Kitaev phase estimation

N. Bohlsen
October 2021

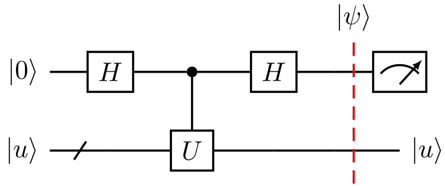
I. Introduction

A common problem of interest in quantum computing is phase estimation, whose problem statement is as follows. Consider a quantum oracle U and a state $|u\rangle$ which we know is an eigenstate of the oracle. Since quantum circuits must be reversible, all quantum oracles must be unitary. Therefore their eigenvalues must be of the form $e^{2\pi i\varphi}$. That is, the operation of an oracle on one of its eigenkets must be a global phase rotation. In phase estimation, we are given U and a particular $|u\rangle$ and we want to determine φ [1]. It is important to note that this must be achieved without destroying the state $|u\rangle$ and hence no measurements can be taken of this state.

There are two methods to compute φ: one of which relies on the Quantum Fourier Transform (QFT); where the other focuses on qubit statistics [1]. This document will look at the statistical method called Kitaev's phase estimation algorithm. It will also discuss the implementation of this algorithm in the Q# quantum programming language.

II. Motivation

Consider the quantum circuit



,

in which a single ancillary qubit is: passed through a Hadamard (H) gate, controls a U operation on the eigenket $|u\rangle$, is passed through another H gate, and then is finally measured. We recognise this as a circuit which will produce phase kickback [1]. The initial state of the qubits is $|0\rangle\,|u\rangle$ and simple algebra gives the state prior to measurement as

$$|\psi\rangle = \left(e^{\pi i\varphi}\cos\pi\varphi\,|0\rangle - ie^{\pi i\varphi}\sin\pi\varphi\,|1\rangle\right)|u\rangle\ . \tag{1}$$

Thus, the probability that the ancillary qubit will be measured as either a 0 or a 1 is given by $|e^{\pi i\varphi}\cos\pi\varphi|^2$ and $|-ie^{\pi i\varphi}\sin\pi\varphi|^2$ respectively. Which we rewrite as

$$P(0) = \frac{1+\cos 2\pi\varphi}{2}, P(1) = \frac{1-\cos 2\pi\varphi}{2}\ .$$

The circuit has encoded the eigenphase of $|u\rangle$ into the measurement probabilities of the ancillary qubit. Therefore, it we repeat this circuit many times and measure the ancillary qubit each time we can determine an estimate to $P(0)$, which we name $\tilde{P}(0)$, and can then estimate the cosine term as $\cos 2\pi\varphi \approx 2\tilde{P}(0) - 1$.

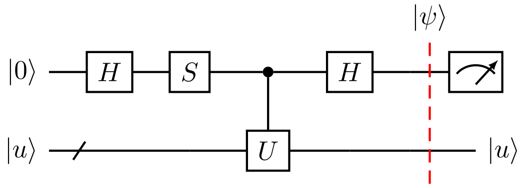
The above demonstrates the fundamental idea of the Kitaev algorithm. That is, we use phase kickback to encode φ in the probability amplitudes of an ancillary qubit and then by repeating the circuit many times we experimentally measure the relevant statistical probabilities and deduce φ from them.

The benefit of this algorithm is that, since the one ancillary qubit can be reused after being measured, only one ancillary qubit is ever required and yet arbitrary precision can be achieved[1]. Specifically, since each measurement is independent, if we iterate this process N times the standard error in our measurement should scale with $\frac{1}{\sqrt{N}}$ similar to how the standard error in the mean scales with the number of measurements [2]. This constitutes a distinct advantage over the QFT based algorithm which requires fewer gates but many more qubits. In fact, the number of qubits needed for the QFT version scales linearly with the number of bits of precision required [1], where for Kitaev's algorithm it remains constant at 1. However, because the eigenphase is only encoded as a probability in the Kitaev algorithm, certainty about the solution is lost, although this is more of a theoretical restriction than a practical one.

III. Some mathematical notes

A key observation regarding the above, is that φ is encoded in the probabilities through $\cos 2\pi\varphi$ This is actually a problem. Since $\cos$ is an even function, the circuit above cannot distinguish between φ and $-\varphi$. We can fix this by designing a circuit which instead encodes φ in an estimate of $\sin 2\pi\varphi$ and then use the two estimations in conjunction[2].

To incorporate a $\sin$ function in the probability amplitudes we need to force the two components of the ancillary qubit to be out of phase by a complex factor [3]. That is, we need insert a $Z(\theta) = \left(\begin{smallmatrix}1 & 0\\ 0 & e^{i\theta}\end{smallmatrix}\right)$ gate between the H and controlled U gate [4]. Technically, the choice of angle here is arbitrary but it is most convenient, and still sufficient, to choose a $Z(\pi/2) = S$ gate. So our second circuit is



.

Again, some simple algebra gives a state prior to measurement

[1] At least in principle

[2] We need to use both estimations because the sin function suffers its own restriction due to its periodicity. Specifically, it cannot distinguish between the left and right halves of a unit circle.

of

$$|\psi\rangle = \left(\frac{1+ie^{2\pi i\varphi}}{2}\,|0\rangle + \frac{1-ie^{2\pi i\varphi}}{2}\,|1\rangle\right)|u\rangle\ , \qquad (2)$$

which then gives measurement probabilities for the ancillary qubit of

$$P'(0) = \frac{1-\sin 2\pi\varphi}{2}, P'(1) = \frac{1+\sin 2\pi\varphi}{2}\ .$$

So, we can see that by repeating this circuit and building up an estimation for the probability of the qubit to be in the $|0\rangle$ state $\tilde{P}'(0)$ we can deduce an approxmation for the $\sin$ of the eigenphase [2].

We now need to determine the best estimation of an angle from its sine an cosine. According to [4] the best choice is the arctangent itself[3]. I will note that [4] provides no justification for this conclusion and does not cite a source so this result cannot be verified. It is however a convenient choice and was used for the implementation of the algorithm. Thus, the Kitaev estimation for the eigenphase can be taken to be

$$\tilde{\varphi} = \frac{1}{2\pi}\arctan\left(\frac{1-2\tilde{P}'(0)}{2\tilde{P}(0)-1}\right)\ . \qquad (3)$$

IV. Simulation results and scaling of the error

After implementing the Kitaev algorithm in Q# a series of tests were run to determine if the program was performing correctly and that the algorithm scaled as expected. Note that the program is constructed so that the U gate is a passable input and hence the choice of U gate for the tests is entirely arbitrary. Hence, for simplicity as a first test it was chosen to take $U = R_z(2\cdot 2\pi\varphi)$. For this gate, $|1\rangle$ is an eigenket with eigenvalue $e^{2\pi\varphi}$. So, this is a gate and eigenstate which are both easy to prepare and which are also sufficient to demonstrate that the phase estimation process produces the correct eigenphase.

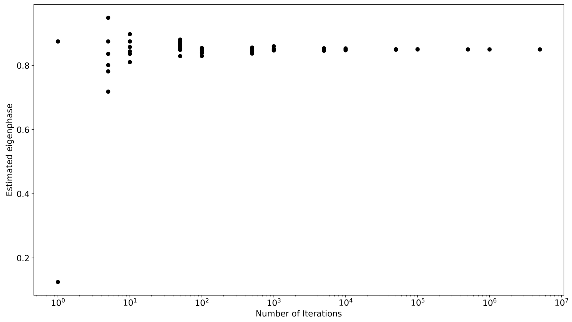


Fig. 1. Simulated results of the Kitaev phase estimation algorithm for φ = 0.85.

Figure 1 presents the results of simulating the Kitaev process for the oracle described above with $\varphi = 0.85$ against the number of measurements[4] N.

For larger numbers of tests the results clearly converge around the correct estimation for the eigenphase. This demonstrates that the Kitaev phase estimation process does function as intended, in that, it can correctly estimate the eigenphase of an eigenket.

To ensure that the error scaled as expected the standard deviation from the simulated dataset was computed and was plotted against N. The log-log plot of this is shown as Figure 2 along with the best linear regression to the data. The slope of this line is $m \approx -0.534$ which we recognise as ≈ -0.5. Hence, this simulation constitutes strong evidence that the expected error σ in the estimate $\tilde{\varphi}$ scales with the inverse of the square root of N. That is

$$\sigma \propto \frac{1}{\sqrt{N}},$$

which confirms the expected error scaling mentioned previously.

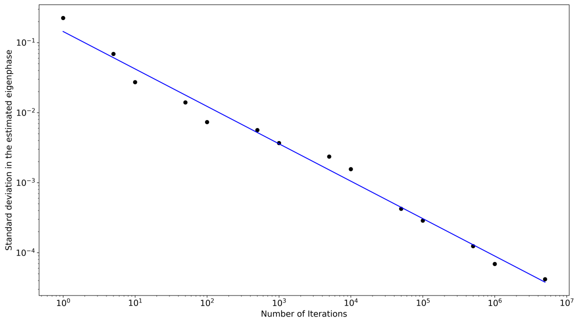


Fig. 2. Scaling of the error for Kitaev phase estimation.

V. A note on interpreting program outputs

Recall that we defined the eigenphase φ by associating it with an overall complex phase factor of $e^{2\pi i\varphi}$. Hence, we see that we need to identify $\varphi = 0$ and $\varphi = 1$ as the same point. That is, the eigenphase does not live in the unit interval $[0,1]$ but instead in the quotient group $\mathbb{R}/\mathbb{Z}$ [4]. This can cause confusion when interpreting the outputs of the program when the eigenphase is $\varphi = 0$. That is, when the operation of U on $|u\rangle$ is the trivial operation $U\,|u\rangle = |u\rangle$.

As a non-trivial example of this, consider the case of $U = H$. This gate has eigenvalues of ± 1 and its $+1$ eigenket is given by

$$|{+u}\rangle = \frac{1}{\sqrt{2}\sqrt{1+\sqrt{2}}}\left((1+\sqrt{2})\,|0\rangle + |1\rangle\right)\ . \qquad (4)$$

We prepare this state by an $R_y(\theta)$ rotation[5] on the $|0\rangle$ state and then pass it and the H gate through the Kitaev phase estimation program. Performing this test for many different N values and plotting gives Figure 3.

In the figure above, the estimations form two clear bands as the estimation sometimes ends just above and just below $\varphi = 0$. The periodicity of $\mathbb{R}/\mathbb{Z}$ forces the values falling below 0 to "wrap back around" to the top of the range of values

[3] Note that the correct quadrant must be inferred from the signs of the $\sin 2\pi\varphi$ and $\cos 2\pi\varphi$ and this must be adjusted for after the fact when using the arctangent here.

[4] Recall that N refers to the the number of measurements used compute $\tilde{P}(0)$ and $\tilde{P}'(0)$. For simplicity N measurements were taken for each circuit so each point in the dataset describes the estimation of φ from $2N$ measurements.

[5] As an aside, the required rotation angle is $\theta = 2\arccos\left(\frac{1+\sqrt{2}}{\sqrt{4+2\sqrt{2}}}\right)$.

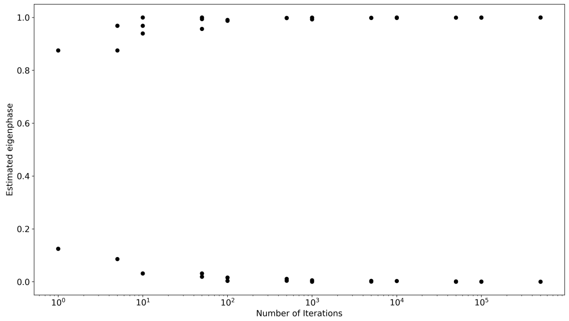


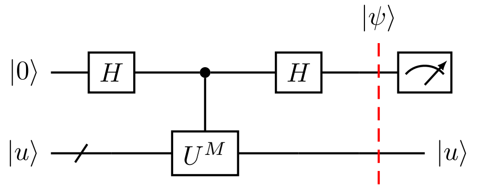
Fig. 3. Simulated results of the Kitaev phase estimation algorithm for the $+1$ eigenket of H.

and emerge near 1. This demonstrates that there is a serious possibility for confusion regarding the outputs of the Kitaev phase estimation algorithm if care is note taken to ensure that all calculations are done modulo 1 and interpreted in that sense.

VI. The best $m + 2$ bit estimation

The basic form of the Kitaev algorithm produces an estimation of φ as a real number. In many cases, such as for order finding in Shor's algorithm [1], we would expect the eigenphase to be an exact binary fraction and so we would prefer if the Kitaev algorithm could be adjusted to give a strong approximation to the closest binary fraction to the eigenphase.

An efficient method to do this is presented in [3]. Note that, the method depends on being able to estimate all integer multiples of the eigenphase $M\varphi$. In practice this is not difficult since $U^M |u\rangle = e^{2\pi i M\phi}$ and hence if we have an efficient oracle to perform the U^M operation then the circuit



will allow us to estimate $M\varphi$ in a manner identical to before. As a note, this does not require an extra assumption when compared to the QFT based phase estimation procedure since it also requires an efficient method to perform integer powers of U[6].

A discussion of how to construct a binary fraction representation is both too unwieldy to present in full here and also not really necessary. For details the Author recommends reference to [4] and [3] which cover the topic is substantial depth. In short, the "gist" of the process is as follows. To compute a precise $m + 2$ bit binary fraction estimate for φ we estimate the phase for multiples $M_j = 2^{m-1-j}$ and use each of these to successively improve our best guess for each of the bits of the answer until we end with a guess of the form $\alpha = 0.\alpha_1....\alpha_{m+2}$ which is likely the best $m + 2$ bit approximation.

VII. The structure of the Kitaev namespace

The Kitaev phase estimation process is implemented in the Kitaev namespace which is included in the *Kitaev.qs* source file[7]. This file includes several functions but most of them are back-end subroutines and debugging functions which are not intended for general use. From a user perspective there are two functions of interest.

The function *estimatePhaseKitaev* takes as input: a unitary oracle U, an eigenstate $|u\rangle$ of U, and a number of measurements N and will return the estimated eigenphase of $|u\rangle$ as a double-precision floating point value between 0 and 1.

The function *estimatePhaseKitaev_Bestmp2Bit* takes as input: a unitary oracle U, an eigenstate $|u\rangle$ of U, a number of measurements N, and a number of bits m and will return the an $m+2$ bit estimation of the eigenphase of $|u\rangle$ as a list of integers representing the bits of the binary fraction of the estimation. Note that the GitHub repository for the Kitaev namespace also includes a python file called *Host.py*. This is a python program which imports the *Kitaev.qs* to ensure that it compiles correctly and then runs a series of test simulations. This program was intended for debugging by the developer and is included for completeness.

VIII. Conclusion

Simulations of the Kitaev phase estimation algorithm found that it was capable of measuring the eigenphase of an eigenket of an oracle and therefore is capable of solving the phase estimation problem. This is implemented in the Kitaev namespace which is available for general use. When using this program care, must be taken to ensure that the periodicity of the outputs is recognised and results are not misinterpreted as a result.

There is more work to be done on this topic. Specifically, [4] presents a series of optimisations to the Kitaev algorithm which improve its speed and accuracy. This has not yet been implemented in the Kitaev namespace and may provide tangible performance benefits.

References

1. Nielsen, M. A. & Chuang, I. L. *Quantum computation and quantum information* English. ISBN: 9780521632355 (Cambridge University Press, Cambridge, 2000).
2. Carrasco, J. G. *Exploration and Implementation of Quantum Phase Estimation Algorithms* Final Project Report. 2020.
3. Kitaev, A. Y., Shen, A. & Vyalyi, M. N. *Classical and quantum computation* English. ISBN: 082182161X (American Mathematical Society, Providence, R.I, 2002).
4. Svore, K., Hastings, M. & Freedman, M. *Faster Phase Estimation* 2013. arXiv: `1304.0741 [quant-ph]`.

[6] You could perform U^M by just applying the U oracle M times. However, this adds an $O(M)$ scaling which is often unfavourable and so it is common to assume that we have access to an oracle which can perform U^M with a scaling better that $O(M)$.

[7] All source code relevant to this document is available in the GitHub repository at *https://github.com/Bohlsen/Kitaev-Phase-Estimation*.