Delft University of Technology

Quantum Information Project
AP3421-PR

Quantum Phase Estimation

Riccardo Conte
Miguel Carrera Belo
Hjalmar Lindstedt

Quarter 2 - 2021/2022

Contents

Abstract

Quantum phase estimation (QPE) is one of the most important subroutines in quantum computing. The purpose of QPE is to estimate the eigenvalue(s) of a unitary operator. Its importance is reflected in the fact that it serves as a building block in many other significant algorithms, and that relevant problems can be rewritten with the QPE formulation. Our objective is to study the most significant ones, analyze their strengths and weaknesses and define which one is more adequate for different conditions. The ones examined in this report are Kitaev's algorithm, the Iterative Phase Estimation Algorithm (IPEA), Quantum Fourier Transform (QFT) and Approximate Quantum Fourier Transform (AQFT). To compare these methods, a small Python library with high-level Qiskit applications was implemented. The library facilitates QPE-related experiments, and was made publicly available on PyPI and GitHub. Experiments were run for several setups, including (but not limited to) variation of algorithms, unitaries, digits of precision and backends. Based on these results, an analysis in terms of precision, accuracy and noise robustness was made. The conclusion was that in general, considering the fidelity of the available quantum processors, Kitaev was the best method.

1 Introduction

The paper is structured as follows. At the start, the theory needed for the four methods is treated. Then, a small summary of the library we implemented in Python is given. To compare the performances and features of the methods, three experiments have been designed. Therefore, the experiments are presented and, successively, the main results are provided.

1.1 Why QPE?

In Fundamentals of Quantum Information, we briefly studied Quantum Phase Estimation and the concept sparked many questions. After researching a bit, we realized that there are many different approaches to QPE, and that this algorithm is of key importance in quantum computing. It is used in a wide range of quantum algorithms, from solving linear systems (*The HHL algorithm*) to minimizing the number of features required in machine learning applications (*quantum principal component analysis*). Which is even more impressive is the fact that all BQP problems [1] can be rewritten as a QPE problem [16]. This includes one, if not the most, well-known quantum algorithms: Shor's factoring algorithm.

Quantum Phase Estimation algorithms allow us to compute the eigenvalues of any unitary matrix, that is, of any gate in a quantum circuit. This is of high interest in many fields of physics. For instance, if we are able to approximate the Hamiltonian of a complex system by a unitary operator, we can estimate its energy levels[1]. As will be explained later, all these algorithms require having access to eigenvectors $|\psi_k\rangle$, which is not an easy task. However, there have been several studies tackling the issue of using eigenvectors superposition as an input [11] [14].

2 Theory

In this section we will describe in detail the particular QPE algorithms we have used in our simulations. Before that, we will briefly explain which phases are we estimating with these algorithms.

[1] A BQP problem is a computational problem which requires *exponential* time in classical computers, but *polynomial* in quantum computers.

2.1 Unitary Operators

In quantum circuits, gates are described by unitary matrices. Such gates can be anything between a simple X-gate and a complex system of multiple gates operating on several qubits. A unitary matrix is symmetric matrix that fulfill $\hat{U}\hat{U}^\dagger = \hat{U}^\dagger\hat{U} = \hat{I}$. As a direct consequence of this property, the eigenvalues λ of an unitary are complex numbers with module 1.

Let us consider an unitary operator $\hat{U}$ such that

$$\hat{U}\left|\psi\right\rangle = \lambda\left|\psi\right\rangle$$

where $|\psi\rangle$ and λ are, respectively, the eigenvectors and eigenvalues of $\hat{U}$. In general, a unitary that operates on N qubits has dimension $2^N \times 2^N$, and also 2^N eigenvectors [2]. Since $|\lambda| = 1$:

$$\lambda = \mathrm{e}^{2\pi i\varphi} \qquad \varphi \in [0, 1)$$

The eigenphase φ is what we are going to estimate with our algorithms. Remember that any quantum gate has to be unitary, which means we can estimate the eigenphases (and therefore the eigenvalues) of any quantum gate by using QPE.

2.2 Truncated phase

Even though an eigenphase φ can assume all values between 0 and 1, we must still be able to represent it numerically. As will be seen, all algorithms explored in this report represent phase in binary. This means that φ will always be truncated to a finite number of precision digits, m. This truncated phase can be expressed with the binary fraction:

$$\overline{.\alpha_1\alpha_2...\alpha_m} = \sum_{j=1}^{m} 2^{-j}\alpha_j \quad \alpha_j \in \{0, 1\}$$

or in simpler notation just $\varphi^m = 0.x_1x_2...x_m$.

Since φ_m is in base 2, it is easy to show that

$$2^l \cdot \varphi^m = x_1x_2...x_l.x_{l+1}...x_m \tag{1}$$

2.3 Exponentiated Unitaries

When working with truncated phases, one can find interesting properties of an exponentiated unitary. Especially the case where the exponent is $P = 2^p$, $p \in \mathbf{N}$:

$$\hat{U}^P\left|\psi\right\rangle = \lambda^P\left|\psi\right\rangle = e^{P2\pi i\varphi}\left|\psi\right\rangle$$

By using equation 1 we can split $P\varphi$ to an integer part and a decimal part, such that

[2] In the case of degenerate eigenvalues, this statement is is not completely precise. For an eigenvalue with degeneracy k, we have a k-dimensional subspace from where we can choose k linearly independent eigenvectors. However, after performing these choices, we will end up with a total of 2^N eigenvectors

$$P\varphi = P \cdot 0.x_1...x_m = 2^p \cdot 0.x_1...x_m = x_1...x_p + 0.x_{p+1}...x_m$$

The left term is an integer, so the 2π-periodicity of the complex exponential give us

$$e^{2\pi i(x_1...x_p + 0.x_{p+1}...x_m)} |\psi\rangle = e^{2\pi i \cdot 0.x_{p+1}...x_m} |\psi\rangle$$

This property is of key importance for the formulation of the QPE algorithms. For example, by using an exponentiated unitary $U^4 = U^{2^2}$ we can "throw away" the first two digits x_1 and x_2. The third digit x_3 is suddenly the most significant bit, and will therefore be easier to access and estimate!

2.4 QPE Algorithms

QPE algorithms can be roughly classified into two types:

1. Multiple rounds QPE: These algorithms use an ensemble of circuits, and repeats them many times. The ensemble consists of many circuits with the same structure. The circuits are not identical, but they only differ by a few parameters. The accuracy of these algorithms depend on how many repetitions we run them for, while the precision depends on the number of circuits in the ensemble. These circuit require few qubits, namely $d + 1$, where d is the qubit-dimension of the unitary. We will implement the original Kitaev method and the iterative version.

2. QFT-based QPE: These algorithms use inverse Quantum Fourier Transform in order to obtain the estimation. The most significant differences are two. Firstly, a circuit can be run one single time to obtain an estimation of the eigenphase. The second is that the it requires $d + m$ qubits where d is the qubit-dimension of the unitary and m is the number of digits of precision. We will analyze the original QFT and the Approximate-QFT, where the number of small-angle rotations is reduced.

There is a trade-off between these two types. In the first one, few qubits are needed, but it requires running the circuits several times and applying classical post-processing. On the other hand, QFT-based algorithms only need one circuit that runs once, although the number of qubits scales linearly with the digits of precision. This tradeoff will be considered when comparing the four methods.

It is significant to remark that in all the methods explained below, we assume that we have access to the desired eigenvector $|\varphi\rangle$ of the corresponding unitary. An arbitrary state - a superposition of several eigenstates - would change the behaviour of the algorithms drastically.

Notation

Before describing the algorithms, we will explain some notation.

- mod 1 distance: Distance on the unit circle. For example, $|0.2 - 0.9|_{\text{mod } 1} = 0.3$.
- Precision δ and probability of error ϵ: Our estimation of φ, which we will denote as α, fulfils, with probability of at least $1 - \epsilon$, that $|\alpha - \varphi|_{\text{mod } 1} < \delta$.

2.4.1 Kitaev's Algorithm

The original Kitaev's algorithm [7] is probably the most well-known QPE algorithm. Many further studies and variations thereof have been published recently (iterative versions thereof, which we cover in the next subsection).

The algorithm is based on the circuit shown in figure 1, where $Z(\theta) = \left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{smallmatrix}\right)$ and k is an integer. The probabilities of the measurement outcomes are:

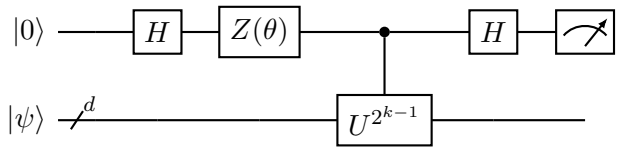


Figure 1: Circuit for Kitaev's algorithm.

$$P(0|k,\theta) = \frac{1+\cos(2\pi 2^{k-1}\varphi+\theta)}{2} \qquad P(1|k,\theta) = \frac{1-\cos(2\pi 2^{k-1}\varphi+\theta)}{2}$$

We will only use two values of θ: 0 and $\pi/2$, so the probabilities depend, respectively, on the cosine and sine of $2\pi 2^{k-1}\varphi$. Assuming we run each circuit s times, we can obtain an estimate of $2^{k-1}\varphi$, which we will denote as ρ_k:

$$\begin{cases} \mathrm{E}(\cos) = \dfrac{n_{0,0}-n_{1,0}}{s} \\ \mathrm{E}(\sin) = \dfrac{n_{0,\pi/2}-n_{1,\pi/2}}{s} \end{cases} \rightarrow \rho_k \equiv 2^{k-1}\varphi = \frac{1}{2\pi}\arctan\left(\frac{\mathrm{E}(\sin)}{\mathrm{E}(\cos)}\right) \tag{2}$$

where $n_{i,j}$ is the number of times we obtained the outcome i for $\theta = j$. Let us remark that we should choose the right quadrant for the arctan, depending on the signs of the cosine and sine.

We want to ensure that our estimation ρ_k is sufficiently close to the real value $2^{k-1}\varphi$. To do so, we must fix the number of times we repeat the circuit (s). It can be proven [7], by means of Chernoff's bound, that the probability of error is bounded by $\sim e^{-\delta^2 s}$. This means that for a given δ, we can find a constant such that the error of our estimation is smaller that ϵ. We find

$$s \sim \mathcal{O}(\log(1/\epsilon) = \mathcal{O}(\log m) \tag{3}$$

We choose $\delta = 1/16$, which means that we have a precision of $1/16$ for approximating $2^{k-1}\varphi$, that is:

$$|\rho_k - 2^{k-1}\varphi|_{mod\ 1} < 1/16 \tag{4}$$

Below we will show an algorithm to sharpen this precision exponentially.

Let us consider we use m different values for k:

$$k \in [m, m-1, \ \ldots, 1]$$

For each one, we will have two values of θ, i.e, we need $2m$ circuits and $2ms$ measurements in total. The estimation of the phase will then be $\alpha = \overline{.\alpha_1\alpha_2\ldots\alpha_{m+2}}$ (we get two more digits than the number of values of k we use!). In order to get the digits of this estimation, we follow the algorithm below.

Kitaev Algorithm

1. `for k = [m,m-1,...1]`:
   We compute ρ_k using $2s$ measurements each (eq 2).

2. `for k = [m,m-1,...1]`:
   Round ρ_k to the closest octant value $\{0/8, 1/8, \ldots 7/8\}$. Define these truncated versions of ρ_k as β_k.

3. Let β_m define the three least significant bits. That is: $\overline{.\alpha_m\alpha_{m+1}\alpha_{m+2}} = \beta_m$

4. `for k = [m-1,...1]`:

   Let $\alpha_k = \begin{cases} 0 & \text{if } |\overline{.0\alpha_{k+1}\alpha_{k+2}} - \beta_k|_{mod\ 1} < 1/4 \\ 1 & \text{if } |\overline{.1\alpha_{k+1}\alpha_{k+2}} - \beta_k|_{mod\ 1} < 1/4 \end{cases}$

5. Our estimation is $\alpha = \overline{.\underbrace{\alpha_1\alpha_2 \ldots \alpha_{m-1}}_{\text{for loop in step 4}}\underbrace{\alpha_m\alpha_{m+1}\alpha_{m+2}}_{\beta_m}}$

Let us now explain with less formalism the algorithm, so the reader can understand more clearly each step. In step 1, we run $2k$ versions of the circuit in figure 1. The 2 comes from the fact that we use 2 values for θ. Every circuit is run s times, which results in a total of $2ks$ runs. With the outcomes, we compute ρ_k using equation 2.

In step two, we round each real number ρ_k to three bits of precision, obtaining β_k. This ensures that $|\rho_k - \beta_k|_{mod\ 1} < 1/16$. Using this fact, equation 4 and the triangular inequality, we easily get

$$|\beta_k - 2^{k-1}\varphi|_{mod\ 1} < 1/8 \tag{5}$$

Thus, with step 2 we get a 1/8-precision estimation for each ρ_k, which may seem like a step backwards compared to equation 4. However, this estimation will be key in step 4, where we really sharpen the precision to make it exponential in m. In step 3, we get the first three digits of our final estimation α, which are the binary representation of β_m. After this, we do a for loop in order to get the remaining $m-1$ digits of α. The condition for choosing each digit might sound arbitrary. Nevertheless, by using that condition and equation 5, it can be proven that for each digit we add to our estimation α, it is guaranteed that:

$$|\overline{.\alpha_k \ldots \alpha_{m+2}} - 2^{k-1}\varphi|_{mod\ 1} < 2^{-m+3-j} \Rightarrow |\alpha = \overline{.\alpha_1\alpha_2 \ldots \alpha_{m+2}} - \varphi|_{mod\ 1} < 2^{-m+2} \tag{6}$$

Thus, α is an estimation of φ with exponential precision. The number of total measurements required is $\mathcal{O}(m \cdot s) = \mathcal{O}(m \cdot \log m)$

In summary, we choose s such that a precision of 1/16 is guaranteed for each estimation of $2^{k-1}\varphi$. Thanks to this precision, and by means of a pretty complex post-process, we get an estimation of φ whose precision scales exponentially with the number of k we use. The classical post-processing shown here is the original one from Kitaev, which is indeed intricate and some steps seem to lack a reason why they are taken. There are other post-processing which are simpler [13], and even some with a geometric interpretation [3].

2.4.2 Iterative Phase Estimation Algorithm (IPEA)

Several iterative QPE algorithms based on Kitaev's have been created [13][15]. In particular, we will focus on the one described in [2] (see [6][5] for a student friendly overview on the method and how to implement it on Qiskit).

The algorithm consists of performing m rounds with the circuit shown in figure 2, where

$$R_z(\omega_k) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\omega_k} \end{pmatrix} \tag{7}$$

is a correction that changes every round and explicitly depends on the results of the previous rounds. Each round we will obtain one digit of our estimation $\alpha = .\overline{\alpha_1\alpha_2...\alpha_m}$ (first α_m, then α_{m-1} and so on). This means that the correction applied when estimating a digit depends only on the less significant digits. We will see in the next subsection that the QFT method also applies corrections based on less significant digits, but in a different manner. Thus, this iterative method can be seen as a mix of Kitaev's (since the circuit is basically the same) and of QFT's. In the table below we describe each round of the algorithm and the applied corrections.

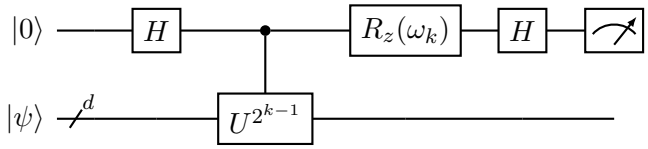


Figure 2: Circuit for k-round of IPEA. k runs from m to 1.

Iterative Algorithm

In every round qubit q_0 is initialized as $|+\rangle$ and q_1 as $|\psi\rangle$.

1. **Round 1**
   - 1.1. We perform 2^{m-1} controlled-$\hat{U}$ operations (q_0 controls q_1).
   - 1.2. The state of q_0 becomes $|0\rangle + \mathrm{e}^{2\pi i 2^{m-1}\varphi}|1\rangle = |0\rangle + \mathrm{e}^{2\pi.\overline{\alpha_m}}|1\rangle$.
   - 1.3. We measure q_0 in the X-basis $\begin{cases}\text{Outcome } +1 \Rightarrow q_0 = |+\rangle \Rightarrow \alpha_m = 0\\ \text{Outcome } -1 \Rightarrow q_0 = |-\rangle \Rightarrow \alpha_m = 1\end{cases}$
2. **Round 2**
   - 2.1. We perform 2^{m-2} controlled-$\hat{U}$ operations (q_0 controls q_1).
   - 2.2. The state of q_0 becomes $|0\rangle + \mathrm{e}^{2\pi.\overline{\alpha_{m-1}\alpha_m}}|1\rangle$.
   - 2.3 We perform a $R_z\left(-2\pi\frac{\alpha_m}{4}\right)$ correction on q_0, so its state becomes $|0\rangle + \mathrm{e}^{2\pi.\overline{\alpha_{m-1}}}|1\rangle$.
   - 2.4. We measure q_0 in the X-basis $\begin{cases}\text{Outcome } +1 \Rightarrow q_0 = |+\rangle \Rightarrow \alpha_{m-1} = 0\\ \text{Outcome } -1 \Rightarrow q_0 = |-\rangle \Rightarrow \alpha_{m-1} = 1\end{cases}$
3. **Round k**
   - k.1. We perform 2^{m-k} controlled-$\hat{U}$ operations (q_0 controls q_1).
   - k.2. The state of q_0 becomes $|0\rangle + \mathrm{e}^{2\pi.\overline{\alpha_{m-k+1}\alpha_{m-k+2}\ldots\alpha_m}}|1\rangle$.
   - k.3 Correction $R_z\left[-2\pi\left(\frac{\alpha_{m-k+2}}{4} + \frac{\alpha_{m-k+3}}{8} + \ \ldots\ + \frac{\alpha_m}{2^k}\right)\right]$ on $q_0 \rightarrow |q_0\rangle = |0\rangle + \mathrm{e}^{2\pi.\overline{\alpha_{m-k+1}}}|1\rangle$.
   - k.4. We measure q_0 in the X-basis $\begin{cases}\text{Outcome } +1 \Rightarrow q_0 = |+\rangle \Rightarrow \alpha_{m-k+1} = 0\\ \text{Outcome } -1 \Rightarrow q_0 = |-\rangle \Rightarrow \alpha_{m-k+1} = 1\end{cases}$
4. Iterate until round m, in which we obtain α_1.

The applied corrections have a clear goal: eliminate the contribution of the prior digits, so when we measure in the X-basis we only see the effect of the current digit. However, this implies that for large m we must have access to really small Z-rotations, which requires very precise quantum processors.

When φ can be perfectly represented with m digits, we deterministically get $\alpha = \varphi$. When this is not the case, we have $\varphi = \alpha + \delta 2^{-m}$ with $\delta \in (0, 1)$. This makes the algorithm probabilistic, with a conditional probability for each bit to be measured correctly. This probability is $P_k = \cos^2(\pi 2^{k-m-1}\delta)$,which gives us the overall probability of the algorithm to extract α:

$$P(\delta) = \prod_{k=1}^{m} P_k = \frac{\sin^2(\pi\delta)}{2^{2m}\sin^2(\pi 2^{-m}\delta)}$$

This means that, for $\delta \leq 1/2$ the best m-approximation to φ is α, while for $\delta > 1/2$, $\alpha + 2^{-m}$ is better. If we accept both answers α and $\alpha + 2^{-m}$, then the IPEA determines the phase with accuracy $1/2^m$ and probability of error $\epsilon < 1 - 8/\pi^2$ (independent of m!).

This iterative version clearly enhance the performance compared to the Kitaev's original version. In the IPEA scheme, the bits of the phase are measured directly, without any need for classical post-processing. Moreover, each bit has to be measured only once, compared to $\log m$ times in Kitaev's. When the phase φ has a binary expansion with no more than m bits, the IPEA *deterministically* extracts all bits, in contrast to Kitaev's which is always probabilistic. The IPEA is also optimal in the sense that a full bit of information is gained in each measurement.

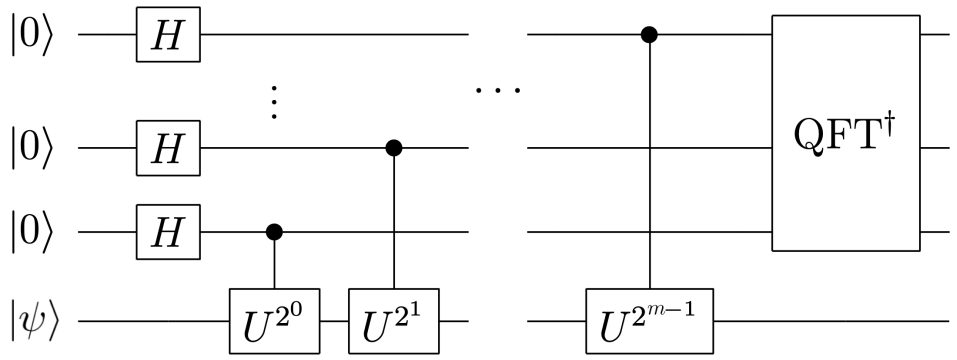


Figure 3: Circuit QPE besed on QFT with one-qubit unitary and m digits of accuracy.

In our implementation, we cannot use conditional operations (still in development for real backends), so we had to variate a bit the scheme. What we did is to send each round individually to IBM (either backend or simulator), and use the results in the following rounds. Therefore, if we set a large number of shots, we will get a distribution of *every digit*, instead of a distribution of the whole estimation, and we take the most frequent outcome in each round.

2.4.3 Quantum Fourier Transform (QFT)

The Quantum Phase Estimation algorithm based on Quantum Fourier Transform [3] [4] [8] [9] is surely the most compact; it can be performed running the circuit in figure 3 only once. The qubits involved in the algorithm can be grouped into two main registers. The first register has m qubits, which corresponds to the number of digits of accuracy chosen for the estimation of the eigenphase. We will call this register *phase register*. The second has the same dimension of the unitary operator U which has to be inspected, that we identify with d. We will call it *ancilla register*. The main steps of the algorithm are as follows:

1. The registers are initialized
2. Hadamard gates are applied to the *phase register*
3. A set of controlled exponentiated unitaries are applied to the *ancilla register* with the *phase register* qubits as controllers
4. The Inverse Quantum Fourier Transform is applied to the *phase register*
5. The qubits in the phase register are measured

The initialization is very similar to the one done in the previous implementation of the QPE: the qubits of the *phase register* are initialized in $|0\rangle$, and then put in a maximal superposition with the help of Hadamard gates. The *ancilla register* is initialized in $|\psi\rangle$, an eigenstate of the unitary operator. Therefore the initial state is:

$$\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes \cdots \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes |\psi\rangle \tag{8}$$

[3] Although the method is called Quantum Fourier Transform, it is actually the **inverse** Quantum Fourier Transform that is used. By convention we will mostly use the name QFT, although it is technically wrong

Then the control unitaries are applied. Since the $|1\rangle$ is the agent that triggers these controlled operations, the state becomes:

$$|y_m\rangle \otimes |y_{m-1}\rangle \otimes ... \otimes |y_1\rangle \otimes |\psi\rangle =$$
$$\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\, \mathrm{U}^{2^{m-1}}\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\, \mathrm{U}^{2^{m-2}}\right) \otimes \cdots \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\, \mathrm{U}\right) \otimes |\psi\rangle =$$

where U operates on the *ancilla register*. Since $|\psi\rangle$ is an eigenstate of U with eigenvalue $e^{2\pi i\varphi}$, we find

$$\frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{2\pi i 2^{m-1}\varphi}\,|1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{2\pi i 2^{m-2}\varphi}\,|1\rangle\right) \otimes \cdots \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{2\pi i\varphi}\,|1\rangle\right) \otimes |\psi\rangle \tag{9}$$

One can see that the qubits are still in a product state, and therefore not entangled. We can therefore ignore the $|\psi\rangle$ state for the rest of the calculation.

Using the properties of exponentiated unitaries described in section 2.3:

$$e^{2\pi i\varphi} = e^{2\pi i 0.x_1x_2x_3...x_mx_{m+1}x_{m+2}...}$$
$$e^{2\pi i 2\varphi} = e^{2\pi i 0.x_2x_3...x_mx_{m+1}x_{m+2}...}$$
$$e^{2\pi i 4\varphi} = e^{2\pi i 0.x_3...x_mx_{m+1}x_{m+2}...}$$
$$\cdots$$
$$e^{2\pi i 2^{m-1}\varphi} = e^{2\pi i 0.x_mx_{m+1}x_{m+2}...}$$

The eigenphase does not need to be representable with m digits, i.e. what we can evaluate with a *phase register* of dimension m. However, we start by describing the particular scenario in which φ has exactly m digits, This is later on extended to the general case.

In this scenario, with $\varphi = 0.x_1x_2x_3...x_m$, the application of U^{m-1} generates the state $\frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{2\pi i 0.x_m}\,|1\rangle\right)$.

Then the Inverse Quantum Fourier Transform is applied. To better explain the effect of this step let us introduce the simple case where $m = 3$. The corresponding circuit is presented in figure 4.

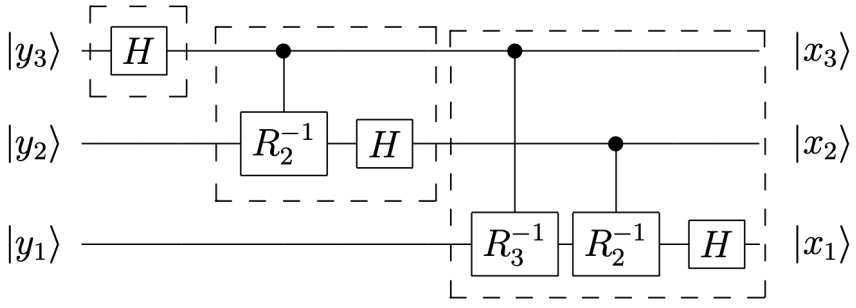


Figure 4: Circuit $\mathrm{QFT}^\dagger$ when $m = 3$.

The state of the top qubit $|y_3\rangle$ becomes:

$$|y_3\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{2\pi i 0.x_3}\,|1\rangle\right) = \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{x_3}\,|1\rangle\right)$$

We see that this state can be written as

$$|y_3\rangle = \begin{cases} |+\rangle & \text{if } x_3 = 0 \\ |-\rangle & \text{if } x_3 = 1 \end{cases}$$

Using this combined with the fact that $H|+\rangle = |0\rangle$ and $H|-\rangle = |1\rangle$, we find

$$|y_3\rangle \xrightarrow{\mathrm{H}} |x_3\rangle$$

The gate R_k^{-1} applies a Z-rotation of the angle $-\frac{2\pi}{2^k}$. The corresponding matrix is:

$$\mathrm{R}_k^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{-2\pi i/2^k} \end{pmatrix}$$

Since $|y_3\rangle = |x_3\rangle$ is the controller of the first rotation gate R_2^{-1} we find

$$|y_2\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{2\pi i 0.x_2x_3}|1\rangle\right) \xrightarrow{\mathrm{cR}_2^{-1}} \frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{2\pi i(0.x_2x_3 - 0.0x_3)}|1\rangle\right) = \frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{2\pi i 0.x_2}|1\rangle\right)$$

With the same reasoning as above, we find that

$$|y_2\rangle \xrightarrow{\mathrm{H}} |x_2\rangle$$

We then have two controlled rotations R_3^{-1} and R_2^{-1} controlled by x_3 and x_2 respectively. These act on the last qubit state $|y_1\rangle$, yielding

$$|y_1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{2\pi i 0.x_1x_2x_3}|1\rangle\right) \xrightarrow{\mathrm{cR}_3^{-1}\mathrm{cR}_2^{-1}} \frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{2\pi i(0.x_1x_2x_3 - 0.0x_2x_3)}|1\rangle\right) \xrightarrow{\mathrm{H}} |x_1\rangle$$

After performing all these steps, we find that each qubit of the *phase register* expresses a digit of the eigenphase we wanted to estimate. Thus the entire register is just $\omega = 2^m\varphi$, the integer version of the eigenphase. This result is not dependent on the assumption $m = 3$. Therefore we can relax it, and consider an arbitrary dimension m. In this case $|y_m\rangle$ is not corrected, $|y_{m-1}\rangle$ is corrected by cR_2^{-1}, $|y_{m-2}\rangle$ is corrected by $cR_3^{-1}cR_2^{-1}$, etc.,and $|y_1\rangle$ is corrected by $(m-1)$ rotations from all the less significant qubits. Assuming that the eigenphase is representable by m bits and that all gates are noiseless, we obtain the eigenphase without any error. This point will be evaluated experimentally in Experiment 1, where using a perfect simulator and choosing a representable phase always bring the correct result.

Let us also note that the combination of rotations applied to each qubit is very similar task to what is done by $R_z(\omega_k)$ in the IPEA method. However, there is a slight difference. In the QFT the rotations are controlled by the states of the previous qubits. These could be a superposition of $|0\rangle$ and $|1\rangle$ if one or both of the two conditions for perfect estimation are not fulfilled. In the IPEA, the angle of rotation is evaluated with the measurements of the qubits.

To understand what happens when the eigenphase is not perfectly representable by m digits, we take the main results of [14] and [12], useful for this analysis.

We consider that the *ancilla register* is initialized in a superposition of eigenstates $|\varphi_k\rangle$, but in the following, we restrict back the result to the case where just one eigenstate is sent in input. Therefore,

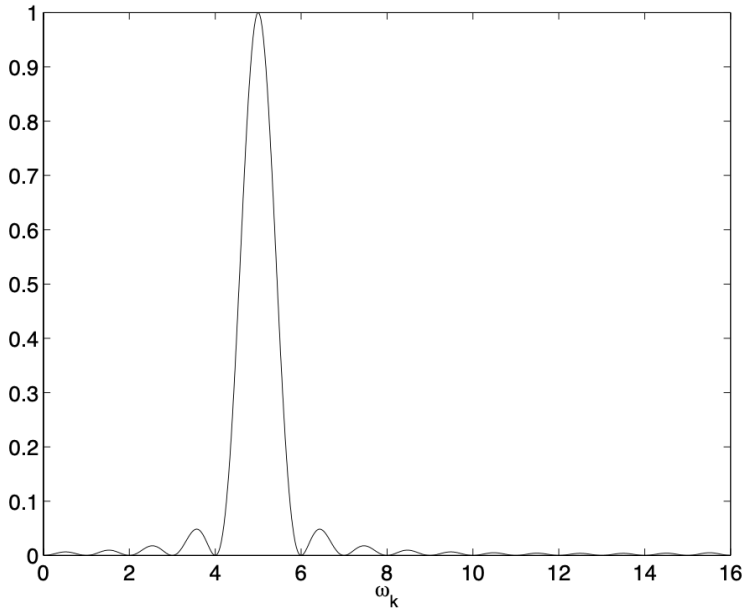


Figure 5: $|f(\omega_k, j)|$ as a function of ω_k, choosing $M = 16$ and $j = 5$.

this analysis leads to highly general conclusions that can be used for further studies to understand how QPE can handle generic states in input. The global state of the *phase register* after the QFT† can be described as:

$$|\Psi_3\rangle = \sum_k c_k \sum_{j=0}^{M-1} f(\omega_k, j) \, |j\rangle \, |\varphi_k\rangle \tag{10}$$

where $M = 2^m$, $\quad \omega_k = \varphi_k M$ (i.e. $x_1^k x_2^k ... x_m^k . x_{m+1}^k ...$), $\quad |j\rangle$ is the state of the *phase register*

and

$$f(\omega_k, j) = \begin{cases} \dfrac{1}{M} \dfrac{\sin(\pi\omega_k)}{\sin(\pi\frac{\omega_k - j}{M})} e^{\pi i(\omega_k - \frac{\omega_k - j}{M})} & : \omega_k \neq j \\ 1 & : \omega_k = j \end{cases} \tag{11}$$

The function $f(\omega_k, j)$ is depicted in the figure 5. It represents the distribution of $f(\omega_k, J = 5)$ with respect to ω_k when $M = 16$ ($m = 4$), and j=5 is estimated by the QPE. To understand the meaning of $f(\omega_k, j)$, let us consider two specific cases:

1. **ω_k are integers**

   In this case $f(\omega_k, j) = \delta_{\omega_k - j}$. So, measuring the value $j = 5$, we can be sure that the associated eigenvalue is $\omega_k = 5$. This corresponds to the same result we found assuming a representable φ because they are the same condition.

2. **One eigenstate in input φ_l**

   In this case $c_k = \delta_{\omega_k - l}$, and the state of the *phase register* is just

   $$|\Psi_3\rangle = \sum_{j=0}^{M-1} f(\omega_l, j) \, |j\rangle \, |\varphi_l\rangle$$

   Therefore, $|f(\omega_k, j)|^2$ results to be the probability that measured j, the real eigenphase we were interested in was ω_l. It is remarkable to notice that measuring $j = 5$, the desired phase cannot be another integer value, e.g. $\omega_l = 6$, but it could be $\omega_l = 6.5$, generating an error in the estimation.

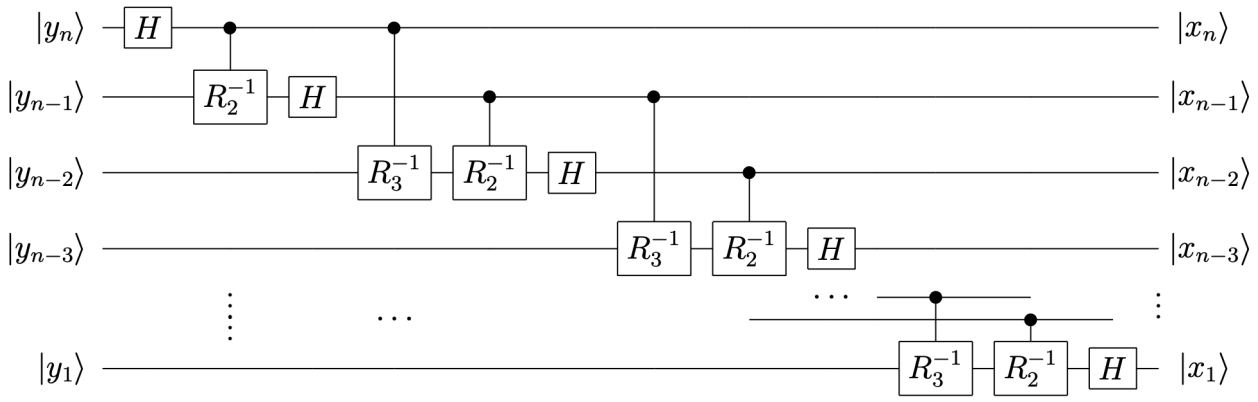


Figure 6: AQFT with constant rotations: $l = 2$.

It can also be noticed that:

$$|f(\omega_k, j)| \geq \text{sinc}(\omega_k - j)$$

Using this observation and restricting again the scenario to the case where the *ancilla register* is initialized with just one eigenstate φ_l, it can be derived a lower bound for the probability to measure the closest integer to ω_l, and for the probability to measure one of the integer ω_l is in between[4].

$$P[\omega_l] > 0.4$$
$$P(\lceil \omega_l \rceil \text{ or } \lfloor \omega_l \rfloor) > 0.8 \tag{12}$$

where $[\omega_l]$: closest integer to ω_l $\lceil \omega_l \rceil$: upper integer to ω_l $\lfloor \omega_l \rfloor$: lower integer to ω_l

From the reference [12], it can be found a slighly better bound:

$$P[\omega_l] \geq \frac{4}{\pi^2}$$
$$P(\lceil \omega_l \rceil \text{ and } \lfloor \omega_l \rfloor) \geq \frac{8}{\pi^2} \tag{13}$$

In Experiment 1 we will test the theory, finding the probability of success when the circuit is run in a perfect simulator, but with an eigenphase that is not perfectly representable.

2.4.4 Approximate Quantum Fourier Transform (AQFT)

Many approximations of the QPE based on QFT have been developed. The one we will analyze is identical to the QFT for the first part of the circuit, where the control unitaries are implemented. It differs in the sense that a constant number of correcting rotations (l) are applied to each qubit (see Figure 6). The phase shift operators which are neglected are those characterized by small angles, used to reach higher precision. Even if these rotations lead to a better theoretical result, they are problematic in practice. Sometimes it is not feasible to realize small rotations with high accuracy in hardware. If this is the case, the small rotations could increase the error inside the protocol,hence being counterproductive.

[4]Note that the brackets around ω_l are different in the different cases. This can easily be overlooked, which could cause a lot of confusion

Considering the constant number of rotations equal to 2, a generic qubit state is:

$$|x_k\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{2\pi i\tilde{\varphi}}\,|1\rangle\right)\frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{2\pi i 0.x_k 00 x_{k+3}\cdots}\,|1\rangle\right)$$

We see that the two digits x_{k+1} and x_{k+2} are cancelled, but the remaining less significant digits are still present.

We can consider the extra digits as an extra phase. Therefore:

$$\tilde{\varphi} = 0.x_{k+1}00x_{k+4}... = 0.x_{k+1} + \theta$$

$$|x_k\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{2\pi i(0.x_{k+1}+\theta)}\,|1\rangle\right)$$

where

$$|\theta| < \frac{1}{8}$$

Let us consider the case in which $x_k = 0$, the probability of measuring the correct state is:

$$P(x_{meas} = 0 \mid x_k = 0) = \cos^2(\pi\theta)$$

Thus,

$$P(x_{meas} = 0 \mid x_k = 0) > \cos^2(\frac{\pi}{8}) \approx 0.85$$

Choosing $x_k = 1$ we would have found a similar result.

Therefore, from this theoretical analysis, we conclude that the probability of measuring correctly each digit of the eigenphase is bounded by the value 0.85. Let us note that even if the phase is perfectly representable by the qubits of the *phase register*, the probability of success of the algorithm is not 1, as in the original QFT. Considering $l \geq \log_2 m + 2$ the global probability of success is :

$$P \geq \frac{4}{\pi^2} - \frac{1}{4m} \tag{14}$$

Because of its intrinsic probabilistic behaviour, this algorithm has to be iterated several times in order to apply a majority vote procedure. Using the Chernoff's bound, it can be shown that in order to obtain a success probability of $1 - \epsilon$, we need at least:

$$s = 4 \ln \frac{1}{\epsilon} \tag{15}$$

Although it requires to be iterated, it can be shown that the AQFT implementation is more efficient than the QFT, considering how they scale with the number of digits. It is $\mathcal{O}(m \log m)$, the same of Kitaev and IPEA, instead of $\mathcal{O}(m^2)$.

With respect to Kitaev, it has some advantages and disadvantages. One iteration requires more gates, increasing the noise of the result. However, it requires 12 times less the number of measurements; it is an important aspect to be considered in case the state in analysis cannot be replicated because each measurement alters the state in the *ancilla register*.

3 Method & Implementation

The different methods were implemented in Python. This section will cover the structure and functionalities of the repository briefly. For more detailed documentation, as well as the actual code, please refer to the GitHub page.

3.1 Backend

The module `QPE.Backend` provides connections to several backends:

- IBMQ 5-qubit processors.
- IBMQ simulators.
- Quantum Inspire - Starmon 5.
- The local machine which the user is running Python on.

To run on IBMQ or Quantum inspire, one must provide their personal API token.

3.2 Unitary

In the module `QPE.Unitary` we can find the `Unitary` and the `Eigenvector` classes. One can choose between creating a random `Unitary` object from a matrix or a Qiskit `QuantumCircuit` or just generate a random one. This object provides the necessary controlled unitaries to our Quantum Circuit, with an optional arbitrary exponent. It also serves the role as an "answer sheet", i.e. it holds information about its eigenvectors and eigenvalues. This is necessary for validation and evaluation of the algorithms we run.

3.3 PhaseEstimator

The center of the code is the `PhaseEstimator` class from the module `QPE.PhaseEstimator`. This is the parent class of all algorithms, and holds some general functionalities for all of them.

Input

To perform an experiment, create a `Unitary` and pass it to a subclass of the `PhaseEstimator` along with *n_digits*, an integer which defines how many digits of precision we want. Some other optional input parameters are

- **Input states:** The states which are initialised and passed through the unitaries can be specified. Either in the computational basis or in the eigenvector basis of the unitary.
- **Backend:** One can give a dictionary with instructions to define the connection with the backend.
- **Shots:** The amount of shots per experiment.

Running an Experiment

After initialization, one can run the experiment. The `PhaseEstimator` will construct the appropriate circuit to match the input parameters and send it to the backend. After this it waits for the result to come back and performs the necessary post processing to obtain the results.

Data export

The `PhaseEstimator` and the `Unitary` have methods for exporting their data to a Python dictionary. All data types in these dictionaries are compatible with the json format, so one can easily save the files with the results. One can also generate drawings of the circuits and save them as files.

Circuit Serialization

In most scenarios, the algorithms require more than one `QuantumCircuit`. For example, the Kitaev algorithm requires 2m different variants of the circuit depicted in figure 1. Or when running the AQFT, you might want to examine more than one eigenvector, which would require a new circuit. When possible, the `PhaseEstimator` addresses these problems by serializing the circuits. That is, it puts them all together in one single, big `QuantumCircuit` before it is sent to the backend. This reduces the queue time, particularly when sending jobs to real hardware.

3.4 Multiple experiments

If one want to run a big set of different experiments, it is possible to use the `ExperimentSet` class from the `QPE.ExperimentRunner` module. It takes a dictionary filled with lists as its input argument. These lists hold parameters that one wish to send to the `PhaseEstimator` class. The `ExperimentSet` then creates all combinations of these different parameters. Let us illustrate this with an example. Lets say you provide the dictionary shown in figure 7. In this scenario we have the different experimental parameters:

```
d = {
    "name": "example_experiment",
    "filename_flags" : ["U","estimator","backend_service"],
    "path" : "Results/exp1-2",
    "U": [U1,U2],
    "n_digits": [4],
    "estimator": [Kitaev, Iterative],
    "backend_params": [
        {"service":"local","backend_name":"ibmq_qasm_simulator"},
        {"service":"IBMQ", "flags" : ["5qubit"]},
        {"service":"QI"}
        ],
    "n_shots": [1024]
}
```

Figure 7: An instruction dictionary for the `ExperimentSet` class

- 2 `Unitary` objects `U1` and `U2`. (One would obviously need to define these two objects earlier in the code.`U1` and `U2` are variable names in this example)
- One specified precision level, namely 4 digits of precision
- 2 algorithms to use, `Kitaev` and `Iterative` (These should be imported from their respective module prior to defining the dictionary)
- 3 backends, each defined by their own dictionary
- One specified amount of shots, 1024.

One can see that these different parameters make up for a total of $2\times1\times2\times3\times1 = 12$ combinations. The `ExperimentSet` class will generate 12 different instances of the `PhaseEstimator` class and run them all.

When an experiment is finished, it saves the results in a `json`-file. The other key-value pairs in the dictionary specify the configuration of these files:

- `"name"`: the name of the experiment
- `"filename_flags"`: This list of flags indicate which details of the experiments that should be included in the filenames. In this example, a `json`-file could have the name `U1_Kitaev_IBMQ.json`
- 2 `Unitary` objects `U1` and
- `"path"`: The directory where the `json`-files should be stored

4 Experiments

In the sections above we have explain how the methods work and the coding tools we have created to implement the algorithms. Now, we will describe the experiments we will run and the metrics they are meant to compare between the methods. Firstly, let us define the parameters which we will vary in the simulations:

- Dimension of the unitaries. [5]
- Depth permitted: In order to make a more realistic experiment, and also to 'fairly' compare Kitaev's QPE (which requires many runs of multiple circuits for a single estimation) and the rest, we fix a given depth all method cannot exceed.
- Digits of precision, so we analyze how the methods scale in terms of number of qubits required, depth etc.
- Representability of the phase. We will use phases that can and cannot be completely represented in binary for a given precision.
- Backend: We will run the algorithms in a perfect simulator, so we get the theoretical results, and in real backend/noisy simulators to see how imperfect fidelities affect the estimations.

4.1 Experiment 1: 5 qubit comparison

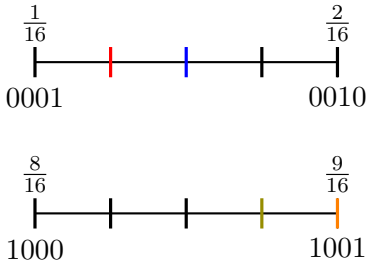
In the first experiment we use 1-qubit unitaries and we set the number of available qubits to five (so we can use IBM's backends [6]) and the digits of precision to four. Moreover, we do not fix any depth, so each method can run an arbitrary number of times to get the best estimation possible. Note that we will obtain one estimation for Kitaev and IPEA, and several estimations for QFT and AQFT, from which we will take the most frequent result. The number of repetitions for each circuit is 1024. We use real 5-qubit backend and a simulator without noise (qasm simulator).

Regarding the unitaries, we use the two diagonal matrices shown in figure 8. We chose them to be diagonal so the backend can easily initialize the eigenvectors. The eigenphases were carefully chosen

[5] We will only use 1 and 2-qubit unitaries, since the transpiled version of higher-dimension gates are incredibly large, which causes significant errors.

[6] We have only access to 5-qubits backends.

so we can analyze the perfomance of the algorithms when the phase can or cannot be represented with 4 bits. In the latter, we also include the possibilities of the actual value of the phase being exactly in the middle between two 4-bitstrings or closer to one of them.



$$U_1 = \begin{pmatrix} e^{i2\pi\frac{9}{16}} & 0 \\ 0 & e^{i2\pi\left(\frac{1}{16}+\frac{1}{32}\right)} \end{pmatrix} \qquad U_2 = \begin{pmatrix} e^{i2\pi\left(\frac{9}{16}-\frac{1}{64}\right)} & 0 \\ 0 & e^{i2\pi\left(\frac{1}{16}+\frac{1}{64}\right)} \end{pmatrix}$$

Figure 8: Unitaries for experiment 1.

In this experiment we also wanted to compare the performance of the methods when they are run in Quantum Inspire's Starmon-5, in order to see if there were appreciable changes due to the change of architecture and provider, and if those can be related to possible changes in the estimated phase. Unfortunately, Starmon-5 does not support both reset operations and multiple measurements of the same qubit, which are the fundamental basis of our Qiskit implementation. Therefore, we are unable to run this experiment with the current version of our code.

4.2 Experiment 2: 2D unitaries and different digits of precision

Now we will study 2-qubit unitaries and run the algorithms for two different digits of precision: 8 and 18. The unitary we will use is

$$U_3 = \begin{pmatrix} e^{i2\pi\frac{1}{3}} & 0 & 0 & 0 \\ 0 & e^{i2\pi\frac{101}{300}} & 0 & 0 \\ 0 & 0 & e^{i2\pi\frac{545}{32768}} & 0 \\ 0 & 0 & 0 & e^{i2\pi\frac{9}{16}} \end{pmatrix}$$

Again, we take a diagonal matrix with thoughtfully chosen eigenphases:

1. A periodic number which we cannot represent exactly in binary: $\frac{1}{3}$
2. A number 'close' to $\frac{1}{3}$ for 8 binary digits, but 'far' away from $\frac{1}{3}$ for 18 binary digits: $\frac{101}{300}$
3. A number which can be expressed in binary with 18 digits but not with 8 digits: $\frac{545}{32768} = \frac{1}{2^6} + \frac{1}{2^{10}} + \frac{1}{2^{15}}$)
4. A number which can be represented in binary for both precisions ($\frac{9}{16}$)

With these conditions, a real backend is no longer an option, but we still have the simulators. We will run these experiments in simulators with and without noise. For the latter, we will make use of `qiskit.test.mock` module, which provides 'fake backends', that is, noise models that try to mock the imperfections of a real backend. More concretely, we will use `FakeTokyo()`, which mocks a retired 20-qubit backend from IBM.

In this experiment, we will also fix the depth of the algorithms, which corresponds to the number of operations of the longest path of the circuit. To do so, we transpile all the circuits needed to perform each algorithm and get their depth. However, this does not seem to be fair, since Kitaev's and IPEA only need 3 qubits each to get an estimate, so we have 17 qubits absolutely unused! To make things

fairer, we will assume we can parallelize the algorithms, so we can send more than one algorithm at a time to the processor, in order to make use of as many qubits as possible, and that their depths are not changed. [7]

For 8 digits of precision, QFT and AQFT methods need 10 qubits, so there can be two algorithms running in parallel, while Kitaev's and IPEA need only three, so six algorithms can be run at the same time. For 18 digits, QFT and AQFT methods need 20 qubits, so we can only run one algorithm, whereas the same number of qubits are required for Kitaev's and IPEA. Taking into account these conditions, we will run the algorithms with the number of shots (n) shown in table 1, which ensures that all the algorithms have the same depth and that they are making the most out of the 20 qubits available. We chose n_1 and n_2 to be orders of magnitude apart, so we can appreciate better the restrictions of having much or little depth available. As expected, in the simulator with noise the differences between the depths, and therefore in the number of shots, are larger (`FakeTokyo()` also restrict the number of native gates available and the connectivity of qubits). We also see that IPEA is the most economic in terms of depth, and actually also in terms of the number of qubits needed, as we explained in section 2.

This experiment aims to be more 'realistic' than the first one. The application of gates and measurements takes some time, so setting the maximum depth is in a sense analogous to setting a maximum runtime. Therefore in this experiment, in contrast to the previous one, we are challenging the algorithms to give us an estimation in a limited time.

Our first idea for this experiment was slightly different. We wanted to simulate the fact that the eigenstate was a result of a physical process (like a photon produced in a nuclear reaction), so we had limited access to it. In order to do so, we would have fixed the depth *and the number of initializations of the phase register*, so we used only one eigenstate for each full algorithm. Nevertheless, the way the code was written made the application of this restriction not doable. This can be a really interesting follow-up experiment in the future.

	Kitaev	**IPEA**	**QFT**	**AQFT**
$\mathbf{n_1}$	13	45	5	5
$\mathbf{n_2}$	1024	3581	208	208

(a) 8 digits with noise

	Kitaev	**IPEA**	**QFT**	**AQFT**
$\mathbf{n_1}$	13	51	5	6
$\mathbf{n_2}$	1024	4044	217	233

(b) 18 digits with noise

	Kitaev	**IPEA**	**QFT**	**AQFT**
$\mathbf{n_1}$	13	19	5	5
$\mathbf{n_2}$	1024	1536	212	212

(c) 8 digits without noise

	Kitaev	**IPEA**	**QFT**	**AQFT**
$\mathbf{n_1}$	13	23	6	6
$\mathbf{n_2}$	1024	1820	247	247

(d) 18 digits without noise

Table 1: Number of shots for each method in two depth regimes

4.3 Experiment 3: distribution of estimations

In this experiment, we will repeat multiple times the set-up of Experiment 1 obtaining different estimations for all the methods. We will do it for different numbers of repetitions and we will compare

[7] We must take into account this is not very realistic, since the architecture of each group of qubits may be different, so the transpiled version of the circuit can slightly change. However, we will assume this change is small enough to neglect it since Tokyo's architecture is pretty symmetric [10].

them, to see how significant the fluctuations are. We will again use U_1. Firstly, we will estimate the phase $\frac{1}{16} + \frac{1}{32}$ in the simulator without noise, so we can see the effect of the probabilistic fluctuations inherent to each model due to non-representable phases. After that, we will estimate in the backend both eigenphases of U_1. With $|(\rangle \psi_0)$ we will analyze the effect of the imperfections of the backend, and with $|(\rangle \psi_1)$ the combination of the fluctuations due to noisy gates and from the probabilistic nature of the algorithms.

We will present the histograms for 10, 30, and 60 estimated values. These are not very large values because for IPEA and Kitaev we had to send many jobs to IBM (in the case of real hardware), and the queue-time was pretty large for the backends we have access to. Let us note that for Kitaev's we need to fix the number of shots of each circuit and then repeat the whole process 10, 30, and 60 times. We chose the number of shots as 60 since it is a value large enough to give us the desired precision.

After analyzing the results, we repeated this experiment with U_2 but only using Kitaev's algorithm, in order to understand better previous findings.

5 Results

5.1 Experiment 1

In the first experiment, we analyze the four methods in their optimal condition; we do not impose any restrictions such as the depth, the total number of gates or the number of initializations. We chose 5-qubits architectures in order to test their performance on the simulator and on real hardware. This is the maximum dimension we have access to on the IBM platform for the real backend. The phases are estimated with 4 digits of precision and they are chosen so that the results span all the interesting cases:

- perfectly representable phase.
- phase in the middle of two representable values.
- phase in between two representable values, but closer to the higher one.
- phase in between two representable values, but closer to the lower one.

Overall result

In the tables 2 and 3 the overall results are provided. Starting from the next section, we present a deeper analysis for each field of the tables in order to better understand how the methods behave.

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.5625	0.5625	0.5625	0.5625	0.5625
Bitstring	1001	1001	1001	1001	1001
Decimal	0.5625	0.5625	0.5625	0.0625	0.5625
Bitstring	1001	1001	1001	0001	1001

(a) $\varphi = \frac{9}{16}$

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.0937	0.0625	0.1250	0.0625	0.1250
Bitstring	0001-0010	0001	0010	0001	0010
Decimal	0.0937	0.0625	0.1250	0.0625	0.1875
Bitstring	0001-0010	0001	0010	0011	0011

(b) $\varphi = \frac{1}{16} + \frac{1}{32}$

Table 2: U_1 for and • simulator and • IBMQ.

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.5468	0.5625	0.5625	0.5625	0.5625
Bitstring	1001	1001	1001	1001	1001
Decimal	0.5468	0.500	0.5625	0.0625	0.0625
Bitstring	1001	1000	1001	0001	0001

(a) $\varphi = \frac{9}{16} - \frac{1}{64}$

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.0781	0.0625	0.0625	0.0625	0.0625
Bitstring	0001	0001	0001	0001	0001
Decimal	0.0781	0.0000	0.0625	0.0625	0.0625
Bitstring	0001	0000	0001	0001	0001

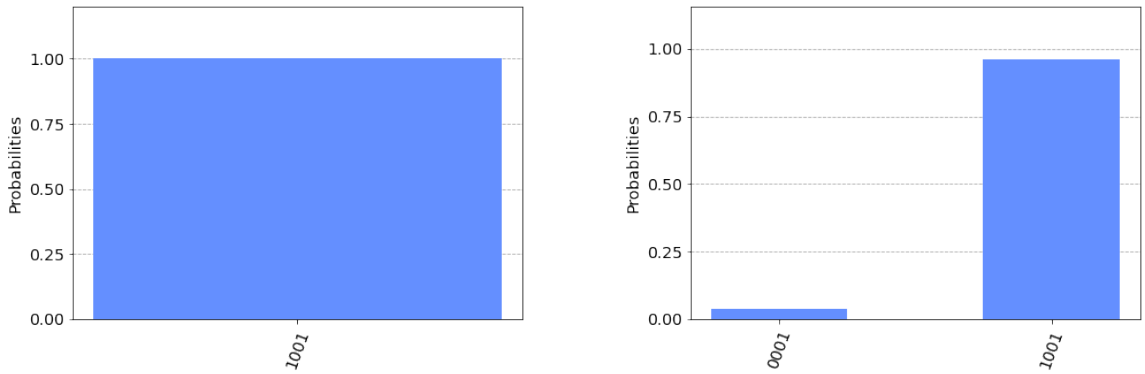
(b) $\varphi = \frac{1}{16} + \frac{1}{64}$

Table 3: U_2 for and • simulator and • IBMQ.

Simulation without noise

In the simulation, all the methods estimate correctly the four different phases. Although these results seem not to help us in the evaluation of the best method, they prove the theory and provide a deeper understanding of the phenomena we are facing. Also, we can confirm that our implementations in qiskit work properly. We focus our attention on two methods - QFT and AQFT - because their outcomes are distributions and not single values.

Let us start with the differences between QFT and AQFT when they estimate a perfectly representable phase (figure 9). The value of the phase is $\varphi_0 = \frac{9}{16} = 0.1001$. The x-axis of the plots corresponds to the integer version of $\varphi_0 : \omega_0 = 1001$, and the y-axis corresponds to the probability of measuring a specific phase at the end of the circuit. These distributions are built running n shots of the same circuit.

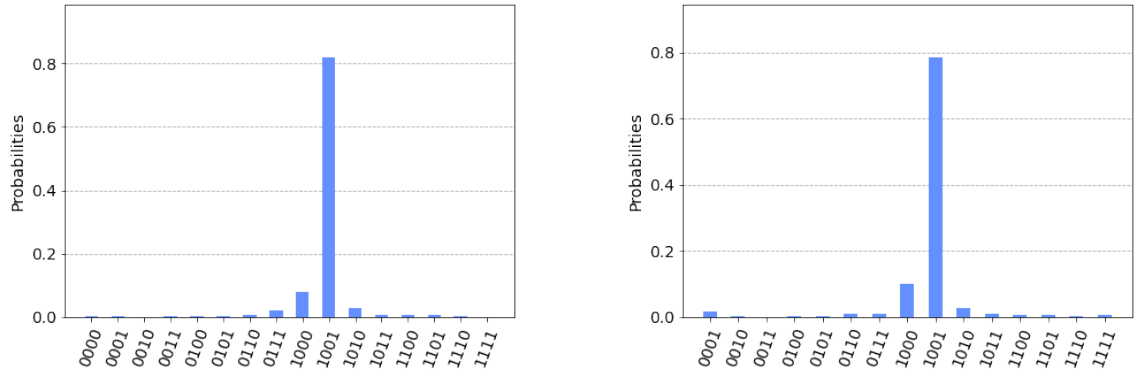


(a) QFT (b) AQFT

Figure 9: Distributions for U_1 eigenstate $|\psi_0\rangle$ in simulator.

With 1024 shots, the QFT always estimates the right phase. This confirms what was theoretically predicted in section 2.4.3. On the other hand, the AQFT with four digits of precision has one correcting rotation less on the most significant bit. Thus, that bit is not always right and it generates the error that can be seen in (b).

Let us now compare what happens if the QFT has a non-representable phase.

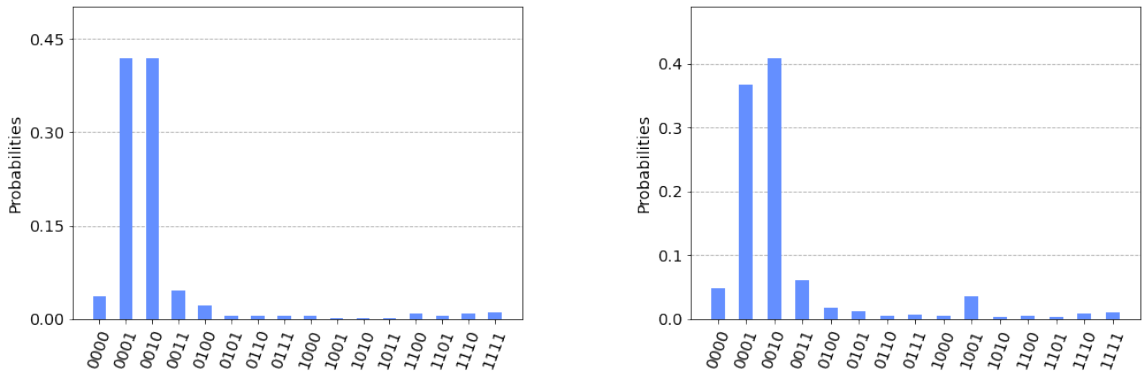


(a) QFT (b) AQFT

Figure 10: Distributions for U_2 eigenstate $|\psi_0\rangle$ in simulator.

In Figure 10, the eigenphase is slightly shifted to the left of '1001'. It is interesting to notice how, for QFT, the plot passed from a single bar in the desired phase to a more spread distribution. Non-representable phases act like noise in the estimation process and, with the gate imperfections, are the main causes of the QPE's errors. Again, this phenomenon is coherent with the theory, which fixes lower bounds for the probability of measuring the closer representable value and for measuring one of the two values the eigenphase is in between (see equations 12 and 13). Thus, the probability for the closer value is roughly 0.8, which is bigger than $\frac{4}{\pi^2}$, and the bars in '1001' and '1000' together bring to a probability similar to 0.9, which is bigger than $\frac{8}{\pi^2}$.

Comparing the previous plot with the one in Figure 11, it can be noticed that, at least in the simulator, changing the position of the estimated phase makes the height of the closest bars vary with the proximity. When the phase is exactly in the middle of two representable integers, they are equiprobable. By increasing the value slightly, the right one becomes more frequent(Figure 10). In the AQFT likewise, the height of the bars gives information on their 'distance' from the phase, but with extra fluctuations. It can be also noticed that the state which has one bit-flip on the MSB, with respect to the desired phase, has a higher probability than its neighbors. In Figure 24 (appendix), where the last eigenphase is examined, this effect is also present.

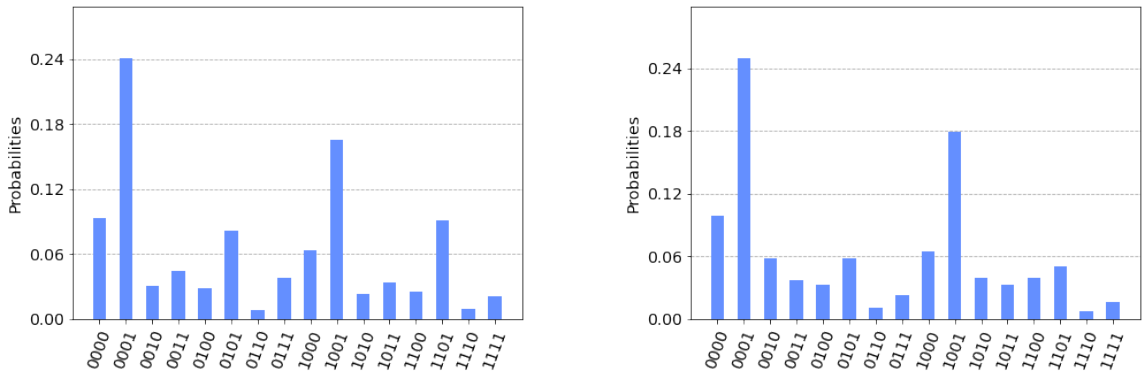


(a) QFT (b) AQFT

Figure 11: Distributions for U_1 eigenstate $|\psi_1\rangle$ in simulator.

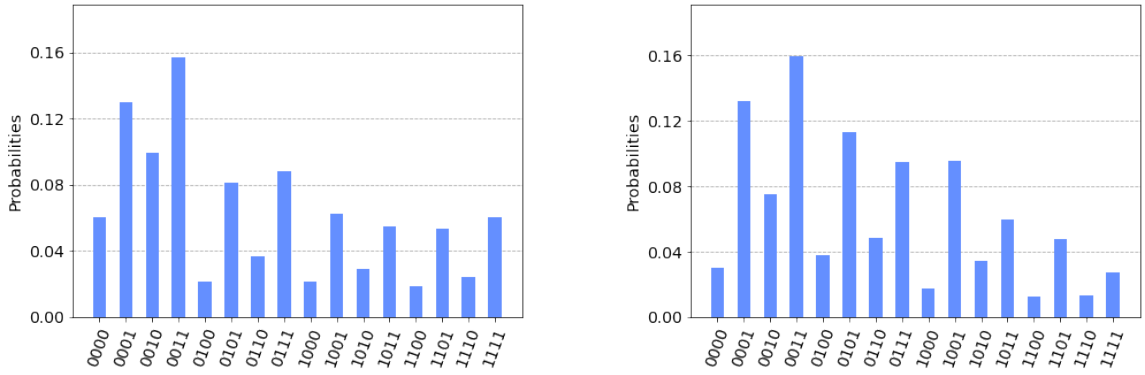
Hardware

The best performer in the hardware is the Iterative algorithm, which succeed four times out of four. The Kitaev implementation failed in two cases, with errors on the LSB. On the other hand, QFT and AQFT had three and two errors respectively , with bit-flips on the MSB. This brings their estimations far from the expected values. AQFT performs a little bit better than QFT because only one of the failures was an error on the MSB. The place where errors typically occur is not random, but it is determined by the different architectures of the methods and their weaknesses. Let us show some plots in order to continue further with our considerations, and see the combined effect of the non-representable phase and the imperfection of the gates.



(a) QFT (b) AQFT

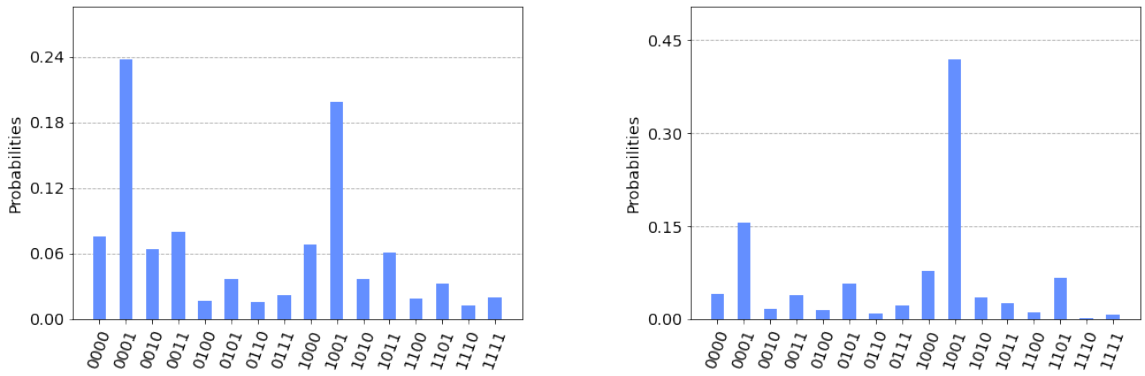
Figure 12: Distributions for U_2 eigenstate $|\psi_0\rangle$ in hardware.



(a) QFT (b) AQFT

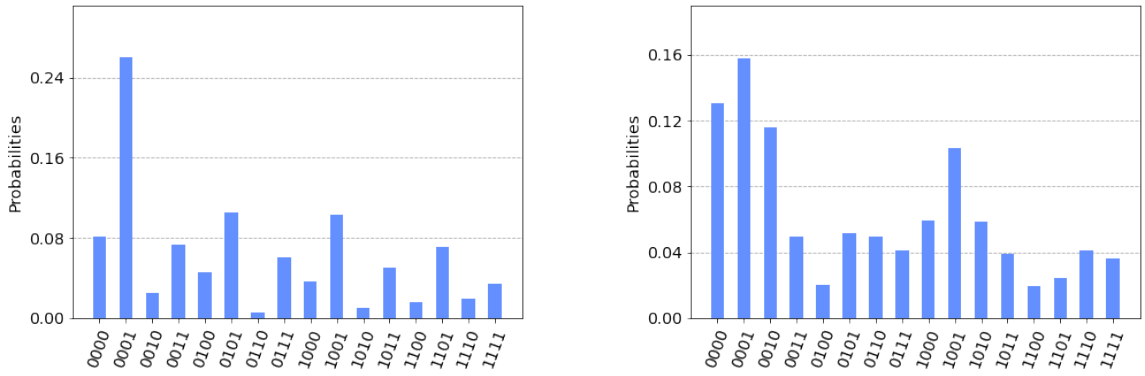
Figure 13: Distributions for U_1 eigenstate $|\psi_1\rangle$ in hardware.

Figures 12 and 13 show a similar behaviour for QFT and AQFT. In the first one, the eigenphase is a little bit on the left of 1001. However, the higher peak is noticeable far; the error coming from the bit-flip of the MSB is so important that the probability of measuring the flipped state is higher than the probability of measuring the right one. This indicates, the effect of noisy gates is the fundamental source of error. In figure 13, the phase is exactly in the middle of '0001' and '0010'. In this case, the noise from the gates is mixed with the noise from the non-representability of the phase. This causes that, on the one hand, the probability is splat in the two closer integer, and on the other hand, the



(a) QFT (b) AQFT

Figure 14: Distributions for U_1 eigenstate $|\psi_0\rangle$ in hardware.



(a) QFT (b) AQFT

Figure 15: Distributions for U_2 eigenstate $|\psi_1\rangle$ in hardware.

noise effect of the real backend causes bitflip for these two integers. As a result of this, we get a wide distribution. In contrast to the previous figure, here the bit-flip on the MSB is not particularly probable, but it is very probable the state '0011' that can be generated by a bit-flip from both '0001' and '0010'.

In the previous figures, the distributions of QFT and AQFT were very similar. In figure 14 and 15 the distributions are a bit different, and globally the AQFT performs a little bit better. However, it is not possible to infer what method minimizes the bit-flips on the MSB, which are the most dangerous, since in Figure 14 QFT is better, and in Figure 15 AQFT is better.

In all these plots of QFT and AQFT, an error on the MSB is highly recurrent. The explanation of this phenomenon is directly related to the circuit they are implemented on. In the $\text{QFT}^\dagger$ block or in its approximate version, many controlled rotations are applied, and the most significant qubit suffers the composite error from all the previous qubits. Although the AQFT version should reduce this error, reducing the dependency of the qubits from the previous ones, it seems that this is not sufficient, watching the results of this experiment. However, this analysis is not complete, and other trials should be done to verify statistically this statement.

To conclude, in experiment 1 the best performance comes from the Iterative method with four success out of four; it follows the Kitaev method with two successes and errors only in the LSB. Then AQFT,

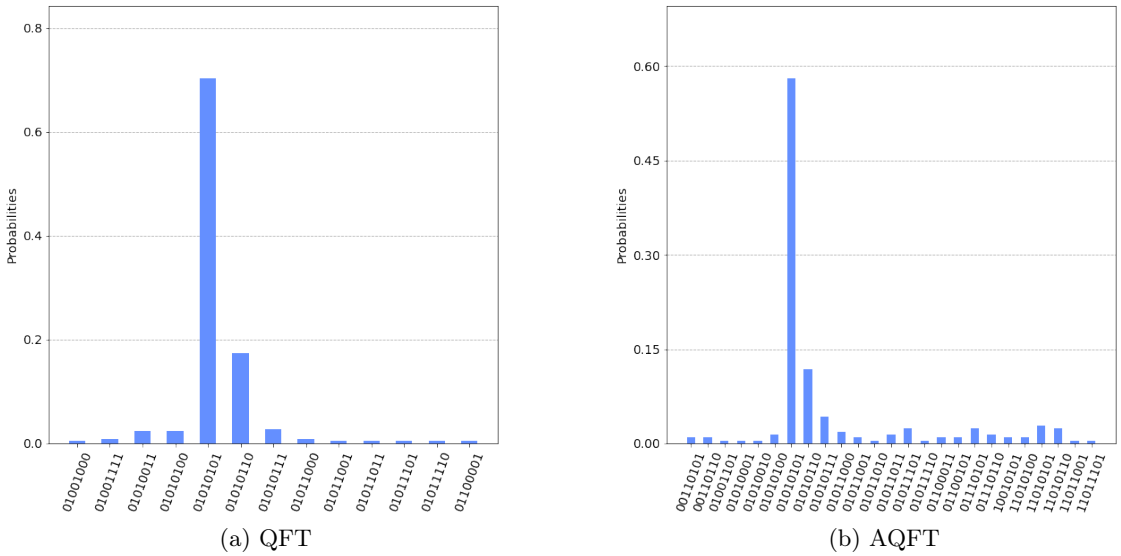
with 2 errors and a high tendency of a bit-flip in the MSB; and lastly QFT, with 3 failures and the higher sensibility to noise.

5.2 Experiment 2

In this experiment, we will test the algorithms on a larger scale: more digits of precision, larger unitaries, and therefore more qubits. Besides that, as we explained earlier, the depth is fixed. We chose the number of shots n_1 (low depth) and n_2 (high depth) shown in table 1 so that the depth is roughly the same for all methods. We will perform the experiments in the simulator with and without noise, and using 8 and 18 digits. The tables with the results can be found in the appendix (tables 6-9), since showing them here would interrupt too much the reading of the report.

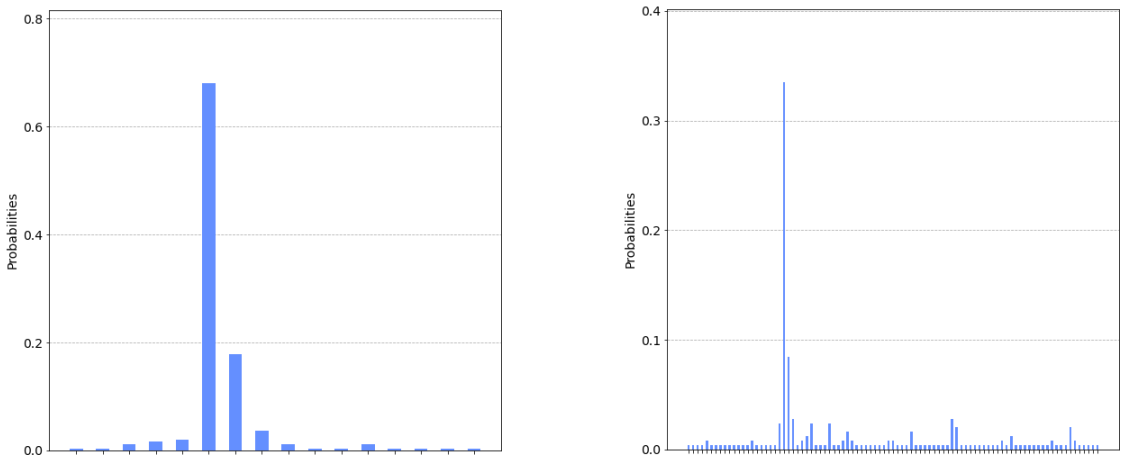
Simulation without noise

Regarding the perfect simulator, the results are not surprising: all methods perform perfectly, getting the closest value possible to the expected one. As in experiment 1, it is interesting to note that for non-representable phases whose remaining is significant, the distributions of QFT and AQFT are pretty different: AQFT's distributions are much wider, due to the imperfections caused by the reduction of the corrections. This happens for both 18 and 8 digits of precision, and it is more visible for high depth (for low depth we have few points so we cannot see any tendency in the distribution). In figures 16 and 17, we show this fact for the first eigenphase (1/3). The other significant case (for the second eigenphase 101/300) is shown in the appendix (figures 29 and 30) , since the distributions are very similar.



(a) QFT (b) AQFT

Figure 16: Distributions for U_3 eigenstate $|\psi_0\rangle$ in perfect simulator with 8 digits of precision.



(a) QFT (b) AQFT

Figure 17: Distributions for U_3 eigenstate $|\psi_0\rangle$ in perfect simulator with 18 digits of precision.

Let us remark that the QFT distributions are in agreement with theory since the probability of success is higher than $\sim 40\%$ and the probability of getting the right result and the closest one is higher than $\sim 80\%$, as we explained in section 2.4.3. For AQFT, we cannot compare the results with the theoretical bound, since we do not fulfil the condition $l \geq \log_2 m + 2$ where in this case $l = 2$ (number of corrections per register) and $m = \{8, 18\}$.

Simulator with noise

The results for the simulator with noise (20-qubit fake hardware) are quite surprising. For low depth, Kitaev's is the only method that is able to give good estimations in almost all eigenphases (excepting the second eigenphase for 18 digits). On the other hand, the iterative can only give one decent result (the first eigenphase for 8 digits). Both QFT and AQFT fail miserably in all the cases, being pretty far from the desired result.

For high depth, the results improve. Kitaev gives us pretty good results in all cases. Both the iterative and AQFT are able to give decent estimations only for the third and fourth eigenphases (545/32768 and 9/16), which are the ones with low and no remaining, respectively. Again, QFT fails miserably and proves that AQFT performs better in noisy backends.

After following this discussion, the reader may wonder why we did not show any result of QFT and AQFT for 18 digits in the noisy simulator. Well, the reason why is pretty simple: the estimation is completely random. All the histograms we got after the noisy simulations are similar to the one shown in figure 18. Even the AQFT, which is supposed to reduce the noise in respect to the QFT, gives us a constant distribution.

Actually, if we take a look at the histograms for 8 digits in the noisy simulator (figures 31-34 in the appendix), we see that the noise is pretty significant. This explains why both methods performed so badly. In fact, it seems like the cases in which the AQFT was able to obtain a decent estimation, it was just a lucky case since the noise is also pretty high.

In conclusion, with this experiment, we have proved that the QFT-based methods perform really badly at big scales. On the other hand, Kitaev algorithm clearly outperformed the iterative one, in contrast to what happened in the first experiment. This may indicate that Kitaev's works better at large scales

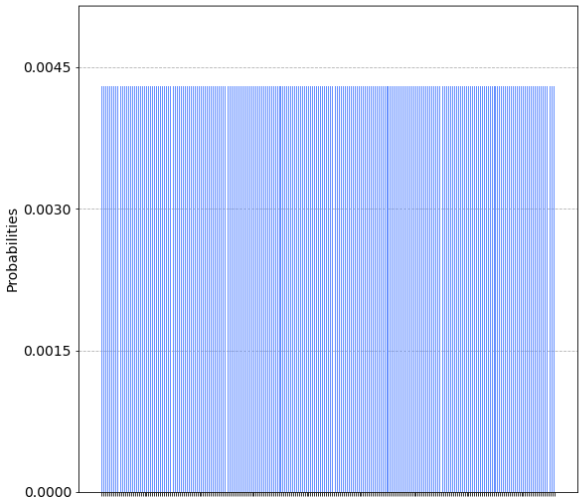


Figure 18: Distribution of the estimations of a single eigenphase for 18 digits of precision in a noisy simulator (AQFT).

than at small ones, and the opposite for the iterative. Nevertheless, in the last experiment, we will go a bit more in-depth in the comparison of these two methods, trying to understand why this seems to happen.

5.3 Experiment 3

In the last experiment we have studied the performance of all methods on a small and large scale and with and without restrictions. Here, we will study how well the estimations given by the methods converge. That is, we will repeat the same estimation many times and analyze the fluctuations of the results. We will use 4 digits of precision and 3 number of repetitions: 10, 30 and 60. This procedure will be done in the simulator for the second eigenphase of $U1$ $(1/16 + 1/32)$ and in IBMQ hardware for both eigenstates of $U1$. All the histograms can be found in the appendix (figures 38-39).

Simulator

With this experiment, we will analyze the probabilistic fluctuations of each method, caused by the fact that the eigenphase cannot be represented exactly with our digits of precision. This phase is between the binary representation 0001 and 0010. The results are shown in figures 35-38.

First of all, Kitaev's has the least fluctuations, since it is the only method that only estimates only two different values, which are the expected ones (the ones in which the real phase is in between). Besides that, we see that QFT concentrates a bit more the probability on the expected values than the AQFT, as we expected and we already proved in the previous experiments. Finally, the IPEA concentrates the probability in the expected values more or less the same as QFT and AQFT. However, there is a non-zero probability of more different results, so it is a bit more scattered.

In conclusion, we can state that the method with the best convergence due to probabilistic imperfections of the method itself is Kitaev's, and with the worst, the iterative.

Hardware

Now we will study the fluctuations only due to using noisy backend (first eigenphase of $U1$, 9/16, which can be perfectly represented with 4 digits) and due to the combination of this noise with the probabilistic one studied above (second eigenphase of $U1$).

The results for the first eigenphase (figures 39 - 42) show again that Kitaev's algorithm is the best in terms of convergence. IPEA also performs really well, concentrating more than 50% of the probability in the right value, with a small fluctuation around it. For the QFT-based methods, the situation is different. The noise has a significant effect, mainly in the MSB. This makes sense, since the rotations applied to that register happen at the end of the circuit, so the propagation of noise causes a larger error there. Actually, for AQFT, the expected phase is wrong (0001 instead of 1001), although for QFT it almost happens the same.

Regarding the second eigenphase (figures 43 - 46), we find some interesting results. As before, Kitaev's has almost probability 1 of giving one of the expected results... but just one of them (recall that the actual phase is between 0001 and 0010). Actually, as we will explain later, it seems that, in hardware, if the phase is between two bitstrings, Kitaev's will always estimate the smaller one. Continuing with the analysis, IPEA has a pretty good convergence, but significantly worse than before, since the estimations with a MSB flip are pretty likely. Moreover, the QFT-based methods give really poor results: the fluctuations are large and the estimated phase is wrong.

From this analysis, we can extract some interesting conclusions. The most obvious one is that Kitaev's method has, by far, the best convergence, regardless of the backend or the representability of the eigenphase. The convergence of IPEA is also really good, although it is highly affected by hardware noise and by the representability of the eigenphases. Regarding the QFT-based methods, the convergence in the presence of noise is bad, even leading to most likely estimations that are wrong, which is compatible with the results of experiment 1.

Experiment with U_2

There seems to be an incompatibility between experiment 1 and 3. In the latter, Kitaev's proved to have the best convergence, with a high probability of getting the right result, while in the former Kitaev's gave us two wrong results. These errors were made in both eigenvalues of U_2. In order to check what went wrong here, we will repeat experiment 3 for U_2, only in the backend and for Kitaev. The results are shown below (figures 19 and 20).

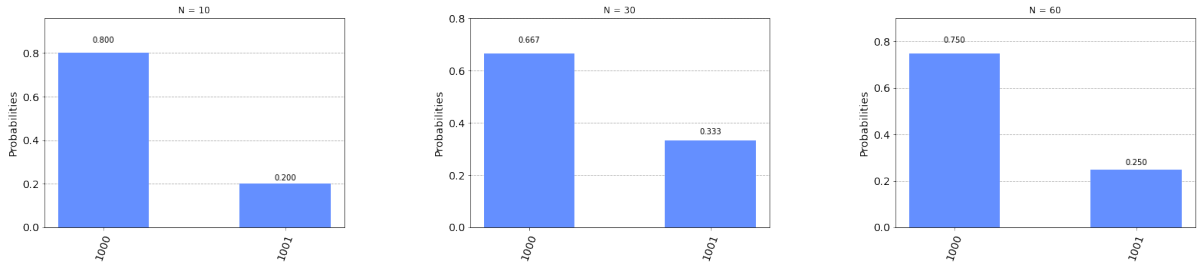
In order to understand the analysis, recall figure 8. The first eigenphase of U_2, $9/16 - 1/64$, is between 1000 and 1001, but significantly closer to 1001. On the other hand, the second eigenphase, $1/16 + 1/64$, is between 0001 and 0010, but closer to 0001. With this in mind, let us look at the histograms now.

Similarly to what happened with U_1, Kitaev's has really small fluctuations. However, as we mentioned before, it seems like Kitaev's algorithm gives us always the bitstring that is on the left of the real value (check figure 8 to understand what 'left' means). This phenomenon only occurs in hardware and there is not a clear explanation why. One cause might be that noise makes Kitaev's algorithm lose track of the remainder of the binary representation of the actual phase. Let us explain this with an example.

The first eigenphase of U_2 has a binary representation of 0.100011 [8]. Therefore, the remainder for four bits of precision is 11. What we propose is that the noise, in some way, makes the algorithm ignore the remainder, so it estimates more frequently 1000, even though the phase is closer to 1001. This phenomenon is only significant for low digits of precision, like in experiment 1, since it only affects the

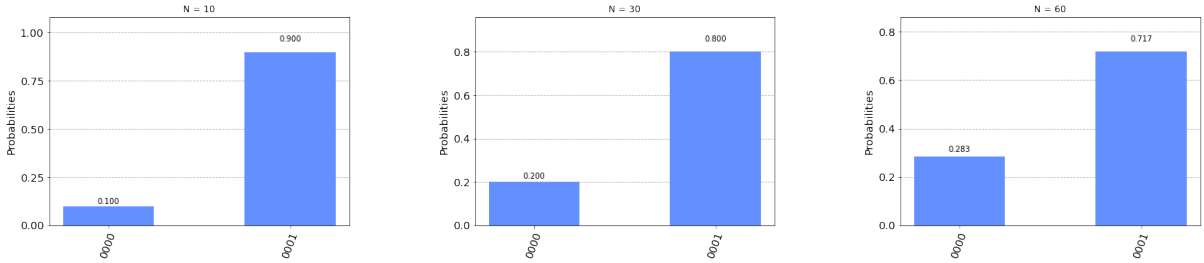
[8] Recall that in the histograms we drop the 0. for simplicity.

LSB.



(a) (b) (c)

Figure 19: Kitaev- Distributions for U_2 eigenstate $|\psi_0\rangle$.



(a) (b) (c)

Figure 20: Kitaev- Distributions for U_2 eigenstate $|\psi_1\rangle$.

6 Conclusion

Let us summarize all the results we have obtained throughout this dense project. First off, in experiment 1 we analyzed how different kind of errors (probabilistic inherent errors caused by the non-representability of the eigenphases and random errors caused by the noisy backend) affect all the methods and their estimations. The perfect simulator enabled us to study the first kinds of error alone, while hardware the second one and the combination of both. The main conclusion was that QFT-based methods were really sensitive to noise, causing them to have wide distributions with the highest peak being a wrong phase sometimes. This indicates that it seems to be better not to have an algorithm based on a large circuit with many control gates, since noise easily propagates through it. Moreover, we believed that IPEA was the best in terms of precision since it did not fail any estimation. However, it was too soon for a final conclusion.

In experiment 2 we ran the algorithms on a larger scale (more qubits, digits of precision, and larger unitaries) with a 'time limit', to see how the methods performed compared to the first one. It was confirmed that QFT-methods were the worst overall in noisy backends, as they failed miserably most of the times. Apart from that, we were shocked by the great precision of Kitaev's estimations, while IPEA performed pretty bad frequently. After this experiment, we were confused about how Kitaev's was not the best performing in experiment 1.

Finally, in experiment 3 we analyzed the fluctuations of each method. The QFT-based methods had the most fluctuations, especially in hardware, as it was proven in the former experiments. Moreover, Kitaev's showed almost no fluctuations compared to the rest. However, we realized there was something wrong when the eigenphase was not representable with the given precision: Kitaev's always chooses

the bitstring on the left of the actual value of the phase, regardless of its position (even if it was closer to the right bitstring!). This fact was proven in Kitaev's histograms of U_2 (figures 19 and 20).

Therefore, as a general conclusion, we can say that given the current noise levels of quantum computers (at least, of the ones we could test), algorithms with classical post-processing, which avoid the overuse of control operations, perform better.

6.1 Possible future expansions

This has been a really dense and fruitful project, both in terms of theory and implementation, in which we have been able to grasp many concepts on how one of the most powerful quantum algorithm works. Nevertheless, we have several ideas to extend the project an keep working towards a better understanding of the QPE problem.

Firstly, it would be interesting to address the superposition of eigenstates as an input. This is strongly associated with the fact of being able to send an arbitrary state as an input, which of course is of great interest in terms of applications. On the other hand, we would also like to find an explanation why Kitaev's algorithm, in real hardware, tends to estimate the bitstring which is on the left of the actual eigenphase, as we explained in experiment 3. For example, we could try to run this algorithm in other real backends with more qubits to see if this is a general tendency, or if it only happens in the 5-qubits computers. Furthermore, at the end of this project, we still have many doubts and curiosities about the fields that can be improved by QPE. Considering that the BQP-problems can be rewritten in the QPE formalism [16], we would like to understand how famous algorithms are translated into it. In these problems, QPE would assume the role of a logic building block in a bigger architecture. It would be interesting to grasp what the main characteristics of these QPE blocks are (such as input and output states) and their relationship with the rest of the logic. Questions that we would like to answer are: does the *ancilla register* receive in input an eigenstate or a general superposition? Is the estimated phase the only information used at the end of the QPE block or is it also important the state that the *ancilla register* assumes after the estimation? Understanding how QPE serves the most important algorithms, could allow us to abstract new variations and optimizations and, why not, totally undiscovered BQP-problems!

References

[1] Alan Aspuru-Guzik, Anthony D. Dutoi, Peter J. Love, and Martin Head-Gordon. Simulated quantum computation of molecular energies. *Science*, 309(5741):1704–1707, Sep 2005.

[2] Miroslav Dobšíček, Göran Johansson, Vitaly Shumeiko, and Göran Wendin. Arbitrary accuracy iterative quantum phase estimation algorithm using a single ancillary qubit: A two-qubit benchmark. *Phys. Rev. A*, 76:030306, Sep 2007.

[3] Miroslav Dobšíček. Quantum computing, phase estimation and applications, 2008.

[4] Chen-Fu Chiang Hamed Ahmadi. Quantum phase estimation with arbitrary constant-precision phase shift operators. *Quantum Information and Computation*, 12(9,10):14, 09 2011.

[5] IBM-qiskit. https://qiskit.org/documentation/tutorials/algorithms/09_IQPE.html.

[6] IBM-qiskit. https://qiskit.org/textbook/ch-labs/Lab04_IterativePhaseEstimation.html.

[7] A.Y. Kitaev, A. Shen, M.N. Vyalyi, and M.N. Vyalyi. *Classical and Quantum Computation*. Graduate studies in mathematics. American Mathematical Society, 2002.

[8] Hamed Mohammadbagherpoor, Young-Hyun Oh, Patrick Dreher, Anand Singh, Xianqing Yu, and Andy J. Rindos. Improved implementation approach for quantum phase estimation on quantum computers, 2019.

[9] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[10] Siyuan Niu, Adrien Suau, Gabriel Staffelbach, and Aida Todri-Sanial. A hardware-aware heuristic for the qubit mapping problem in the nisq era. *IEEE Transactions on Quantum Engineering*, 1:9, 09 2020.

[11] Thomas E O'Brien, Brian Tarasinski, and Barbara M Terhal. Quantum phase estimation of multiple eigenvalues for small-scale (noisy) experiments. *New Journal of Physics*, 21(2):023022, Feb 2019.

[12] R. Laflamme P. Kaye and M. Mosca. *An Introduction to Quantum Computing*. Oxford University Press, 2007.

[13] Krysta M. Svore, Matthew B. Hastings, and Michael Freedman. Faster phase estimation, 2013.

[14] Ben Travaglione and GJ Milburn. Generation of eigenstates using the phase-estimation algorithm. *Physical Review A*, 63, 03 2001.

[15] Nathan Wiebe and Chris Granade. Efficient bayesian phase estimation. *Phys. Rev. Lett.*, 117:010503, Jun 2016.

[16] Pawel Wocjan and Shengyu Zhang. Several natural bqp-complete problems, 2006.

7 Appendix

For denoting the eigenstates, we will adopt the notation $|\psi_i\rangle = |i\rangle$ where $|i\rangle$ is referred to the computational basis. This way, it would be easier to understand of which eigenphase we are talking about.

7.1 Code

Code and documentation can be found on the projects GitHub page.

7.2 Experiment 1

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.5625	0.5625	0.5625	0.5625	0.5625
Bitstring	1001	1001	1001	1001	1001
Decimal	0.5625	0.5625	0.5625	0.0625	0.5625
Bitstring	1001	1001	1001	0001	1001

(a) $\varphi = \frac{9}{16}$

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.0937	0.0625	0.1250	0.0625	0.1250
Bitstring	0001-0010	0001	0010	0001	0010
Decimal	0.0937	0.0625	0.1250	0.0625	0.1875
Bitstring	0001-0010	0001	0010	0011	0011

(b) $\varphi = \frac{1}{16} + \frac{1}{32}$

Table 4: U_1 for and • simulator and • IBMQ.

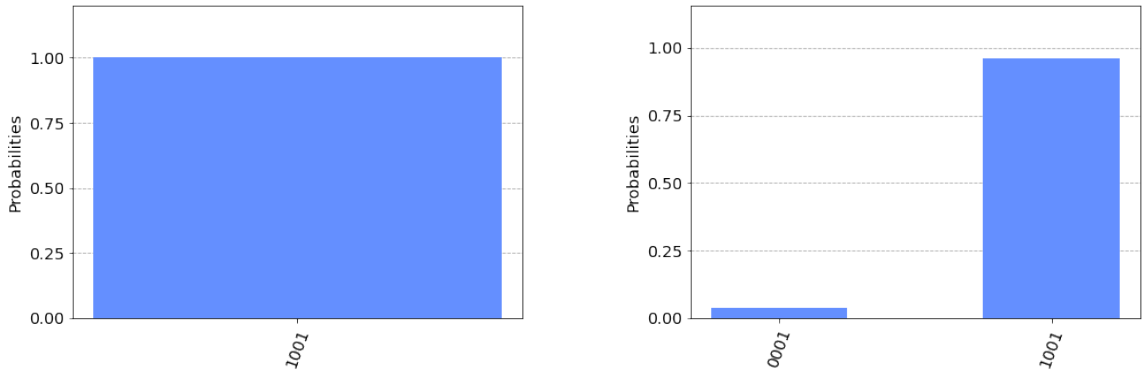
	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.5468	0.5625	0.5625	0.5625	0.5625
Bitstring	1001	1001	1001	1001	1001
Decimal	0.5468	0.500	0.5625	0.0625	0.0625
Bitstring	1001	1000	1001	0001	0001

(a) $\varphi = \frac{9}{16} - \frac{1}{64}$

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.0781	0.0625	0.0625	0.0625	0.0625
Bitstring	0001	0001	0001	0001	0001
Decimal	0.0781	0.0000	0.0625	0.0625	0.0625
Bitstring	0001	0000	0001	0001	0001

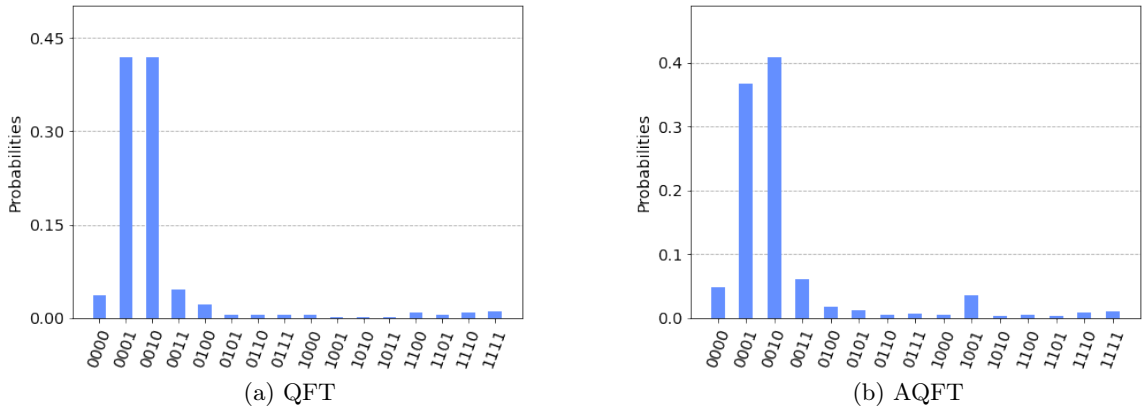
(b) $\varphi = \frac{1}{16} + \frac{1}{64}$

Table 5: U_2 for and ● simulator and ● IBMQ.



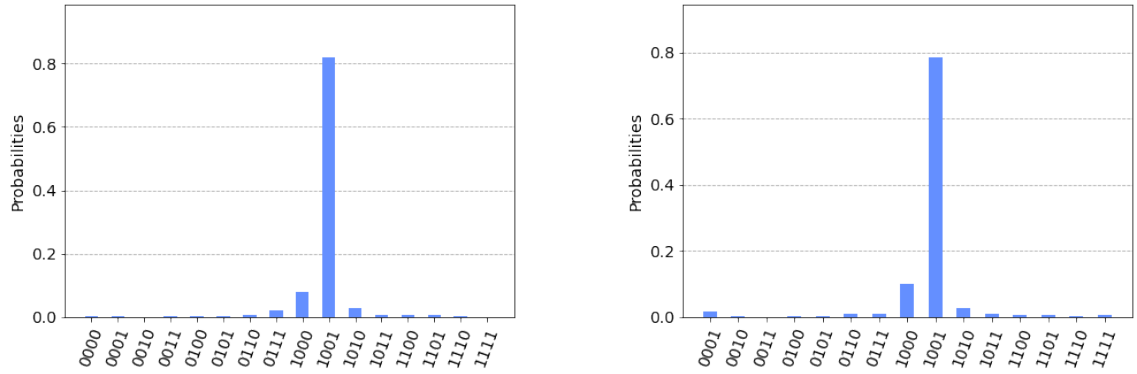
(a) QFT (b) AQFT

Figure 21: Distributions for U_1 eigenstate $|\psi_0\rangle$ in simulator.



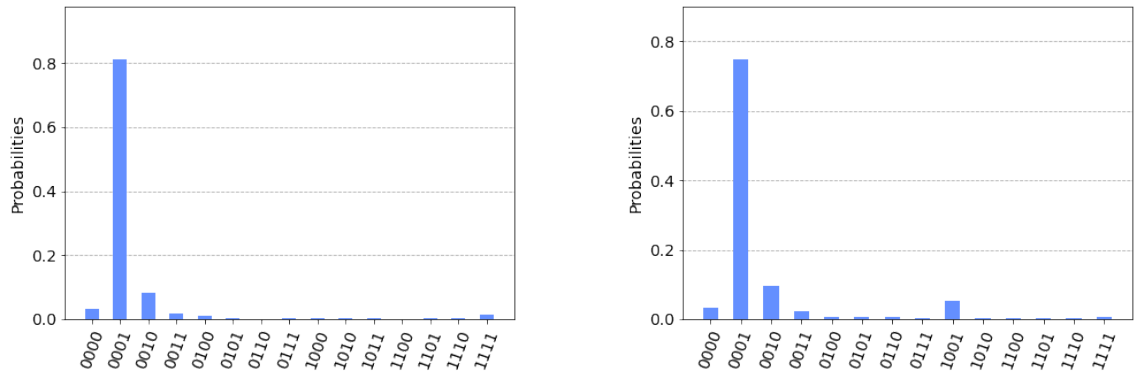
(a) QFT (b) AQFT

Figure 22: Distributions for U_1 eigenstate $|\psi_1\rangle$ in simulator.



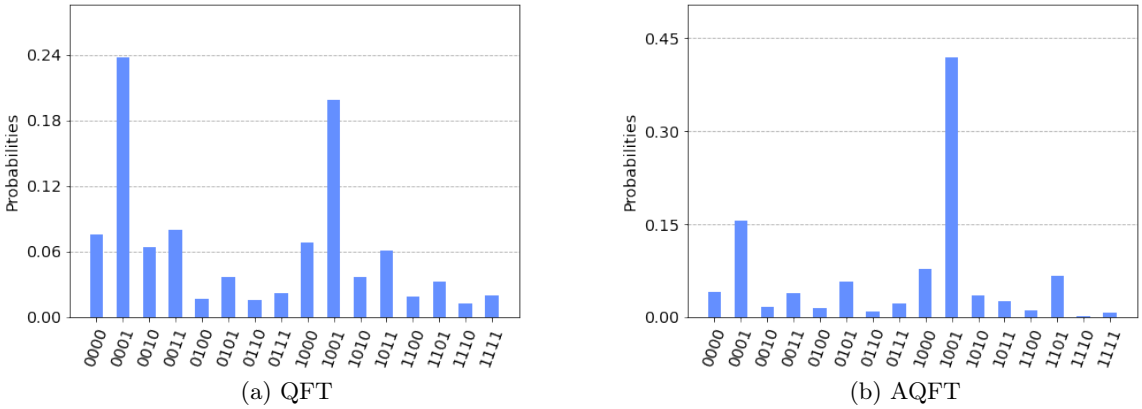
(a) QFT (b) AQFT

Figure 23: Distributions for U_2 eigenstate $|\psi_0\rangle$ in simulator.



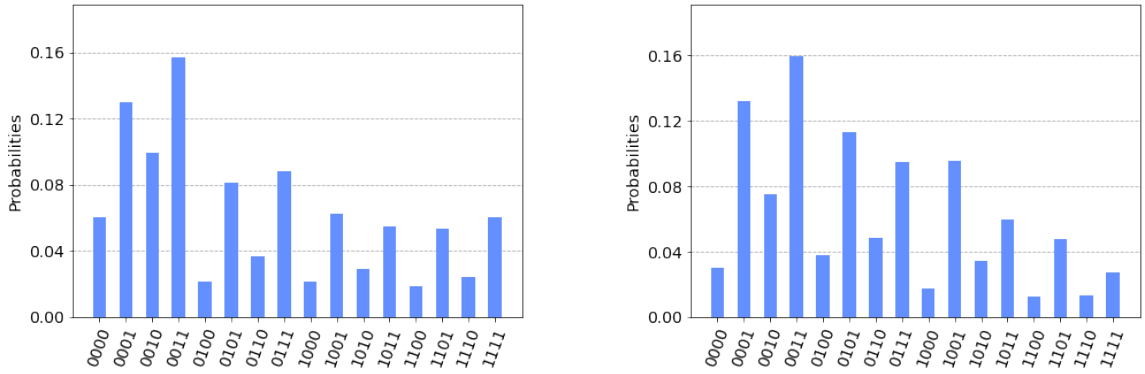
(a) QFT (b) AQFT

Figure 24: Distributions for U_2 eigenstate $|\psi_1\rangle$ in simulator.



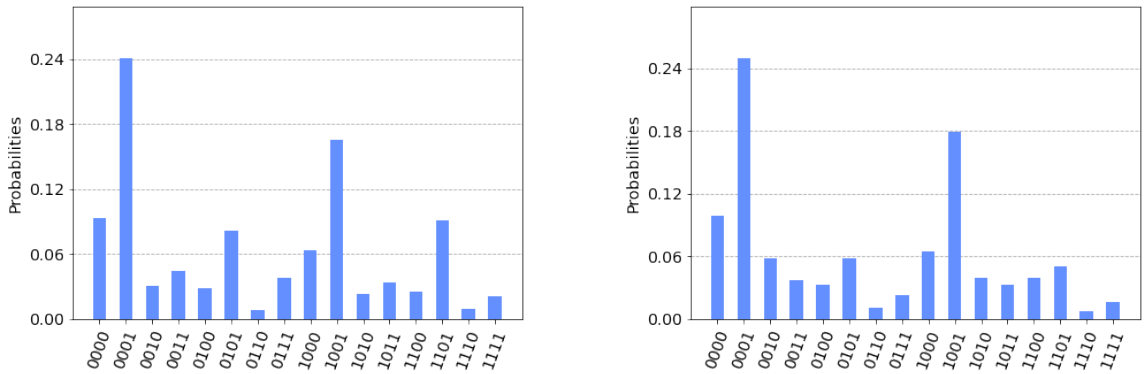
(a) QFT (b) AQFT

Figure 25: Distributions for U_1 eigenstate $|\psi_0\rangle$ in hardware.



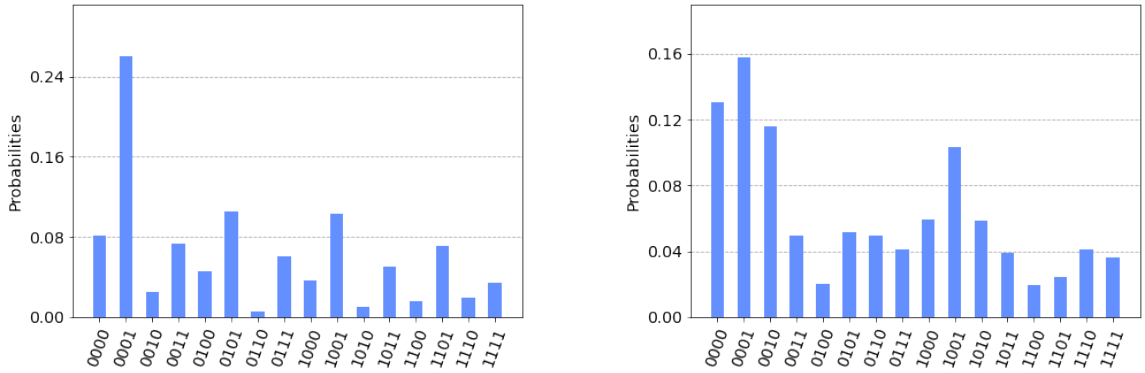
(a) QFT (b) AQFT

Figure 26: Distributions for U_1 eigenstate $|\psi_1\rangle$ in hardware.



(a) QFT (b) AQFT

Figure 27: Distributions for U_2 eigenstate $|\psi_0\rangle$ in hardware.



(a) QFT (b) AQFT

Figure 28: Distributions for U_2 eigenstate $|\psi_1\rangle$ in hardware.

7.3 Experiment 2

7.3.1 Tables

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.3333	0.3320	0.3320	0.3320	0.3320
Bitstring	01010101	01010101	01010101	01010101	01010101
Decimal	0.3333	0.3281	0.3164	0.0078	0.9023
Bitstring	01010101	01010100	01010001	00000010	11100111

(a) $\varphi = \frac{1}{3}$

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.3366	0.3359	0.3359	0.3359	0.3359
Bitstring	01010110	01010110	01010110	01010110	01010110
Decimal	0.3366	0.8398	0.0859	0.1367	0.1289
Bitstring	01010110	11010111	00010110	00100011	100001

(b) $\varphi = \frac{101}{300}$

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	$1,441 \cdot 10^{-5}$	0.0000	0.0000	0.0000	0.0000
Bitstring	00000000	00000000	00000000	00000000	00000000
Decimal	$1,441 \cdot 10^{-5}$	0.0000	0.3906	0.5675	0.4218
Bitstring	00000000	00000000	00100100	10010000	01101100

(c) $\varphi = \frac{545}{32768}$

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.5625	0.5625	0.5625	0.5625	0.5625
Bitstring	10010000	10010000	10010000	10010000	10010000
Decimal	0.5625	0.5625	0.2812	0.3476	0.1953
Bitstring	10010000	10010000	1001000	01011001	00110010

(d) $\varphi = \frac{9}{16}$

Table 6: U_3 8 digits of precision and low depth for • perfect and • noisy simulator.

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.3333	0.3320	0.3320	0.3320	0.3320
Bitstring	01010101	01010101	01010101	01010101	01010101
Decimal	0.3333	0.3359	0.3789	0.6562	0.3438
Bitstring	01010101	1010110	01100001	10101000	01011000

(a) $\varphi = \frac{1}{3}$

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.3366	0.3359	0.3359	0.3359	0.3359
Bitstring	01010110	01010110	01010110	01010110	01010110
Decimal	0.3366	0.3359	0.5859	0.2773	0.4687
Bitstring	01010110	01010110	10010110	10000111	1111000

(b) $\varphi = \frac{101}{300}$

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	$1,441 \cdot 10^{-5}$	0.0000	0.0000	0.0000	0.0000
Bitstring	00000000	00000000	00000000	00000000	00000000
Decimal	$1,441 \cdot 10^{-5}$	0.0000	0.0000	0.5625	0.0039
Bitstring	00000000	00000000	00000000	100010000	00000001

(c) $\varphi = \frac{545}{32768}$

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.5625	0.5625	0.5625	0.5625	0.5625
Bitstring	10010000	10010000	10010000	10010000	10010000
Decimal	0.5625	0.5625	0.5625	0.3750	0.5664
Bitstring	10010000	10010000	10010000	1100000	10010000

(d) $\varphi = \frac{9}{16}$

Table 7: U_3 8 digits of precision and high depth for • perfect and • noisy simulator.

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.3333	0.3333	0.3333	0.3333	0.3333
Decimal	0.3333	0.3333	0.2236	-	-

(a) $\varphi = \frac{1}{3}$

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.3366	0.3366	0.3366	0.3366	0.3366
Decimal	0.3366	0.0866	0.7590	-	-

(b) $\varphi = \frac{101}{300}$

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	$1,441 \cdot 10^{-5}$	$1,526 \cdot 10^{-5}$	$1,526 \cdot 10^{-5}$	$1,526 \cdot 10^{-5}$	$1,526 \cdot 10^{-5}$
Decimal	$1,441 \cdot 10^{-5}$	0.0078	0.3045	-	-

(c) $\varphi = \frac{545}{32768}$

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.5625	0.5625	0.5625	0.5625	0.5625
Decimal	0.5625	0.5783	0.2000	-	-

(d) $\varphi = \frac{9}{16}$

Table 8: U_3 18 digits of precision and low depth for • perfect and • noisy simulator.

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.3333	0.3333	0.3333	0.3333	0.3333
Decimal	0.3333	0.3333	0.1445	-	-

(a) $\varphi = \frac{1}{3}$

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.3366	0.3366	0.3366	0.3366	0.3366
Decimal	0.3366	0.3366	0.5319	-	-

(b) $\varphi = \frac{101}{300}$

	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	$1,441 \cdot 10^{-5}$	$1,526 \cdot 10^{-5}$	$1,526 \cdot 10^{-5}$	$1,526 \cdot 10^{-5}$	$1,526 \cdot 10^{-5}$
Decimal	$1,441 \cdot 10^{-5}$	$1,526 \cdot 10^{-5}$	0.0000	-	-

(c) $\varphi = \frac{545}{32768}$

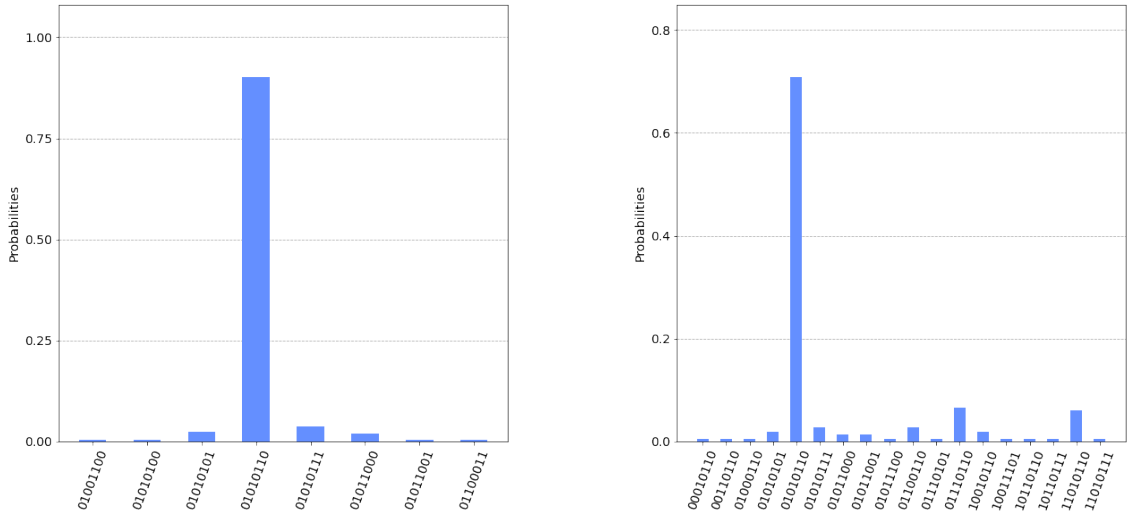
	Expected	Kitaev	Iterative	QFT	AQFT
Decimal	0.5625	0.5625	0.5625	0.5625	0.5625
Decimal	0.5625	0.5625	0.5625	-	-

(d) $\varphi = \frac{9}{16}$

Table 9: U_3 18 digits of precision and high depth for • perfect and • noisy simulator.

7.3.2 Figures

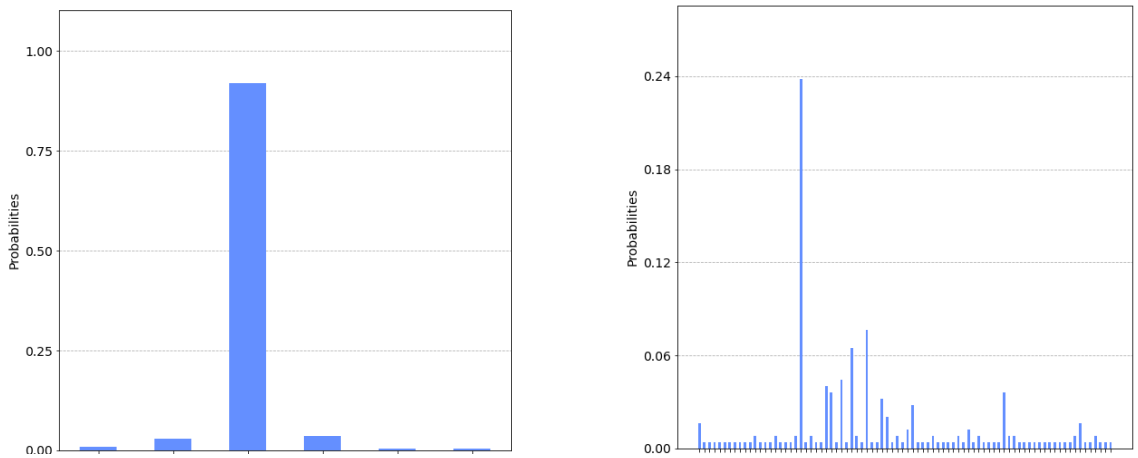
Perfect Simulator



(a) QFT

(b) AQFT

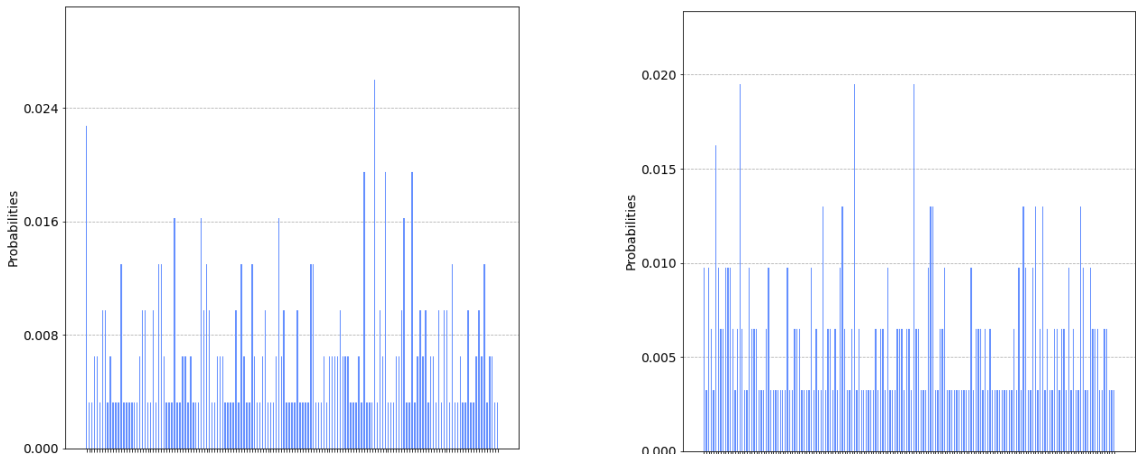
Figure 29: Distributions for U_3 eigenstate $|\psi_1\rangle$ in perfect simulator with 8 digits of precision.



(a) QFT (b) AQFT

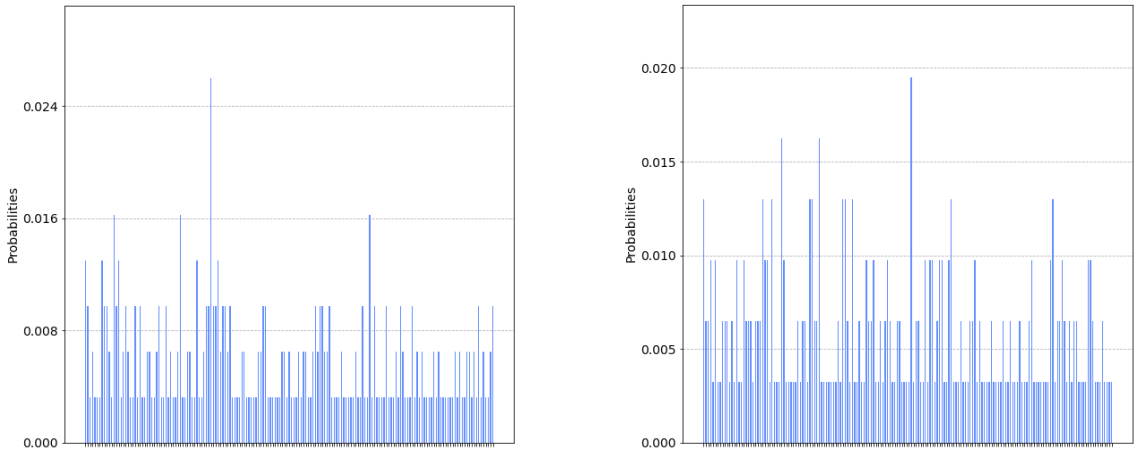
Figure 30: Distributions for U_3 eigenstate $|\psi_1\rangle$ in perfect simulator with 18 digits of precision.

Noisy Simulator



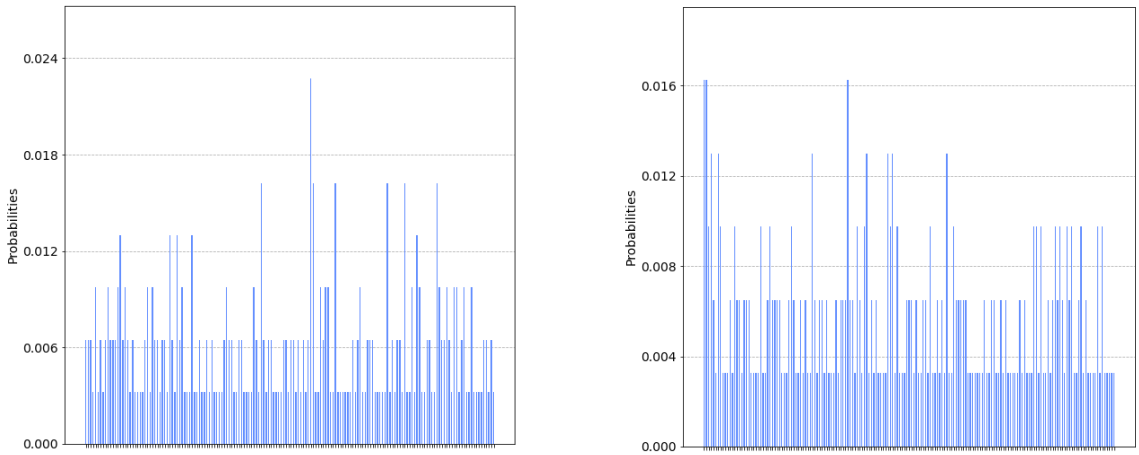
(a) QFT (b) AQFT

Figure 31: Distributions for U_3 eigenstate $|\psi_0\rangle$ in noisy simulator with 8 digits of precision.



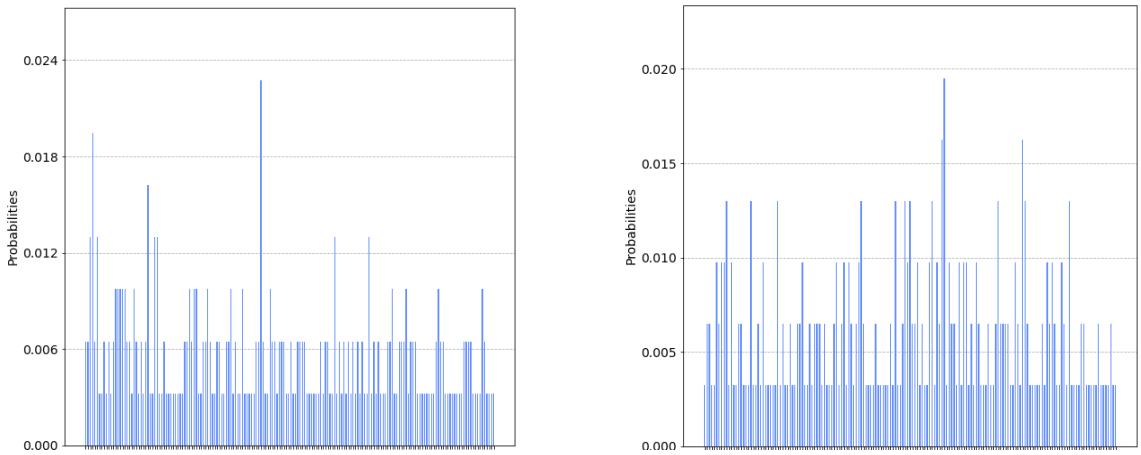
(a) QFT (b) AQFT

Figure 32: Distributions for U_3 eigenstate $|\psi_1\rangle$ in noisy simulator with 8 digits of precision.



(a) QFT (b) AQFT

Figure 33: Distributions for U_3 eigenstate $|\psi_2\rangle$ in noisy simulator with 8 digits of precision.

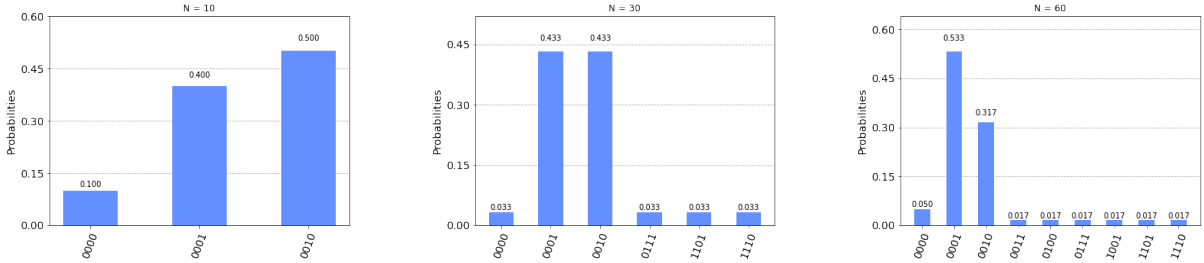


(a) QFT (b) AQFT

Figure 34: Distributions for U_3 eigenstate $|\psi_3\rangle$ in noisy simulator with 8 digits of precision.

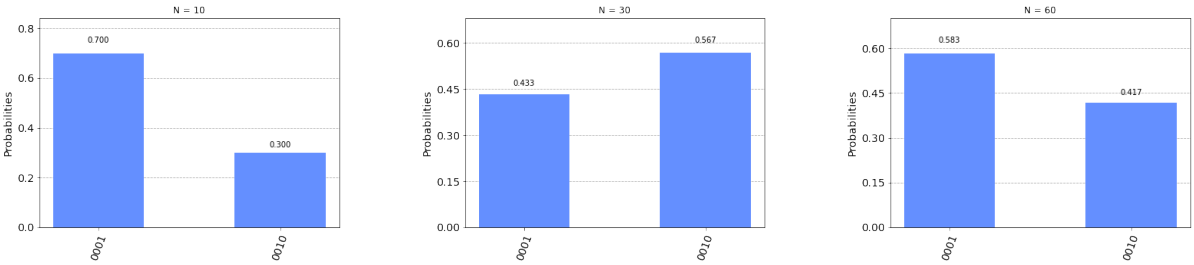
7.4 Experiment 3

Simulation



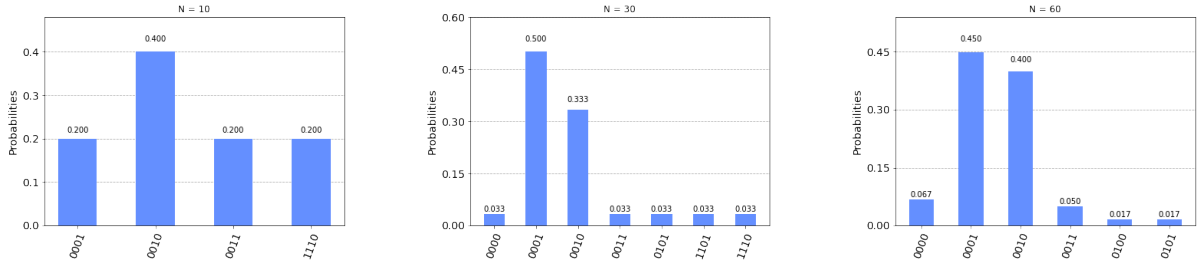
(a) (b) (c)

Figure 35: IPEA - U_1 Eigenstate $|\psi_1\rangle$



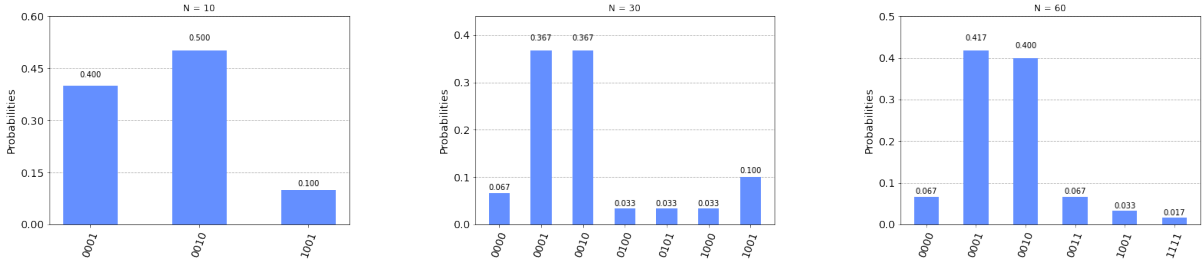
(a) (b) (c)

Figure 36: Kitaev - U_1 Eigenstate $|\psi_1\rangle$



(a) (b) (c)

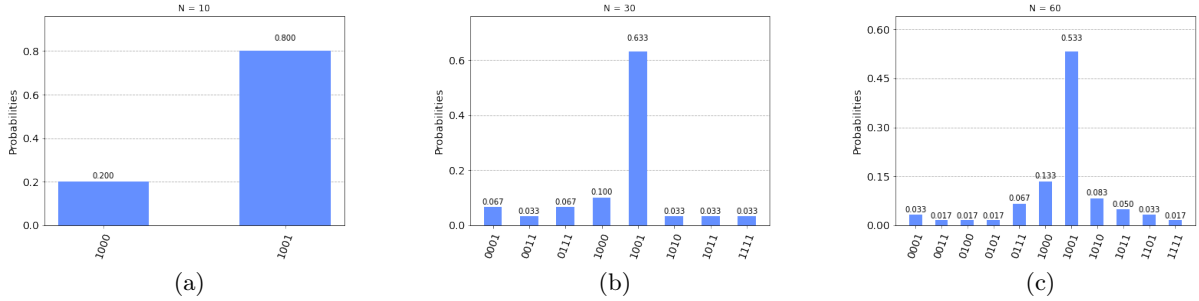
Figure 37: QFT - U_1 Eigenstate $|\psi_1\rangle$



(a) (b) (c)

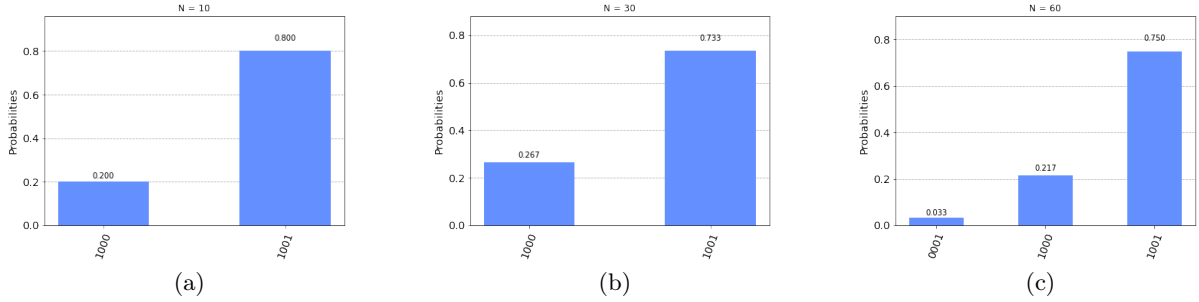
Figure 38: AQFT - U_1 Eigenstate $|\psi_1\rangle$

Hardware



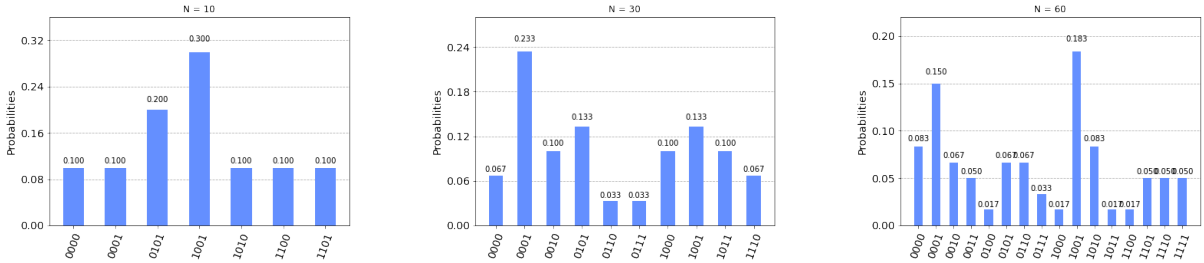
(a) (b) (c)

Figure 39: IPEA - U_1 Eigenstate $|\psi_0\rangle$



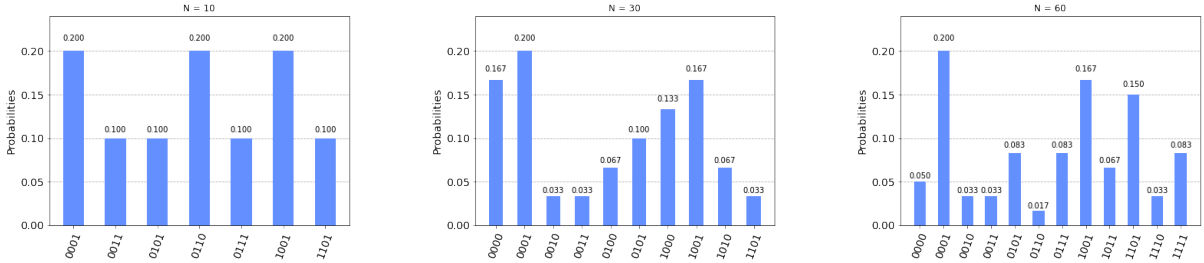
(a) (b) (c)

Figure 40: Kitaev - U_1 Eigenstate $|\psi_0\rangle$



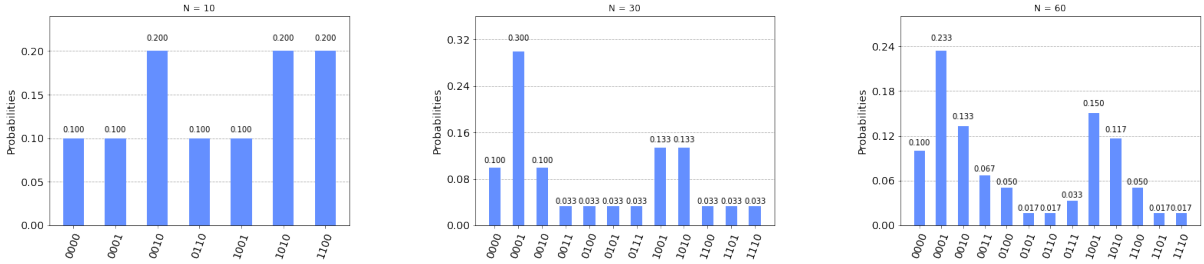
(a) (b) (c)

Figure 41: QFT - U_1 Eigenstate $|\psi_0\rangle$



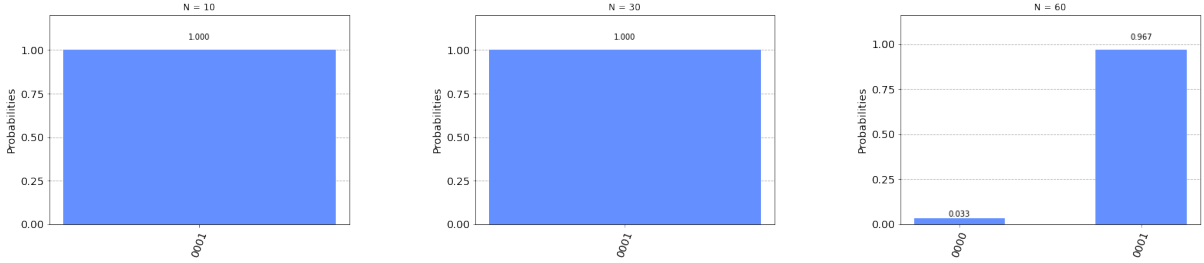
(a) (b) (c)

Figure 42: AQFT - U_1 Eigenstate $|\psi_0\rangle$



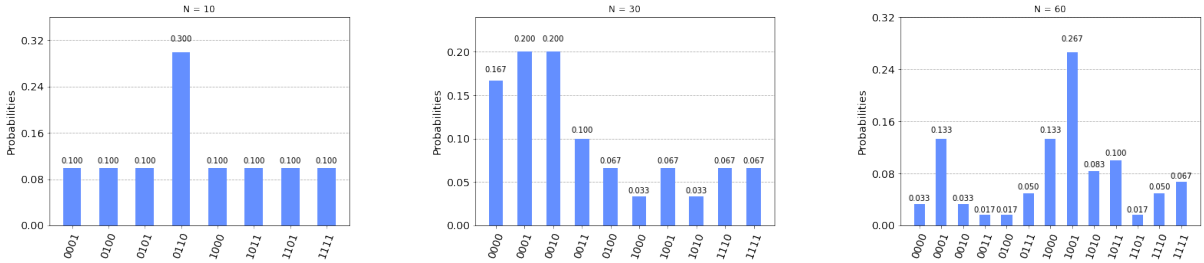
(a) (b) (c)

Figure 43: IPEA - U_1 Eigenstate $|\psi_1\rangle$



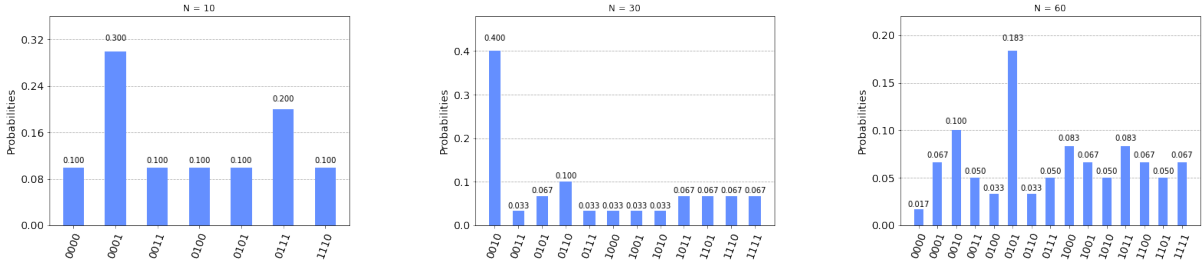
(a) (b) (c)

Figure 44: Kitaev - U_1 Eigenstate $|\psi_1\rangle$



(a) (b) (c)

Figure 45: QFT - U_1 Eigenstate $|\psi_1\rangle$



(a) (b) (c)

Figure 46: AQFT - U_1 Eigenstate $|\psi_1\rangle$