qubits.

This problem occurs during this experiment as well, as demonstrated during in increment and decrement circuits.

Thankfully, there is a solution to this conundrum, given many years ago by Nielsen and Chuang in their textbook on Quantum Computation and Quantum Information [10]. The idea is that, in order to control an operation with more than two qubits, we need to take the addition modulo 2 (or AND operation) of the first two control qubits and store the result in an ancilla qubit. Then, we take the AND operation between this ancilla qubit and the next control qubit. We continue in this fashion until all the control qubits have been accounted for. The last ancilla register used will contain the result of all the AND operations between the control qubits (i.e iff all the control qubits are |1⟩ then the resulting ancilla will be |1⟩). Finally, the last operation is a simple controlled-NOT (CNOT) gate between the last ancilla qubit and the target qubit.

In the case of an inverter gate, if the last ancilla qubit is |1⟩ then the target qubit will be inverted. Otherwise (i.e if one or more control qubits where |0⟩), then nothing will happen to the target qubit. The resulting circuit is what we call a generalised Toffoli or generalised CNOT gate.

![FIG. 18: generalised Toffoli gate with 3 control qubits (q0 to q2), 2 ancilla qubits (anc0, anc1) and 1 target qubit (tgt0).](image)

A very elegant visualization of this solution is shown in Figure 18. Note here, it is extremely important that, after every operation, there has to come an uncomputation of the qubits, so that all the ancilla qubits can return to their initial state. To achieve this we just need to apply the same operations again, after the computation has finished and the target qubit has either been flipped or stayed the same. Of course, this uncomputation is not applied on the target qubit.

**Appendix B: n Qubits Toffoli Gate Using Rotations**

The following analysis tries to provide an insight to the gates used throughout the rotational quantum circuit that implements a generalised CNOT gate. The gates used are mainly pulled from IBMQ’s backend and development kit.

First of all, we explore the $R_z$ gate, which implements the operator

$$R_z(\phi) = \begin{pmatrix} e^{i\phi/2} & 0 \\ 0 & e^{-i\phi/2} \end{pmatrix}$$

Thus, we can pull the unitary output for different values of $\phi$. 

$$R_z(\phi) |0\rangle = e^{i\phi/2} |0\rangle$$

$$R_z(\phi) |1\rangle = e^{-i\phi/2} |1\rangle$$

Thus, we can pull the unitary output for different values of $\phi$. 

$$R_z(\phi) |\psi\rangle = e^{i\phi/2} |\psi\rangle$$

Therefore, we can use the $R_z$ gate to implement a generalised Toffoli gate with $n$ control qubits.
\( \phi \). For \( \phi = \pi/2 \), as used in our case, the operator becomes
\[
R_x(\pi/2) = \begin{pmatrix}
\sqrt{2}/2 & \sqrt{2}/2 & 0 & 0 \\
\sqrt{2}/2 & -\sqrt{2}/2 & 0 & 0 \\
0 & 0 & \sqrt{2}/2 & \sqrt{2}/2 \\
0 & 0 & \sqrt{2}/2 & -\sqrt{2}/2
\end{pmatrix}.
\]

Next is the \( U_3 \) gate, which implements the operator
\[
U_3(\theta, \phi, \lambda) = \begin{pmatrix}
\cos(\theta/2) & -e^{i\lambda} \sin(\theta/2) \\
-e^{i\phi} \sin(\theta/2) & e^{i\lambda+\phi} \cos(\theta/2)
\end{pmatrix}
\]
The output for our case, where \( \theta = \pi, \phi = \lambda = 0 \), becomes
\[
R_y(\pi) = U_3(\pi, 0, 0) = \begin{pmatrix}
\sqrt{2}/2 & \sqrt{2}/2 \\
\sqrt{2}/2 & -\sqrt{2}/2
\end{pmatrix}
\]

Finally we have the phase gate \( \Phi(\delta) \) which is used to take the special matrix rotation to the general unitary case. This implements the operator
\[
\Phi(\delta) = \begin{pmatrix}
e^{-i\delta} & 0 \\
0 & e^{i\delta}
\end{pmatrix},
\]
where for our case of \( \delta = \pi/2 \), we get
\[
\Phi(\pi/2) = \begin{pmatrix}
i & 0 \\
0 & i
\end{pmatrix}.
\]

The analogous matrices for the other attribute values can be found via the aforementioned equations. Namely, for our quantum circuit we have
\[
R_x(-\pi/2) = \begin{pmatrix}
\sqrt{2}/2 & -\sqrt{2}/2 & 0 & 0 \\
-\sqrt{2}/2 & \sqrt{2}/2 & 0 & 0 \\
0 & 0 & \sqrt{2}/2 & -\sqrt{2}/2 \\
0 & 0 & -\sqrt{2}/2 & \sqrt{2}/2
\end{pmatrix}
\]
and
\[
R_y(-\pi/2) = U_3(-\pi/2, 0, 0) = \begin{pmatrix}
\sqrt{2}/2 & \sqrt{2}/2 \\
\sqrt{2}/2 & -\sqrt{2}/2
\end{pmatrix}.
\]

We can now find the matrix \( 2 \times 2 \) representation of the unitary operator, \( W \), that represents the rotational quantum circuit as
\[
W = \Phi(\delta) R_x(\alpha) R_y(\theta/2) \sigma_x R_y(-\theta/2) R_x \left( -\frac{\alpha + \beta}{2} \right)
\]
\[
= \Phi(\pi/2) R_x(\pi/2) R_y(\pi/2) \sigma_x R_y(-\pi/2) R_x(0)
\]
\[
= \begin{pmatrix}
i & 0 \\
0 & i
\end{pmatrix} \begin{pmatrix}
\sqrt{2}/2 & \sqrt{2}/2 & 0 & 0 \\
\sqrt{2}/2 & -\sqrt{2}/2 & 0 & 0 \\
0 & 0 & \sqrt{2}/2 & \sqrt{2}/2 \\
0 & 0 & \sqrt{2}/2 & -\sqrt{2}/2
\end{pmatrix} \begin{pmatrix}
\sqrt{2}/2 & \sqrt{2}/2 \\
\sqrt{2}/2 & -\sqrt{2}/2 \\
0 & 0 \\
0 & 0
\end{pmatrix}
\]
\[
\times \begin{pmatrix}
0 & 1 \\
1 & 0
\end{pmatrix} \begin{pmatrix}
\sqrt{2}/2 & \sqrt{2}/2 \\
\sqrt{2}/2 & -\sqrt{2}/2 \\
0 & 0 \\
0 & 0
\end{pmatrix} \begin{pmatrix}
0 & 1 \\
1 & 0
\end{pmatrix}
\]
\[
\times \begin{pmatrix}
\sqrt{2}/2 & \sqrt{2}/2 \\
\sqrt{2}/2 & -\sqrt{2}/2 \\
0 & 0 \\
0 & 0
\end{pmatrix} = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & e^{i\phi/2} & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & e^{-i\phi/2}
\end{pmatrix}.
\]

We again use \( \pm \phi = \pi \).

The matrix multiplication in this case results to
\[
W_c = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

This is quite different from the generalised form of unitaries that derives from our theory. The reason for this difference is that Qiskit uses an inverse endian representation in its circuit. In other words, the most significant qubit is the left-most qubit on the circuit representation, as opposed to it being the right one in the usual mathematical representation used in bibliography and quantum mechanics literature. For example, the mathematical representation of the basis state \( |1\rangle \) in Qiskit is \( |001\rangle \) for a three qubit register, whereas in the literature we find it as \( |100\rangle \).

This difference introduces numerous complications on designing and implementing the circuit. Essentially, when we implement the generalised \texttt{CNOT} gate with rotations, the different way the matrices are implemented actually breaks down the quantum circuit. On the other hand, when we inverse the logic of the qubits and treat them the opposite endian way, the operators are implemented the correct way and the circuit actually works and produces the expected results.

This means that we can now, using Lemma\textsuperscript{[III.3]} generalize the quantum walk for any number of states we want, as long as the number of qubits can be simulated from the machine or supported by the quantum computer. The number of ancilla qubits will no longer be a limit towards the state space of the implementation.

On the downside, the number of gates used for the circuit is getting bigger with the number of control qubits used. Thus, the complexity of the circuit is rising. For example, for 3 control qubits, we need 5 \texttt{CNOT} gates to implement the generalised
CNOT gate, whereas for the rotation we need 6. For 4 control qubits we need 7 CNOT and 16 rotation gates. For 5 control qubits we need 9 CNOT and 36 rotation gates etc.

For easy comparison we can count the gates for a 5 qubit state space (4 control qubits) of the Toffoli gate in both the generalised ‘CNOT’ and the rotational approaches. For the generalised ‘CNOT’ with 4 control qubits we need 7 Toffoli gates, whereas for the rotational implementation we need 55 gates.

The main reasons for this difference are two: (1) the lack of a gate in Qiskit that can implement \( \Phi(\delta) \) leads to a very complicated method to implement it using \( U_3, R_z \) and ‘CNOT’ gates. To give more context, 14 quantum gates and 2 ancilla qubits are necessary as, if we didn’t use them, we would have to add even more gates in the quantum circuit; and (2) the whole idea of the rotational implementation rests on the trade-off between workspace and quantum gate count. In other words, the smaller the workspace (the less ancilla qubits) the more quantum gates are necessary to rotate the qubits in a way that can compensate for the loss in intermediatively saved states.

Appendix C: Method of Stationary Phase

1. Method

The method of mathematical analysis called the method of stationary phase, as explained by [17], allows us to estimate the integrals derived in equation (23). Based on the fact that in integrals of type

\[
I(t) = \int_a^b g(k) e^{it\phi(k)} \, dk
\]

the largest contribution comes from areas where the oscillations in phase are small, that is, near stationary points of the function \( \phi(k) \). Generally, supposing that the function \( g(k) \) is reasonably smooth and does not vanish in the only stationary point, \( a \), the order \( p \) of the interval \( [a; b] \) with \( t \to \infty \), we can make the approximation

\[
I(t) \sim g(a) \exp \left\{ it\phi(a) + \text{sgn} \left[ \phi^{(p)}(a) \right] \frac{\pi}{2p} \right\}
\]

where \( \phi^{(p)}(a) \neq 0 \) from the assumption and \( \text{sgn} \) is a function that returns the sign.

Especially for the case of \( p = 2 \) we get

\[
I(t) \sim \sqrt{\frac{\pi}{2t \left| \phi''(a) \right|}} g(a) \exp \left\{ it\phi(a) + \text{sgn} \left[ \phi''(a) \right] \frac{\pi}{4} \right\}\] (C1)

2. Approximation of Hadamard Walk Evolution

The integrals from equation (23) can be transformed to suit equation (C1), first by setting \( n = \lambda t \), when these integrals obtain the form

\[
I(t; \lambda) = \frac{1}{2\pi} \int_0^\pi g(k) e^{i\phi(k; \lambda) t} \, dk \quad (C2)
\]

where \( \phi(k; \lambda) = k\lambda - \omega_k \) and \( g(k) \) is either an even or an odd function. In this slightly generalised case there are no stationary points for \( \lambda > 1/\sqrt{2} \) or \( \lambda < -1/\sqrt{2} \) and \( I(t; \lambda) \) by getting with \( \lambda \) further from zero decreases exponentially fast. For \( \lambda \in (-1/\sqrt{2}, 1/\sqrt{2}) \) we find stationary points \( \pm k_\lambda \in [0; \pi] \) of order \( p = 2 \), where

\[
\cos k_\lambda = \frac{\lambda}{\sqrt{1 - \lambda^2}}
\]

and

\[
\frac{\partial^2 \phi}{\partial k^2} \left( \pm k_\lambda; \lambda \right) = \pm \left( 1 - \lambda^2 \right) \sqrt{1 - 2\lambda^2} \quad (= \omega''_\lambda)
\]

Under these conditions, and by dividing the integration range \([-\pi; \pi]\) in equation (C2) into four subintervals by points 0 and \( \pm k_\lambda \) we find

\[
I(t; \lambda) = \frac{2g(k_\lambda)}{\sqrt{2\pi t |\omega''_{k_\lambda}|}} \begin{cases} \cos \left( t\phi(k_\lambda; \lambda) + \frac{\pi}{4} \right), & \text{for } g \text{ even} \\ \sin \left( t\phi(k_\lambda; \lambda) + \frac{\pi}{4} \right), & \text{for } g \text{ odd} \end{cases}
\]

Finally, for equations (23) we obtain

\[
\alpha'(\lambda t) \sim \frac{2}{\sqrt{2\pi t |\omega''_{k_\lambda}|}} \cos \left( t\phi(k_\lambda; \lambda) + \frac{\pi}{4} \right),
\]

\[
\beta'(\lambda t) \sim \frac{2\lambda}{\sqrt{2\pi t |\omega''_{k_\lambda}|}} \cos \left( t\phi(k_\lambda; \lambda) + \frac{\pi}{4} \right),
\]

\[
\gamma'(\lambda t) \sim \frac{-2\sqrt{1 - 2\lambda^2}}{\sqrt{2\pi t |\omega''_{k_\lambda}|}} \sin \left( t\phi(k_\lambda; \lambda) + \frac{\pi}{4} \right).
\]

Now, the probability of being in position \( n = \lambda t \) after \( t \) steps is

\[
P'\lambda t) \sim \frac{2(1 + \lambda)}{\pi t (1 - \lambda^2) \sqrt{1 - 2\lambda^2}} [1 + \lambda \sqrt{2} \cos \theta]
\]

where \( \theta = 2t\phi(k_\lambda; \lambda) + \frac{\pi}{4} + \mu \) and \( \tan \mu = \frac{\sqrt{1 - 2\lambda^2}}{1 + 2\lambda^2} \).