Optimizing the Phase Estimation Algorithm Applied to the Quantum Simulation of Heisenberg-Type Hamiltonians

Scott R. Johnstun

A senior thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Bachelor of Science

Jean-François S. Van Huele, Advisor

Department of Physics and Astronomy

Brigham Young University

ABSTRACT

Optimizing the Phase Estimation Algorithm
Applied to the Quantum Simulation
of Heisenberg-Type Hamiltonians

Scott R. Johnstun
Department of Physics and Astronomy, BYU
Bachelor of Science

The phase estimation algorithm is a powerful quantum algorithm with applications in cryptography, number theory, and simulation of quantum systems. We use this algorithm to simulate the time evolution of system of two spin-1/2 particles evolving under a Heisenberg Hamiltonian. The simulation is performed on both classical simulations of quantum computers and real quantum computers. We also introduce three optimizations to the algorithm: circular, iterative, and Bayesian. We apply these optimizations to our simulations and investigate how the performance improves. We find that the circular and Bayesian optimizations exhibit the best performance in noiseless simulations and the iterative and Bayesian optimizations exhibit the best performance on quantum computers which are subject to noise. We also discuss the paradigms of iterative and update-based algorithms, which are attributes of these optimizations that can improve quantum algorithms generally.

Keywords: quantum computing, quantum simulation, phase estimation, quantum algorithms, noise, optimization, noisy systems

ACKNOWLEDGMENTS

We acknowledge support from the College of Physical and Mathematical Sciences at Brigham Young University. We also thank the members of the Quantum Information and Dynamics group at BYU for many helpful and insightful discussions. We also express gratitude to Dr. Debi Galley and various peers for their feedback and ideas for improvement while this document was being written.

Contents

Table of Contents vii

List of Figures ix

List of Tables ix

1 Background 1
- 1.1 Quantum Information . . . 1
- 1.2 Quantum Simulation . . . 2
- 1.3 Quantum Phase Estimation Algorithm (PEA) . . . 4
- 1.4 Heisenberg Hamiltonian . . . 6
- 1.5 Prior Work in the Quantum Group at BYU . . . 8
- 1.6 Overview . . . 8

2 Methods 9
- 2.1 IBM Q System . . . 9
- 2.2 Implementation of the PEA . . . 10
  - 2.2.1 Base Algorithm . . . 10
  - 2.2.2 Circular Optimization . . . 12
  - 2.2.3 Iterative Optimization . . . 13
  - 2.2.4 Bayesian Optimization . . . 14
- 2.3 Experimental Protocol . . . 17

3 Discussion 19
- 3.1 Results . . . 19
  - 3.1.1 Simulation Results . . . 20
  - 3.1.2 Quantum Experiment Results . . . 22
- 3.2 Conclusions . . . 23
- 3.3 Future Research . . . 23

Bibliography 25

Index 29

List of Figures

1.1 Quantum circuit diagram for time evolution operator of Heisenberg Hamiltonian . . 7

2.1 Quantum circuit diagram for the base phase estimation algorithm 11

2.2 Quantum circuit diagram for the iterative optimization to the phase estimation algorithm 13

2.3 Quantum circuit diagram for the Bayesian optimization to the phase estimation algorithm 15

3.1 Presentation of results from all four algorithms with expected, simulated, and experimental phase estimates 20

List of Tables

3.1 Comparison of calculated energy values from simulations and experiments with error and correlation calculations 21

Chapter 1

Background

This work is a contribution to the growing field of quantum technology. Quantum computers are a particularly significant advancement in this field due to the prospect of leveraging quantum resources to solve real-world problems. These computers come with their own share of problems, however; due to a combination of increased sensitivity to noise and more channels for noise to enter, quantum computers are subject to errors that are substantially more detrimental to computation than classical computers are. After introducing quantum information and simulation, we connect these concepts to the quantum phase estimation algorithm. We implement this algorithm and use it to simulate quantum systems on a real quantum computer. We also present three optimizations to the algorithm and apply them to our simulations. These optimizations improve performance in both simulations and on real quantum computers.

1.1 Quantum Information

Since the turn of the 21st century, quantum information has become nothing short of a revolution in physics. Quantum information exists as a union between quantum mechanics, statistical physics, and information theory and has recently acquired a mainstream following with the promising prospect

of quantum computing. Richard Feynman first proposed quantum systems as a computational tool in 1985 [1, 2]. Since then, algorithms have been found that are expected to, in theory, solve difficult problems faster than any classical computer possibly could [3]. These include Peter Shor's algorithms for factoring large integers and computing discrete logs [4]; Grover's algorithm for efficient database search [5]; and various algorithms for simulation of quantum systems [6]. With these (and other) algorithms and sufficiently capable quantum computers, many difficult or unsolved problems in mathematics, physics, chemistry, and other fields could be solved efficiently.

Requisite for the implementation of such problems is the development of physical systems that can function as large, controllable quantum computers with minimal noise. Quantum computers are made up of a collection of "qubits", the quantum version of classical bits with which classical computers compute. Whereas a classical bit can only hold a value of 0 or 1, a qubit is a quantum system that can hold any normalized linear combination of two states referred to as $|0\rangle$ and $|1\rangle$. Similar to bits in a classical computer, qubits serve as both computational space and storage space for quantum computers. However, whereas modern classical computers contain billions of bits with which to compute, modern quantum computers have less to work with: at the end of 2020, the largest quantum computers to date were a 76-qubit photonic quantum computer produced by a team of researchers in China [7] and a 72-qubit superconducting quantum computer produced by Google [8]. All quantum computers are also susceptible to noise altering the state of their qubits. Noise will be described more specifically, and in the context of our work, in Chapter 2.

1.2 Quantum Simulation

We are particularly interested in the use of quantum computers for quantum simulation, which is the process of using controllable quantum systems to simulate other, harder-to-control systems. Quantum simulation has promising applications in developing new drugs, making sense of certain

chemical reactions, and creating new materials [9]. This is because classical simulation of quantum systems is inherently inefficient. The amount of memory required to represent a quantum system on a classical computer grows exponentially as the size (number of particles, degrees of freedom, etc.) increases: a spin-1/2 system of 40 particles would require 4 terabytes to represent classically [6]. On the other hand, a quantum simulator would only require 40 qubits plus perhaps a few extra qubits, depending on the objective of the simulation. Evidently classical simulation algorithms for quantum systems are naturally limited to small systems, prompting the need for quantum simulation algorithms for future scientific discovery.

Quantum simulation can be divided into digital and analog quantum simulation. In analog quantum simulation, we have two systems, one controllable and one difficult to control, and a mapping between the two systems such that evolution of the controllable system corresponds to evolution in the difficult system. Analog quantum simulation is currently more realistic than digital quantum simulation, yet examples of systems and mappings between them are rare [10]. Digital quantum simulation, on the other hand, uses programmable universal quantum computers—quantum computers on which any unitary operation can be implemented—to simulate quantum systems. Digital quantum simulation is more versatile, allowing many types of systems to be simulated with a single physical system [11], yet currently universal quantum computers are both too small to simulate systems beyond those that can be simulated classically and too noisy to produce reliable results.

In this work we use digital quantum simulation to evolve a two-particle, spin-1/2 system described by a Heisenberg Hamiltonian and recover its first excited energy. This problem is entirely solvable on a classical computer (or with analytic methods), but our implementation of a simulation algorithm and its optimizations demonstrates some of the same challenges and questions that must be addressed in order for a large scale quantum computer to simulate the evolution of more complex systems. We assume that the reader has some basic familiarity with quantum algorithms and

notation; for general references on the subject, see [12] and [13].

1.3 Quantum Phase Estimation Algorithm (PEA)

Among the most celebrated algorithms in quantum computing is the Phase Estimation Algorithm (PEA). The purpose of the PEA is to estimate the phase ϕ conferred by a unitary operator U to an eigenstate $|u\rangle$. Mathematically, if $|u\rangle$ is an eigenstate of U, then

$$U|u\rangle = \lambda|u\rangle,$$

which indicates that λ is the eigenvalue corresponding to the eigenstate $|u\rangle$. If U is unitary, then λ must satisfy the condition $|\lambda|^2 = 1$; since λ may be complex, we can write $\lambda = e^{2\pi i\phi}$ for some $\phi \in [0,1)$. The goal of the PEA is to estimate ϕ given U. One common usage of unitary operators is for time evolution operators in quantum mechanics. We will decide on a choice for U in Section 1.4. As we will see, knowing ϕ allows us to estimate the energy of a system evolving in time, since the rate at which the phase of a stationary state oscillates is proportional to its energy.

The PEA can be represented on a quantum circuit with two quantum registers, or groups of qubits. The first register is the *counting* register, which can hold any amount of qubits; we denote the number of qubits by n. This register is the register that will actually store the estimate of the phase, so more qubits in this register allows for a more accurate estimate of the phase. The second register is known as the *state* register, and it exists solely to hold the eigenstate $|u\rangle$ from which the phase is deduced. The PEA performs its task in three steps: superposition, wherein the two registers are initialized into appropriate states; phase kickback, wherein simulation happens and phase information is recorded; and Quantum Fourier Transform (QFT) (see Chapter 5 of [12]), wherein the recorded phase information is converted to a binary representation through the inverse QFT. Note that in the following discussion, a subscript after a state indicates the number of qubits that state represents, and for an m-bit integer x, $|x\rangle_m$ represents the m-qubit state $|x_m\rangle|x_{m-1}\rangle\ldots|x_1\rangle|x_0\rangle$,

where x_i is the i^{th} bit of the binary representation of x. Tensor products, represented by $\otimes$, between the kets are implied; as an exponent to an operator U, we denote $U^{\otimes n} \equiv U \otimes U \otimes \cdots \otimes U$, where n instances of U are linked by $n-1$ tensor products. The three steps of the PEA are:

1. *Superposition*: Initialize a uniform superposition of all 2^n possible states on the n-qubit counting register:

$$|count\rangle_n = |0\rangle_n \;\longrightarrow\; H^{\otimes n}|0\rangle_n = \frac{1}{2^{n/2}}\left(|0\rangle + |1\rangle\right)^{\otimes n}.$$

   If necessary, also initialize the state register to the eigenstate $|u\rangle$ (here assumed to be known beforehand; see [14] for PEA with superposition states):

$$|state\rangle_m = |0\rangle_m \;\longrightarrow\; |state\rangle_m = |u\rangle,$$

   where m is the number of qubits required to represent $|u\rangle$; we have dropped the m subscript from $|u\rangle$ itself for brevity.

2. *Phase Kickback*: For each k from 0 to $n-1$, use the k^{th} qubit of the counting register as the control for a controlled-U^{2^k} gate acting on the state register. Each application performs the following transformation on the partial system consisting of only the k^{th} qubit of the counting register and the full state register

$$|count_k\rangle|u\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|u\rangle \;\longrightarrow\; \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\phi\cdot 2^k}|1\rangle)|u\rangle$$

   since $U|u\rangle = e^{2\pi i\phi}|u\rangle \implies U^{2^k}|u\rangle = e^{2\pi i\phi 2^k}|u\rangle$ and the controlled nature of the gate ensures U is only applied in the subspace where the k^{th} counting qubit is in the $|1\rangle$ state. Note that since $|u\rangle$ is an eigenstate of U and the phase transfers to the counting register, the state register remains unchanged. That is why this step is referred to as phase kickback. Once this has been applied to all n register qubits, the counting register is left in the state

$$|count\rangle = \frac{1}{2^{n/2}}\sum_{k=0}^{2^n-1} e^{2\pi i\phi k}|k\rangle. \tag{1.1}$$

3. *Quantum Fourier Transform*: The right hand side of Eq. (1.1) happens to be exactly the QFT of input $|2^n\phi\rangle$. Therefore, applying the inverse quantum Fourier Transform (denoted as $QFT^\dagger$, since the QFT is unitary) to the counting register will leave it in the state

$$QFT^\dagger(QFT\,|2^n\phi\rangle) = |2^n\phi\rangle\,,$$

from which phase information can be obtained. We will detail specific methods for obtaining the phase as well as variations of this algorithm in Chapter 2.

1.4 Heisenberg Hamiltonian

In the discussion of the implementation of the PEA, we did not specify U. In this work we choose to use a Heisenberg Hamiltonian, which represents a spin interaction between two particles

$$H = \frac{\lambda}{4}(\sigma_1 \cdot \sigma_2) = \frac{\lambda}{4}(X_1X_2 + Y_1Y_2 + Z_1Z_2), \tag{1.2}$$

where X_i, Y_i, and Z_i are the Pauli spin operators corresponding to particle i and $\sigma_i = X_i\hat{x} + Y_i\hat{y} + Z_i\hat{z}$. This Hamiltonian is parameterized by λ, which controls the strength of the interaction.

Evolution under this Hamiltonian is governed by its time evolution operator

$$U = \exp(-iHt/\hbar) = \exp(-iH\tau) \equiv U(\tau), \tag{1.3}$$

where we have defined the evolution parameter $\tau = t/\hbar$. Note that τ has units of inverse energy but increases exactly proportionally to time during the simulation, so it can be thought of as a parameter akin to time. Applying U to an eigenstate yields the result

$$U\,|u\rangle = \exp(-iH\tau)\,|u\rangle = \exp(-i\varepsilon\tau)\,|u\rangle\,.$$

Here ε denotes the system's energy. Estimating the phase produced by the Heisenberg Hamiltonian for various values of τ gives a function $\phi(\tau)$ which satisfies

$$2\pi i\phi(\tau) = -i\varepsilon\tau.$$

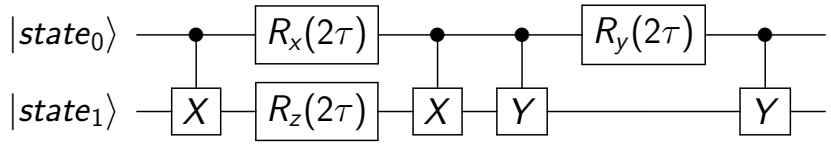


Figure 1.1 Quantum circuit diagram for the Heisenberg Hamiltonian's time evolution operator $U(\tau)$ in Eq. (1.3). The X and Y gates indicates Pauli X and Y operators respectively. The R_x, R_y, and R_z gates indicate a Pauli rotation about the corresponding axis by the input angle 2τ. To implement a controlled U gate as will be necessary in Chapter 2, only the three rotation gates need to be controlled by a qubit external to the system.

Differentiating with respect to τ and solving for ε gives us the important equation

$$\varepsilon = -2\pi \frac{d\phi}{d\tau}. \tag{1.4}$$

This is how phase estimation can be used to determine energy levels in a system.

Implementing U on a quantum computer is a nontrivial task. Although any unitary operator can be represented on a quantum computer, decomposing one into a sequence of simple unitary operations that can be implemented on a quantum computer is difficult. In Cruz *et al.* it was shown how the time evolution operator corresponding to a Hamiltonian containing Z_1Z_2 can be decomposed into *cNOT* gates and a Z rotation gate [14]. Following this result, we implement the Heisenberg Hamiltonian with the gate sequence pictured in Fig. 1.1. Gates involving X, Y, or Z refer to the corresponding Pauli operators. A filled circle on a qubit indicates that it is the control for the operation it is linked to. If T is an n-qubit gate, the controlled-T gate is the $(n+1)$-qubit gate defined by $|0\rangle\langle 0| \otimes \mathbb{1} + |1\rangle\langle 1| \otimes T$. Armed with this decomposition, we are able to use the PEA to simulate evolution under the Heisenberg Hamiltonian and determine energy levels.

1.5 Prior Work in the Quantum Group at BYU

Previous work on quantum computing in the Quantum Information and Dynamics group of BYU consists of some early work on classical simulations of quantum computers [15–17] in the 1990s and, in 2020, a proposed implementation for a Deutsch gate, which is a powerful quantum logic gate that can be used to form a universal gate set [18]. Our work does not directly build off of any of this prior work.

1.6 Overview

We build off work in References [14, 19–21] and apply their PEA optimization techniques to simulation of evolution under a Heisenberg Hamiltonian. We implement these simulations in classical quantum computer simulators and on real IBM quantum computers in Chapter 2, and in Chapter 3 we find that all optimization techniques offer some sort of improvement to the base PEA.

Chapter 2

Methods

This chapter describes our implementations of the PEA as well as the IBM Q System on which they run. We explain the concepts needed to understand the implementation of the base algorithm as well as three optimizations: circular, iterative, and Bayesian. We also describe the quantum operations needed for these algorithms and provide diagrams of the corresponding quantum circuits.

2.1 IBM Q System

The IBM Q System provides free public access to superconducting quantum computers maintained by IBM [22], which we use to run our experiments. Experiments on these computers can be designed in Qiskit [23], a Python library maintained by IBM, then sent to a quantum computer to be executed. In this way, we can run experiments on actual quantum computers to determine how well our algorithms perform in real conditions. Qiskit also provides functionality to simulate the result of a computation with classical computers among its many features. Running an experiment on an IBM quantum computer amounts to creating a circuit to be run, specifying the amount of shots (times the computer is initialized, the circuit is run, and the qubits are measured; multiple shots are necessary to reduce uncertainty due to probabilistic measurements), sending the experiment request,

and obtaining measurement results.

Noisy Intermediate Scale Quantum (NISQ) computers—including those available through IBM Q—are prone to challenges that classical computers do not have to deal with. Having far more states available to them, qubits are more sensitive to small perturbations than classical bits are. Qubits in superconducting quantum computers like those available through IBM are affected by three types of stochastic noise [24]: Johnson noise (or Johnson-Nyquist noise), which is white noise proportional to temperature; quantum noise, which is noise that results from the possibility of the qubit releasing energy to its environment, even at zero temperature; and classical 1/f noise from stray electromagnetic fields, which can cause dephasing to occur. In addition to these stochastic errors, systematic errors from imperfect hardware or incorrect pulse timing may also affect the computers [25]. Larger circuits require more time to run, making them more prone to errors than shorter circuits. In this context, circuit length is determined by the number of gates. In the next chapter we will see how circuit length affects the success of each algorithm.

2.2 Implementation of the PEA

In this section we describe our four algorithms for implementing the PEA. Throughout this section ϕ refers to the true phase and $\hat{\phi}$ refers to an estimate of the phase produced by an algorithm.

2.2.1 Base Algorithm

We first detail our base implementation of the PEA on IBM quantum computers. By base implementation we refer to the common implementation of the PEA as found in textbooks on quantum computing [26]. Most public quantum computers at IBM have five qubits, so we choose to implement a five-qubit version of the PEA. We also use three classical bits to record information onto. The resulting circuit is pictured in Fig. 2.1. Since our Hamiltonian acts on two qubits, we choose to use

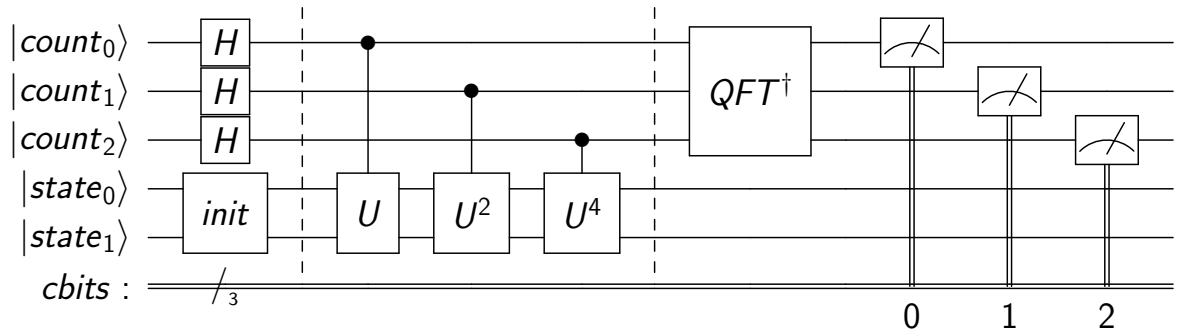


Figure 2.1 Circuit diagram for the base implementation of the PEA with a Heisenberg Hamiltonian on a 5-qubit quantum computer. The upper three qubits form the counting register and the lower two form the state register. The double line at the bottom represents the three classical bits (indicated by the slash with a three) onto which measurement information is stored. Numbers below measurements indicate the classical bit onto which the measurement result is read.

two qubits for our state register. This leaves three qubits for the counting register. Three Hadamard gates on the input register put it into the desired superposition state while the *init* gate initializes the state register to the eigenstate $(|01\rangle + |10\rangle)/\sqrt{2}$ of the Heisenberg Hamiltonian. We then apply a sequence of controlled U gates, where U is the time evolution operator corresponding to the Heisenberg Hamiltonian, acting on the state register and controlled by a single qubit on the counting register. The k^{th} qubit on the counting register controls the operation U^{2^k}. This sequence results in a counting register state of $QFT(|2^3\hat{\phi}\rangle)$, so we apply the inverse Quantum Fourier Transform $QFT^\dagger$ to recover the state $|2^3\hat{\phi}\rangle$. Finally, we measure each of the counting qubits and record the result on the corresponding classical bit. After performing many shots of this circuit, we obtain a distribution of binary integers betwen zero and seven, inclusive, which we represent as $P(x) : \{0,1,...,7\} \to \mathbb{R}$. For the base implementation of the PEA, we simply choose the result that happened with the largest frequency, $\underset{x}{\text{argmax}}\, P(x)$. Dividing this result by eight reveals an approximation of the phase:

$$\hat{\phi}_{maj} = \frac{\underset{x}{\text{argmax}}\, P(x)}{8}.$$

This technique is simple, but it is limited in resolution. In general, we can obtain one binary digit of precision per counting qubit. With only three counting qubits, the estimator $\hat{\phi}$ is limited to the eight possibilities $\{k/8\}_{k=0}^{7}$, whereas ϕ itself can be any real number in the interval $[0,1)$. This leads to discontinuous jumps in $\hat{\phi}$ as ϕ varies. Thus the base algorithm computes a rudimentary phase estimate that is limited in precision.

2.2.2 Circular Optimization

Circular statistics offer one way to optimize the PEA. This first optimization uses the same circuit as the base PEA implementation, pictured in Fig. 2.1. The optimization is in how we process the measurement data. Instead of using a simple majority rule, we calculate the circular mean of the data and use its argument as the phase estimator. Given a measurement probability distribution $P(x)$ defined for integers x, its circular mean μ is defined as

$$\mu = \sum_{x=0}^{7} P(x) e^{i\pi x/8}.$$

The circular mean is a complex number whose argument can be used as an approximator for ϕ:

$$\hat{\phi}_{circ} = \frac{1}{2\pi} \arg \mu.$$

This is a more natural estimator for ϕ because, like ϕ, $\arg\mu$ is a circular measure—the difference between $\phi = 0.01$ and $\phi = 0.99$ is much smaller than the difference between $\phi = 0.01$ and $\phi = 0.50$ due to modular arithmetic. In addition to this, utilizing the circular mean in this way allows for better resolution in $\hat{\phi}$. Although $\hat{\phi}$ cannot always exactly represent ϕ, it does vary smoothly as ϕ changes. Cruz *et al.* showed that the circular optimization used on data from a counting register of N qubits has a smaller error bound (defined as $|\hat{\phi} - \phi|$) than a majority rule estimator for a counting register of $N+1$ qubits [14].

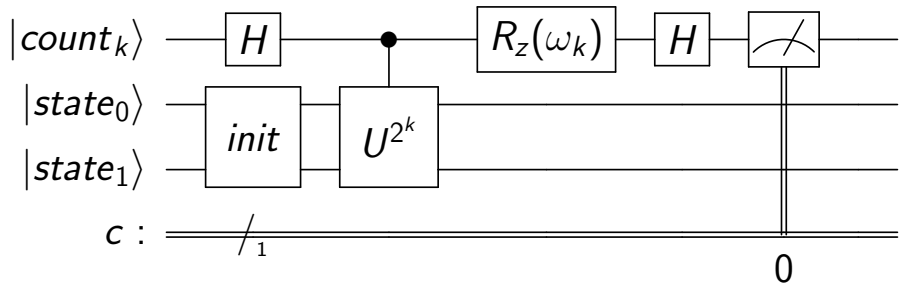


Figure 2.2 Circuit diagram for the iterative optimization to the PEA with a Heisenberg Hamiltonian. The upper qubit forms the counting register and the lower two form the state register. The bottom-most line represents a classical bit onto which measurement information is stored. The R_z gate performs one step of the semiclassical QFT when parameterized by ω_k as defined in Eq. (2.1). Note that for the iterative algorithm, three versions of this circuit must be run, each with different values for k, which affects both the U^{2^k} gate and the R_z gate.

2.2.3 Iterative Optimization

The iterative PEA is another optimization that we tested. Unlike the circular optimization, the iterative optimization uses a different circuit than the base algorithm. It is so named because instead of performing the whole phase estimation in one circuit, it uses multiple smaller circuits to build up an estimation. In our implementation, we used three 3-qubit circuits. These circuits are identical except for the power of the controlled U and the angle of the R_z gate. A diagram for the circuits is pictured in Fig. 2.2. Just as each counting qubit in the base circuit provides one binary digit of precision, each individual circuit in the iterative PEA provides one binary digit of precision. Instead of applying the controlled U^{2^l} gate in parallel on one circuit, we create each PEA circuit with one counting qubit and apply a controlled U^{2^l} gate, where l is determined by the order in which the circuits are run. In place of the QFT, we perform a semiclassical QFT [27], which separates the typical QFT algorithm into an iterative sequence of independent measurements and rotations. The iterative algorithm constructs $\hat{\phi}$ from the most ($(n-1)^{\text{st}}$) to least significant (0^{th}) bit with the following procedure:

1. Run the $k = n-1$ circuit with $\omega_{n-1} = 0$ to find the largest bit of $\hat{\phi}$.

2. Repeat while $k > 0$:

   (a) Decrement k.

   (b) Calculate the rotation angle ω_k for circuit k via the formula

   $$\omega_k = -2\pi \sum_{j=0}^{n-k-2} \frac{b_{j+k+1}}{2^{j+2}}, \tag{2.1}$$

   where b_x refers to the measurement outcome of the x^{th} circuit (the x^{th} bit of $\hat{\phi}$).

   (c) Run circuit k with the newly calculated ω_k to find b_k.

3. Recover the estimator $\hat{\phi}$ from its binary expansion:

   $$\hat{\phi}_{iter} = \frac{1}{2^3} \sum_{x=0}^{n-1} 2^x b_x.$$

In this way, we are able to perform each counting qubit of the PEA separately. To provide a fair comparison with the base algorithm, we used $n = 3$ circuits. This optimization trades a larger multiplicity of circuits to run for smaller circuit sizes. It also has the same resolution as the base algorithm, permitting only values of $\hat{\phi}$ that are multiples of $1/2^n = 1/8$. Note that circular statistics cannot be used to to improve the iterative optimization because each bit is measured independently, whereas the base algorithm permitted us to determine coincidences among qubits.

2.2.4 Bayesian Optimization

Our final optimization for the PEA utilizes Bayesian statistics. In the versions of the algorithm we have already discussed, the estimator of the phase we are trying to determine is stored in the quantum state of a few qubits in some way. With the Bayesian PEA, we instead store the estimator as a parameter in the quantum circuit and use the states of the qubits to inform us how close our

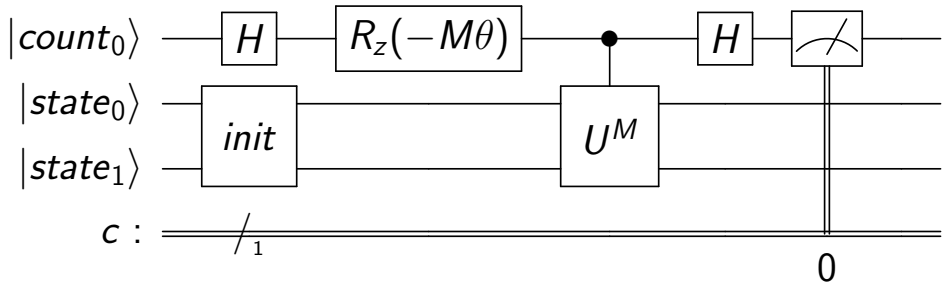


Figure 2.3 Circuit diagram for the Bayesian PEA with a Heisenberg Hamiltonian. The upper qubit is the counting register and the lower two form the state register. The bottom-most lines represent a classical bit onto which measurement information is stored. The circuit is parameterized by θ and M, which approximately represent our estimate for the phase and the certainty of our estimate, respectively. In the Bayesian algorithm, several versions of this circuit are run as the parameters M and θ are updated.

estimator is. The circuit is pictured in Fig. 2.3. It is reminiscent of the circuits used for the iterative optimization, but the role of binary digit precision is replaced by a parameter M, which is repeatedly updated in the algorithm. It also includes an extra R_z gate like the iterative algorithm; however, this R_z gate serves a different purpose. In the Bayesian algorithm, this R_z gate is also parameterized by θ, which is our current guess for the true phase. When θ is close to the true phase ϕ, the counting qubit ends up being almost purely in the $|0\rangle$ state; exactly how close depends on the parameter M. When θ is far from ϕ and M is small, the counting qubit is almost purely in the $|1\rangle$ state. This property is used to guide our Bayesian updates.

Bayesian statistics is focused on the idea of updating a guess with newly acquired information. We begin with an initial guess of the mean and variance of the phase distribution, together known as the *prior*. We choose a wide prior distribution to account for our total uncertainty about the phase. Specifically, we start with a normal distribution with a mean $\mu = \pi$ and variance $\sigma^2 = \pi^2$. Once an initial prior distribution is established, we perform the following algorithm, adapted from [19–21]:

1. Repeat N times:

   (a) Sample θ from the current prior distribution and assign $M = \lceil 1.25/\sigma \rceil$.

(b) Run the quantum circuit in Fig. 2.3 with the parameters M and θ. Record the measurement result as $E \in \{0,1\}$.

(c) Sample 100 x values from the prior distribution. For each x and based on the measurement result E, compute either

$$P(E=0|x;\theta,M) = \frac{1}{2} + \frac{\cos(M(\theta - x))}{2} \tag{2.2}$$

or

$$P(E=1|x;\theta,M) = \frac{1}{2} - \frac{\cos(M(\theta - x))}{2}, \tag{2.3}$$

which tell us the likelihood of measuring 0 or 1 if the true phase ϕ was equal to x.

(d) Generate 100 random values $u \in [0,1)$ from a uniform distribution.

(e) For each x_i, if the result of Eq. (2.2) or (2.3) is greater than u_i, add x_i to a new data set.

(f) Calculate the mean and variance of the new data set. Use the new mean and variance as the parameters of the updated prior distribution.

2. Use the final distribution's mean μ_N as the approximator for ϕ:

$$\hat{\phi}_{Bayes} = \mu_N.$$

The purpose of the Bayesian PEA is to perform successive updates on this prior in the hopes that it will become a sharp distribution around the true phase ϕ, at which point sampling $\hat{\phi}$ from the distribution yields a good estimate. Because the phase estimate is stored in a classical parameter θ instead of the quantum state of a few qubits, we obtain a significantly better resolution for this algorithm than for the other optimizations, limited only by the numerical precision available to classical computers. The trade-off is the fact that the classical portion of the algorithm is probabilistic as well as the quantum portion, so our estimate may be incorrect even when no errors happen in the quantum calculation.

2.3 Experimental Protocol

For each algorithm, we replace the U gate with the time evolution operator for the Heisenberg Hamiltonian as discussed in Section 1.3.1. After constructing the relevant circuits in Qiskit, we run them with 8192 shots (the maximum allowed by IBM) to simulate evolution under the Hamiltonian for a time parameterized by the evolution parameter τ. We choose τ to range from 0 to 2π, representing one full period of time evolution. At each τ, the phase is estimated, allowing us to build up the phase as a function of τ. This evolution is performed on both quantum simulators and real quantum computers provided by IBM Q. We then calculate the energy of the system with Eq. (1.4), using a finite difference method and taking the average derivative across the full evolution as our estimate.

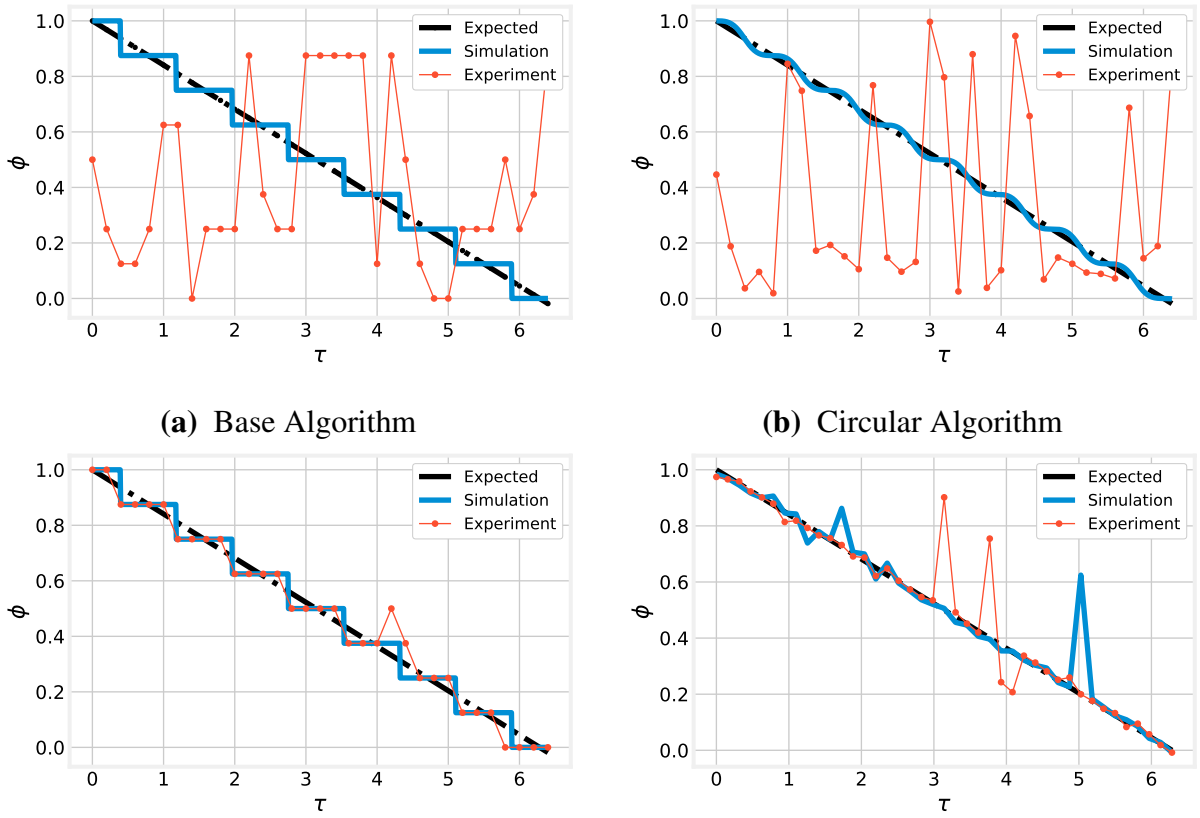
Chapter 3

Discussion

As we completed the the simulations and experiments, we looked at each algorithm's accuracy of phase estimation at each time step and overall effectiveness at estimating the energy in the system—which may not be correlated. In the following sections, we explain how we found that the algorithms perform very differently in quantum simulations and experiments on real quantum computers. In particular, the circular and Bayesian methods performed best in simulations, whereas the iterative and Bayesian optimizations performed best in quantum experiments.

3.1 Results

In this section we compare the results of simulations of quantum computers to the results of experiments on actual quantum computers. The former is useful because it provides an objective measure of the algorithm's individual performance, regardless of its implementation. The latter is important as it represents the current computational power available with modern quantum computers.



(a) Base Algorithm

(b) Circular Algorithm

(c) Iterative Algorithm

(d) Bayesian Algorithm

Figure 3.1 Comparison of simulation and experimental data from each algorithm using a Heisenberg Hamiltonian with the parameter $\lambda = 4$, which gives an energy of $\varepsilon = 1$. All three lines plot the phase ϕ as a function of the evolution parameter τ. Plotted in blue is the result of the evolution when the PEA is run on a simulation of a quantum computer and in orange is the result of the evolution when the PEA is run on an actual quantum computer at IBM. The actual phase of the Hamiltonian at each simulated time is plotted in dotted black lines.

3.1.1 Simulation Results

In simulations of noiseless quantum computers, we find that the circular optimization is ideal. A comparison of the base algorithm as well as the three optimizations is presented in Fig. 3.1, which plots calculated phase ϕ against evolution parameter τ for all four algorithms. In Figs. 3.1a and 3.1c it is apparent that the simulation results of the base and iterative algorithms are identical. The

Algorithm	Simulation			Experiment		
	ε	% Error	R^2	ε	% Error	R^2
Base	1.013	1.3%	0.986	0.887	11.3%	-1.245
Circular	1.009	0.9%	0.997	0.419	58.1%	-1.140
Iterative	1.013	1.4%	0.986	1.013	1.4%	0.975
Bayesian	0.992	0.8%	0.941	0.990	1.0%	0.913

Table 3.1 Comparison of calculated energy values from simulations and experiments with each type of PEA. The energy ε is calculated from Eq. (2.4) based on the evolution results, with an expected value of $\varepsilon = \lambda/4 = 1$ (unitless). R^2 represents the coefficient of determination, calculated using the true phase function (plotted in black dotted lines in Fig. 3.1) instead of a line of best fit.

flat plateaus followed by sharp jumps indicate the limited resolution of these two algorithms. By comparison, the continuous variation capability of the circular and Bayesian algorithms is visible in Figs. 3.1b and 3.1d. The probabilistic nature of the Bayesian algorithm is apparent in the spikes away from the trend. All four algorithms' phase estimates follow the black dotted line in the figure, which indicates the true phase value. On the left side of Table 3.1 we present the energy estimate ε, the percent error from the true energy value, and the coefficient of determination R^2, which is calculated from the true phase plotted in black instead of a line of best fit. This R^2 value gives us quantitative measure of the reliability of our energy estimate. We can see that all four algorithms are good at estimating the energy of the system, boasting both a small percentage error and a high R^2 coefficient. Of particular note is the Bayesian algorithm, which obtains the closest energy estimate to the true value but has the worst R^2 value in simulations.

While the improved resolution of the circular and Bayesian algorithms makes the actual phase estimation more accurate, it does not significantly affect the average derivative in Eqn. (1.4) from which the energy is calculated. This indicates that if an accurate result for one particular phase is desired, then the circular and Bayesian algorithms are ideal—but for energy estimation in simulation

of time evolution, all four algorithms perform acceptably well in simulations.

3.1.2 Quantum Experiment Results

Looking at the orange lines in Fig. 3.1, it is apparent that experiments on real quantum computers behave very differently from simulations of quantum computers. In simulations, there is no noise present; however on real quantum computers, noise enters as explained in Section 2.1. The base and circular algorithms give phase estimates that vary wildly, almost never giving the true phase value. In fact, the noise seems to be sufficient to render the output of these algorithms essentially random. However, the iterative and Bayesian algorithm do not demonstrate the same result. In particular, we can see that the iterative algorithm's phase estimate in experiment only deviates from that of the simulation at three time steps, and the Bayesian algorithm's experimental phase estimate varies from the true value about as much as the simulation estimate does. In the right half of Table 3.1, we present parameters to quantify the results of the experiments. The base and circular algorithms demonstrate much worse performance, both having over 10% error in the energy estimate. Even worse, their R^2 values are *negative*. This is not a numerical error; instead, it indicates that the results of the experiment better approximate a horizontal line at the mean of the phase data than the expected phase value plotted in black in Fig. 3.1. On the other hand, we also see that the iterative and Bayesian algorithms maintain a very small error in the energy calculation as well as a fairly high R^2 value, indicating that they perform very well in experiments on quantum computers. We can deduce that these latter two optimizations are essential to get reliable results from the PEA if it runs on a real quantum computer, regardless of whether phase estimates at particular times or average energy estimates are desired.

3.2 Conclusions

It is clear from the results that the base algorithm is not sufficiently robust (with respect to noise) to provide acceptable performance on real quantum computers. This is due to a combination of the significant amount of noise that they are subject and the length and size of the circuit—whereas the Bayesian and iterative algorithms use three-qubit circuits with no more than ten gates, the base and circular algorithms use a five-qubit circuit with over 50 gates. Both circuit size (qubit count) and circuit length (gate count) contribute to error buildup in quantum computers, though determining the relative contributions is outside of the scope of this paper.

We conclude that algorithms which recycle or otherwise reduce qubit count are likely to exhibit better performance than those which use more qubits. The iterative algorithm recycles a single qubit that is used with the functionality of multiple counting qubits. Although this was not possible in our implementation with the IBM Q system, it is possible for the state register to be re-used while the counting qubit is recycled [28]. This produces the effect of reduced circuit size while maintaining identical functionality. Obtaining this effect in a quantum algorithm is typically difficult as larger circuit sizes generally provide a larger algorithmic space to work in, increasing the range of problems it can solve. The Bayesian algorithm confronts this challenge by effectively becoming a "phase checker" instead of a "phase estimator," as it dynamically updates itself to produce an estimate that converges to the true phase. Circuit space is reduced by having the estimate stored as a real number in the classical system that controls the quantum computer instead of storing it on the computer itself.

3.3 Future Research

In the future, the PEA needs to perform its task accurately on large quantum systems in order to make quantum computing a useful computational tool in the hands of scientists. Our Hamiltonian

was a simple two-qubit, four-dimensional Hamiltonian whose eigenvalues are simple to calculate analytically, yet it is one of the larger Hamiltonians that is currently feasible to simulate without noise taking over. For quantum simulation to lead to novel scientific results, we need to be able to implement it on systems with Hamiltonians that are too large or complex to handle analytically or numerically. Examples include many-body problems, spin glasses, and open quantum systems [6]. Implicit in this problem is also the need to reduce a given Hamiltonian into a sequence of gates that can be implemented efficiently on a quantum computer as we did with the Heisenberg Hamiltonian in Chapter 1. Using the PEA perform previously difficult tasks in cryptography or number theory, such as breaking RSA, requires it to be run on thousands to millions qubits. Considering we found that a quantum circuit of five qubits was too large to get reliable results from a real quantum computer, it is clear that the field has a lot of room to grow and develop.

While noise reduction is a common thread that underlines the development of quantum computers everywhere, it is also important to consider other paths to optimal quantum computer performance, such as qubit connectivity and error correction. Qubits on quantum computers are typically not fully connected, which complicates qubit allocation in algorithms and can result in the addition of unwanted yet necessary gates to quantum circuits [29]. Quantum error correction is another active field of research that aims to produce robust quantum computers that are subject to noise yet remain able to produce accurate results by recognizing when an error has occurred [30]. With these and other advancements in the field, we hope to utilize quantum technology to solve problems that have never before been solved.

Bibliography

[1] R. P. Feynman, "Quantum Mechanical Computers," Optics News pp. 11–20 (1985).

[2] A. Hey, *Feynman And Computation* (CRC Press, 2018).

[3] A. Montanaro, "Quantum algorithms: an overview," npj Quantum Information **2,** 15023 (2016).

[4] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," SIAM Review **41,** 303–332 (1999).

[5] L. K. Grover, "A fast quantum mechanical algorithm for database search," (1996).

[6] I. M. Georgescu, S. Ashhab, and F. Nori, "Quantum simulation," Rev. Mod. Phys. **86,** 153–185 (2014).

[7] P. Ball, "Physicists in China challenge Google's 'quantum advantage'," Nature, https://www.nature.com/articles/d41586-020-03434-7 (Accessed February 16, 2021).

[8] J. Kelly, "A Preview of Bristlecone, Google's New Quantum Processor," Google AI Blog, https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html (Accessed February 16, 2021).

[9] G. Popkin, "Waiting for the Quantum Simulation Revolution," American Physical Society, https://physics.aps.org/articles/v12/112 (Accessed February 16, 2021).

[10] J. Argüello-Luengo, A. González-Tudela, T. Shi, P. Zoller, and J. I. Cirac, "Analogue quantum chemistry simulation," Nature **574,** 215–218 (2019).

[11] B. P. Lanyon *et al.*, "Universal Digital Quantum Simulation with Trapped Ions," Science **334,** 57–61 (2011).

[12] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. (Cambridge University Press, USA, 2011).

[13] N. D. Mermin, *Quantum Computer Science: An Introduction* (Cambridge University Press, USA, 2007).

[14] P. M. Q. Cruz, G. Catarina, R. Gautier, and J. Fernández-Rossier, "Optimizing quantum phase estimation for the simulation of Hamiltonian eigenstates," Quantum Science and Technology **5,** 044005 (2020).

[15] B. E. Kraczek, "Study of Quantum Computation Through Simulation of Shor's Algorithm on a Classical Computer," Honors Thesis (Brigham Young University, Provo, UT, 1999).

[16] Z. A. Johnson, "Simulation of a Quantum Computer on a Classical Computer," Honors Thesis (Brigham Young University, Provo, UT, 1997).

[17] D. P. Menscher, "Modeling the Quantum Computer on the Classical Computer," Honors Thesis (Brigham Young University, Provo, UT, 1997).

[18] P. W. Bailey, "Universal Quantum Circuitry: Deutsch Gate Construction Using GaAs/InAs Quantum Dots," Bachelor of Science Senior Thesis (Brigham Young University, Provo, UT, 2020).

[19] N. Wiebe and C. Granade, "Efficient Bayesian Phase Estimation," Phys. Rev. Lett. **117,** 010503 (2016).

[20] K. M. Svore, M. B. Hastings, and M. Freedman, "Faster Phase Estimation,", 2013.

[21] S. Paesani, A. A. Gentile, R. Santagati, J. Wang, N. Wiebe, D. P. Tew, J. L. O'Brien, and M. G. Thompson, "Experimental Bayesian Quantum Phase Estimation on a Silicon Photonic Chip," Phys. Rev. Lett. **118,** 100503 (2017).

[22] "Quantum Computing | IBM," https://www.ibm.com/quantum-computing/ (Accessed January 28, 2021).

[23] H. Abraham *et al.*, "Qiskit: An Open-source Framework for Quantum Computing,", 2019.

[24] W. Oliver, "Quantum Engineering of Superconducting Qubits | Seminar Series with Will Oliver," https://youtu.be/aGAb-GbrvMU?t=983 (Accessed January 28, 2021).

[25] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, "A quantum engineer's guide to superconducting qubits," Applied Physics Reviews **6,** 021318 (2019).

[26] A. Asfaw *et al.*, "Learn Quantum Computation Using Qiskit,", 2020.

[27] R. B. Griffiths and C.-S. Niu, "Semiclassical Fourier Transform for Quantum Computation," Physical Review Letters **76,** 3228–3231 (1996).

[28] E. Martín-López, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O'Brien, "Experimental realization of Shor's quantum factoring algorithm using qubit recycling," Nature Photonics **6,** 773–776 (2012).

[29] "Transpiler Passes," https://qiskit.org/documentation/tutorials/circuits_advanced/04_transpiler_passes_and_passmanager.html (Accessed February 11, 2021).

[30] J. Roffe, "Quantum error correction: an introductory guide," Contemporary Physics **60,** 226–245 (2019).

Index

Bayesian statistics, 14

Circular statistics, 12
Controlled gate, 7

Error, 10, 23
Evolution parameter τ, 6, 17

Heisenberg Hamiltonian, 6, 7, 23

IBM Q, 9, 10

NISQ, 10
Noise, 10, 24

Pauli matrices, 6
Phase estimation algorithm, 4, 10
Phase kickback, 5

Qiskit, 9, 17
Quantum circuit, 4, 7
Quantum Fourier transform, 4, 11
    semiclassical, 13
Quantum simulation, 2, 17, 24
    digital vs. analog, 3
    of Heisenberg Hamiltonian, 3
Qubit, 2, 4, 10, 24
    recycling, 23
    superconducting, 10