Quantum Implementation for Solving Linear Systems of Equations

John Harakas

Fall 2017

Abstract

Quantum computing is an emerging field of computer science that holds tremendous potential across different disciplines. Although in its infancy, the field is rapidly expanding with new advances in quantum circuit architecture. In this work, a quantum algorithm for solving 2×2 linear systems of equations is implemented using an iterative phase estimation technique and demonstrated using IBM's Quantum Experience platform. The implementation is one of the first general methods of the original algorithm.

1 Introduction

Linear systems of equations are relevant to virtually all fields of engineering, science, physics and mathematics. Finding solutions to linear systems of equations still remains a costly procedure; particularly as systems are getting larger and larger. In fields such as machine learning and high dimensional data analysis, linear systems can become exceedingly large. Quantum algorithms have been devised to solve linear systems of equations with a significant reduction in runtime over classical algorithms. The deployment of IBM's Quantum Experience (IQX) cloud platform has made quantum computing accessible to the public, making the development of such algorithms important to creating a foundation of knowledge. Currently IBM has three quantum backends available to the public: ibmqx2 and ibmqx4 having 5 quantum registers, and the ibmqx5 with 16 [1].

A linear system of equations can be represented as $A\vec{x} = \vec{b}$, where A is the matrix containing coefficients of some unknown vector $\vec{x}$. Given $\vec{b}$, we want to find a $\vec{x} = A^{-1}\vec{b}$. Notationally, vectors are represented in Dirac's "bra"-"ket" form: "ket" $|u\rangle$ and its adjoint "bra" $\langle u|$. Operators, that act as logic gates, can be expressed in terms of the outer product between two vectors: $|\psi\rangle\langle\psi|$. Quantum computations are carried out by a primitive set of unitary operators that unlike classical logic gates, are reversible; an important notion quantum algorithms. The X gate is the classical NOT gate (or bit-flip). The Z gate flips the phase of the operand and the Y gate flips both the bit and the phase. That is, $Y = XZ$. Putting a state in quantum superposition is achieved by using the Hadamard gate H. More importantly, H can be applied to multiple qubits in the sense that a uniform superposition of n states can be reached by applying $H^{\otimes n}$. Another fundamental gate is the controlled-NOT,($\bigwedge X$) a reversible negation acting on two qubits for which can entangle and disentangle multiple qubits. Controlled gates conditionally act on a target state if the control bit is set to one. In general, any quantum gate G can serve to act as a controlled gate and is denoted $\bigwedge G$ in this text. Greater detail on operators and controlled gates is given in Appendix A.2.

The general algorithm for solving linear systems of equations was first outlined by Harrow-Hassidm-Lloyd (HHL) [2] which gives an exponential speedup over classical algorithms given certain assumptions about the system. For an $s-$sparse, $N \times N$ system of linear equations, the runtime is

$O(\log(N)s^2\kappa^2/\varepsilon)$ where $\kappa = \frac{\max \lambda_i}{\min \lambda_j}$ is the condition number of the matrix and ε is the desired precision. Under similar conditions, the conjugate gradient method has a runtime of $O(Ns\kappa \log(1/\varepsilon)$. In the case where $s << N$ and k is relatively small, the speedup is exponential compared to classical algorithms [2]. Although the HHL algorithm has similar dependence on the condition number, the fact that its performance scales $\log(N)$ is particularly important to fields such as data science where N can be very large.

Given a Hermitian matrix A and input vector $\vec{b}$, the algorithm computes a state proportional to $\vec{x} = A^{-1}\vec{b}$. The runtime performance rests on the assumption that A, $\vec{b}$ and $\vec{x}$ be efficiently computable. In other words, initializing $\vec{b}$ in a quantum state $|b\rangle$ requires only a finite sequence of gate operations that can represent $|b\rangle$ to some arbitrary precision. Generally, the algorithm consists of three steps: phase estimation, eigenvalue inversion, and reverse phase estimation. The general case circuit is outlined in Figure 1. Phase estimation is a well known technique to estimate the eigenvalues of some Hermitian operator. With respect to this work, representing a matrix as a two-qubit time evolution operator acting on the input state will decompose the input in the eigenbasis of the matrix while the control qubit will pick up a relative phase shift at each step.

Once the eigenvalues are are computed, inverting them can be done using a rotation in the circular (Y) basis. The angle of rotation is determined by the measured eigenvalues. For a diagonalized matrix, the eigenvalues are directly on the trace and inverting the diagonal inverts the matrix. The last step is to uncompute the phase estimation steps and measuring the ancilla register to determine a successful outcome.

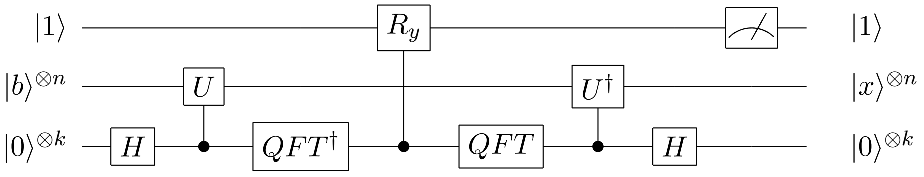


Figure 1: General circuit for $N \times N$ system where $N = 2^n$. Phase estimation is done by applying U to the input register, conditioned on the bottom register and performing the inverse Quantum Fourier Transform (QFT). The state of the bottom register contains the eigenvalues of U. Inverting the eigenvalues is done by through a rotation about the y-axis. A measurement of 1 in the top register indicates the algorithm was successful.

2 Methodology

If the ratio between the largest and smallest eigenvalues is close to 1, A is well-conditioned and is invertible without significant loss of precision. Without loss of generality, we will limit implementation to a 2×2 system whose eigenvalues are powers of two. As a result, the algorithm can be implemented using less than five registers. Implementation of a quantum circuit is largely dependent on the coupling between qubit registers. Two registers must be connected by a superconducting bus reasonator in order for a general controlled gate to act on them. The resonance frequency interaction between two connected qubits determines the gate direction; which qubit acts as the control [3]. Because the eigenvalue register needs to act on the input register $|b\rangle$ in addition to an ancillary register, a similar circuit implementation in [4] was ideal for the ibmqx4 backend based on the coupling map in Figure 2.

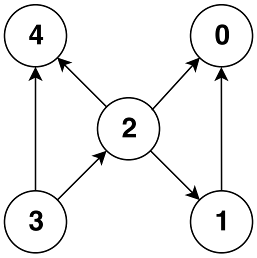


Figure 2: Coupling map for the ibmqx4 backend. The ability for a register to apply a controlled operation on another register is denoted by a directed edge. The circuit implementation uses register q_0 as the ancilla, q_1 for the eigenvalues, and q_2 for the input.

Phase Estimation

The first step involves quantum phase estimation to compute the eigenvalues of A. In general, this involves applying the Walsh-Hadamard transform, $W = H^{\otimes k}$, on a k-qubit register $|0\rangle^{\otimes k}$, resulting in a uniform superposition. Each of the k qubits applying a controlled unitary operation on $|\psi\rangle$. With a single-qubit control register, only a single Hadamard gate is needed, producing a control qubit $|+\rangle = H|0\rangle$. Typically phase estimation is done by using the Quantum Fourier Transform (QFT). Instead, Kitaev's Iterative Phase Estimation Algorithm (IPEA) is used. Although each bit requires a measurement, precision of $O(2^{-m})$ can be reached in $\log(m)$ iterations using only m ancillary qubits or in $m\log(m)$ iterations with one ancillary qubit [5].

Although IPEA is exact when eigenvalues of powers of two, trivial adjustments can be made to reach an arbitrary precision if that is not the case. Each iteration returns the least significant bit of the phase of an eigenvalue which can be classically used to determine the approximate eigenvalue.

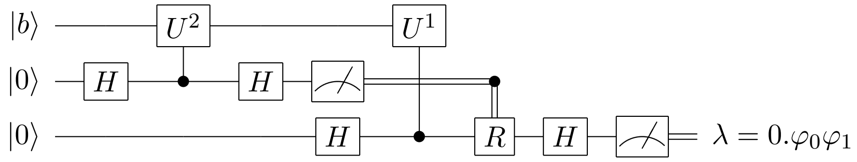


Figure 3: Circuit diagram of iterative phase estimation to precision of 2 bits.

Figure 3 outlines the circuit for a two-bit precision estimate. An operator U is applied to the input vector N times. After the kth bit is measured, the R gate rotates the phase based on the last measured bit. In outer product notation, $R(\varphi_k) = |0\rangle\langle 0| + f(\varphi_k)|1\rangle\langle 1|$ where $f(\varphi_k)$ is given by

$$f(\varphi_k) = \begin{cases} e^{\frac{-i\varphi_k\pi}{2^k}} & k > 0 \\ e^{\frac{-i\pi}{2^{k-1}}} & k = 0. \end{cases}$$

A more detailed explanation is provided in A.3.

Inversion

Under the assumption that the eigenvalues of A are powers of two, their values can computed exactly. Measurement from phase estimation should then produce $|01\rangle$ or $|10\rangle$, for which the angle of rotation about the y-axis can be calculated such that

$$\theta_j = 2\arcsin\Big(\frac{\min\{\lambda_i\}}{\lambda_j}\Big)$$

for which a rotation in the circular basis by θ_j should produce a solution state proportional to $|x\rangle$ with high probability. If the smallest eigenvalue is known, the angle θ_j will minimize error associated with the spectral gap of an operator. Since the eigenvalues are measured from the previous step, the error should be minimized by this step.

Uncomputing

Normally, the circuit will have to reverse all computations done during phase estimation in order to uncompute any frivolous states. However since phase estimation is done iteratively, only the input vector will be prepared.

3 Results

Two different implementations tested: a single eigenvalue register and two eigenvalue registers. For both cases, the input states are completely arbitrary and any computable state within the confines of memory can be used as input. The first is being dubbed "trivial" because it is a unitary system; the matrix is its own inverse and its eigenvalues are ±1. The second case involves solving a system using one register for each eigenbasis of a nonunitary matrix.

3.1 Trivial Case

Although a trivial case, it is the most direct method of solving systems of linear equations compared to the next case. The algorithm was tested by constructing A and $|b\rangle$ as

$$A = U_3(5\pi/6, 0, \pi) = \begin{bmatrix} 0.259 & 0.966 \\ 0.966 & -0.259 \end{bmatrix} \qquad |b\rangle = U_3(1/4, 0, 0)H|0\rangle = \begin{bmatrix} 0.613 \\ 0.790 \end{bmatrix}$$

However, the matrix is unitary ($A^{-1} = A^{\dagger}$), and its eigenvalues lie on the unit circle. In this case, since the eigenvalues are ± 1, it only requires a rotation about π because $\theta = 2\arcsin(\pm 1) = \pm\pi$. Expanding the computation, it is evident that the inversion of a unitary matrix corresponds to a rotation by 2π:

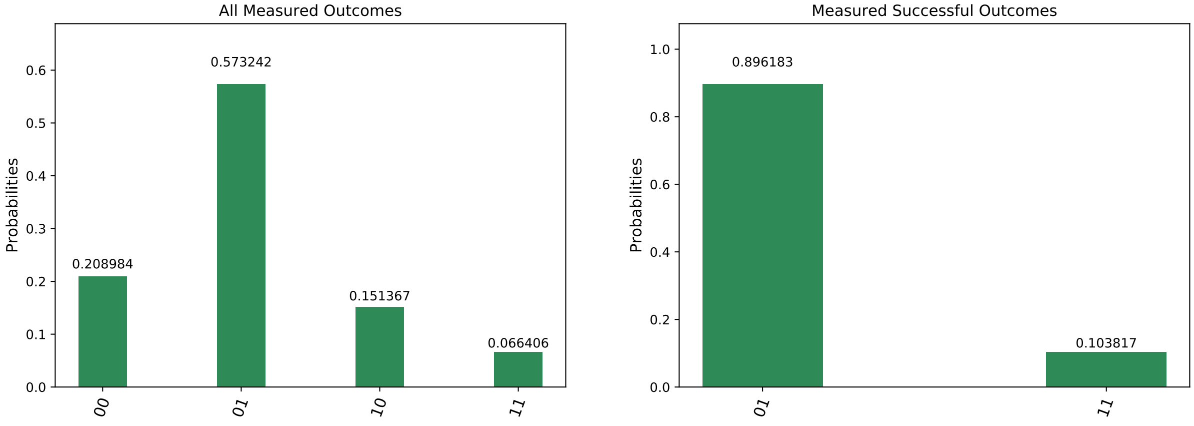
$$\begin{aligned}
&\bigwedge R_Y(\theta_1) \bigwedge R_Y(\theta_2)\Big(((A|b\rangle) \otimes |1\rangle\Big) \\
&= \bigwedge R_Y(\pi) \bigwedge R_Y(\pi)\Big(0.922|01\rangle + 0.388|11\rangle\Big) \\
&= \big(I \otimes ZX\big)\big(I \otimes XZ\big)\big(0.922|01\rangle + 0.388|11\rangle\big) \\
&= \big(I \otimes ZX\big)\big(0.922|01\rangle + 0.388|10\rangle\big) \\
&= 0.922|01\rangle + 0.388|11\rangle
\end{aligned}$$

which is consistent with the numerical results. The experimental outcome in Figure 4 is also consistent since the probability of measuring an outcome is the square of the amplitude (complex modulus). That is,

$$P(z) = |z|^2 = \mathrm{Re}(z)^2 + \mathrm{Im}(z)^2.$$

	x_0	x_1
Measured x_i	0.94667	0.32221
Numerical $\tilde{x_i}$	0.92161	0.38813
Error ε_i	0.02506	0.06592
Error $\varepsilon_i(\%)$	2.71958	16.98443

Table 1: Experimental outcomes and error of each component from Figure 4. The error ε_i is given by $|\tilde{x_i} - x_i|$ and percent error by $\frac{|\tilde{x_i}-x_i|}{\tilde{x_i}} \times 100\%$. Out of 1024 shots, the success rate was 64%



(a) All measured outcomes.

(b) Successful outcomes.

Figure 4: Measurement outcomes from Table 1 where the probability of measuring each outcome is the square of the amplitude's modulus. A successful outcome is measured when the rightmost bit is 1. Out of 1024 shots, 655 were successful - approximately 64%

3.2 Nontrivial Case

Another approach taken was given a diagonalizable matrix, apply unitary matrices U that diagonalize the matrix. For Hermitian matrices, the condition that it be diagonalizable is equivalent. The decomposition can be represented as:

$$UAU^\dagger = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix} \implies A^{-1} = U^\dagger \begin{bmatrix} \lambda_1^{-1} & & \\ & \ddots & \\ & & \lambda_N^{-1} \end{bmatrix} U$$

A simple implementation for systems of linear equations was encoded as the circuit in Figure 5 using QISKit. The matrix and its associated decomposition used is given by

$$A = \begin{bmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{bmatrix} = U\Lambda U^\dagger = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

where U can be exactly parameterized by $U_3(\pi/2, 0, 0)$. The input vector $|b\rangle$ and expected solution $|x\rangle$

$$|b\rangle = |+\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad |x\rangle = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

The circuit implementation is not significantly different than that of the first case. Since an individual register cannot store eigenvalues greater than 1, an additional register is adjoined. The left and right operators are applied to their respective registers followed by rotation. Following a successful measurement of the ancilla register, the remaining two registers are left in an eigenbasis corresponding to $A^{-1}|x\rangle$.

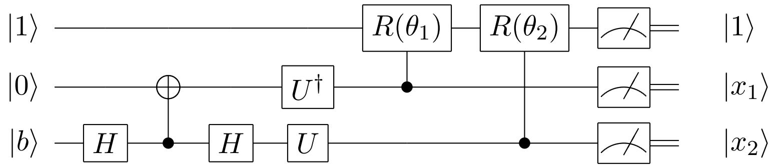


Figure 5: Circuit diagram for two registers following the phase estimation step.

In terms of how the matrix is represented, its values are normalized by some factor C since A is not unitary. However, the normalization factor C can be effectively calculated by sampling the failed outcomes. For this particular case, the successful outcomes in Figure 6b indicate an even superposition corresponding to the Hadamard basis state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. However, the true solution has a component $1 - C^2 = 1 - (\frac{1}{\sqrt{2}})^2 = \frac{1}{2}$ giving

$$|x\rangle = \frac{1}{2\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

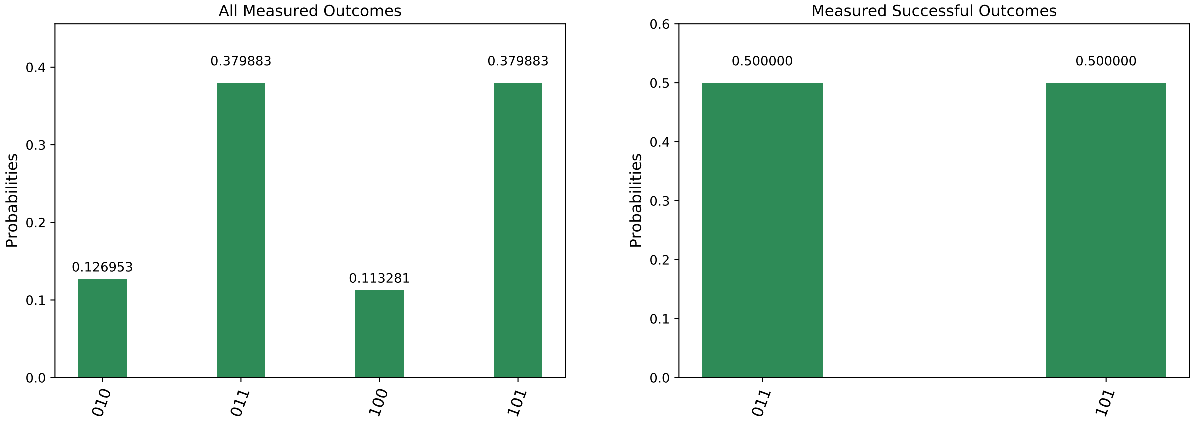
	x_0	x_1
Measured x_i	0.37367	0.33343
Numerical $\tilde{x_i}$	0.35355	0.35355
Error ε_i	0.02012	0.02012
Error $\varepsilon_i(\%)$	5.69106	5.69106

Table 2: Experimental outcomes and error of each component from Figure 6.

4 Discussion

As far as we know, iterative phase estimation has not been used in the context of solving linear systems of equations. However, it is flexible with respect to making time and memory trade-offs. It is also requires much fewer gates than QFT-based phase estimation [5]. In cases where high precision is needed, the iterative approach can do so with as few as two registers but at the expense of more runs. However, considering that there are already 20 qubit quantum computers available for commercial use, estimating the eigenvalues can conceivably be done in very few runs. For instance, given n registers, n bits of precision can be calculated in a single run. For k eigenvalues of n-bit precision, this only requires k runs since iterative phase estimation is deterministic [5]. Comparatively, QFT-based phase estimation is probabilistic and contributed heavily to the error analysis in [2]. With respect to solving linear systems of equations, this approach can minimize error attributed to mapping the inverse.

With respect to the results, the success rate and accuracy may not have been minimized. QiSKit uses OpenQASM to compile circuits for execution on their physical backends. Some of the compiler optimizations can alter circuits in unintended ways; particularly for experimental executions. The



(a) All measured outcomes.

(b) Successful outcomes.

Figure 6: Measurement outcomes. The probability of measuring each outcome is the square of the amplitude's modulus. A successful outcome is measured when the rightmost bit is 1.

methods in which the code was compiled for execution was not investigated in this work, but is important to minimize errors [6].

4.1 Related Work

Nearly all prior implementations of HHL have been done using specific apertures designed for the experiment. In addition, all published research has be done using quantum linear optics as opposed to a nuclear magnetic resonance quantum computer. An experimental realization of solving linear systems of equations was implemented in [7] requiring one input qubit for $|b\rangle$, one ancillary qubit, and two eigenvalue qubits. In this case, the system under consideration had eigenvalues being powers of two and the inversion step was ad hoc; achieved by swapping the two eigenvalue registers.

Other work was done implementing a solver with eigenvalues between 0 and 1 using only two qubit registers in the best cases [4]. The approach was analogous to that of [5], where eigenvalue bits were obtained in a single shot version of Kitaev's phase estimation algorithm. This work followed similar principles in implementing the phase estimation step.

Solving linear systems of equations was demonstrated on IQX previously for the purpose of demonstrating homomorphic encryption [8]. The circuit construction followed [7] and the operator construction relied on the IBM Composer interface. However, the inversion method was also ad-hoc. The matrix under consideration had eigenvalues that could not be effectively stored in the circuit.

Previous experimental implementations have all used ad hoc methods for solving linear systems of equations. Due to a limited number of quantum registers, experiments are conducted with a priori knowledge of the eigenvalues of a system [4], [7]–[9]. In this work, measurement from phase estimation allowed for an exact angle of rotation to be computed; maximizing the probability of success [2]. In theory, this approach could be used to get an arbitrarily close approximation of any eigenvalues with some slight modifications in the case it is not an exact power of two.

4.2 Limitations

The principle limitation in this work was that of memory. Although arbitrary precision can be achieved, register size is the limiting factor. This limits the size of linear systems that can be tested. Another important limitation in quantum algorithms in general is that outputting the results requires N runs since only one measurement can be observed at a time. However, there are cases in which the entire state needs to be measured [2].

The vector $|b\rangle$ represented by a single qubit also constrains that its coefficients b_1 and b_2 be such that $|b_1|^2 + |b_2|^2 = 1$ For a given system $A|x\rangle = |b\rangle$, the runtime performance of the algorithm outlined in [2] rested on specific assumptions about A, $|x\rangle$, and $|b\rangle$. Most importantly, both $|x\rangle$ and $|b\rangle$ must be efficiently computable.

The requirements that a matrix be Hermitian and well-conditioned were addressed in [2]; for which some superficial modifications can be made. The case where A is not square Hermitian, linear system of equations can be written equivalently as

$$\begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix} \begin{bmatrix} 0 \\ \vec{x} \end{bmatrix} = \begin{bmatrix} \vec{b} \\ 0 \end{bmatrix}$$

which implies $A\vec{x} = \vec{b}$. As pointed out in [2], ill-conditioned matrices can be potentially solved by using a preconditioner matrix B such that $\kappa(AB) << \kappa(A)$.

Although qubit registers can store an exponentially large state in quantum superposition, we are still limited to a single measurement. Naively, one might think that any performance gains are lost because it would require at least N trials to output all entries in a solution vector of length N. However, it is not always the case that the entire solution vector is needed. Instead, one might only be interested in the expectation of some operator on the output [2].

5 Conclusion

This work was a demonstration of using the IBM Quantum Experience platform in order to solve linear systems of equations using iterative phase estimation. Specifically, the implementation was carried out using QiSKit, a Python SDK. Currently, it is among the first, general implementations of solving linear systems of equations on a quantum computer. No prior work has effectively implemented the algorithm for a general use case. Although memory is the primary limitation, all the algorithm subroutines can be scaled across multiple registers. As of this work, IBM has opened access to their 16-qubit ibmqx5 backend [10]. Since the state space increases exponentially with the number of qubits, much larger systems can be represented using relatively small number of registers.

Remark

Source code repository can be found at [11]

References

[1] (). IBM q - beta access upgrade to the IBM QX, [Online]. Available: `https://quantumexperience.ng.bluemix.net/qx/community/question?questionId=db5f64ac99c6edc78e34932dfe593f36&channel=news` (visited on 09/29/2017).

[2] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for solving linear systems of equations," *Physical Review Letters*, vol. 103, no. 15, Oct. 7, 2009, ISSN: 0031-9007, 1079-7114. DOI: `10.1103/PhysRevLett.103.150502`. arXiv: `0811.3171`. [Online]. Available: `http://arxiv.org/abs/0811.3171` (visited on 11/05/2017).

[3] *Ibmqx-backend-information: Information about the different backends on the IBM q experience*, original-date: 2017-06-12T17:49:15Z, Jan. 12, 2018. [Online]. Available: `https://github.com/QISKit/ibmqx-backend-information` (visited on 01/13/2018).

[4] S. Barz, I. Kassal, M. Ringbauer, Y. O. Lipp, B. Dakic, A. Aspuru-Guzik, and P. Walther, "Solving systems of linear equations on a quantum computer," *Scientific Reports*, vol. 4, no. 1, May 2015, ISSN: 2045-2322. DOI: `10.1038/srep06115`. arXiv: `1302.1210`. [Online]. Available: `http://arxiv.org/abs/1302.1210` (visited on 12/02/2017).

[5] M. Dobsicek, G. Johansson, V. S. Shumeiko, and G. Wendin, "Arbitrary accuracy iterative phase estimation algorithm as a two qubit benchmark," *Physical Review A*, vol. 76, no. 3, Sep. 19, 2007, ISSN: 1050-2947, 1094-1622. DOI: `10.1103/PhysRevA.76.030306`. arXiv: `quant-ph/0610214`. [Online]. Available: `http://arxiv.org/abs/quant-ph/0610214` (visited on 01/17/2018).

[6] *Openqasm: Gate and operation specification for quantum circuits*, original-date: 2017-03-03T17:03:30Z, Sep. 27, 2017. [Online]. Available: `https://github.com/QISKit/openqasm` (visited on 09/29/2017).

[7] X.-D. Cai, C. Weedbrook, Z.-E. Su, M.-C. Chen, M. Gu, M.-J. Zhu, L. Li, N.-L. Liu, C.-Y. Lu, and J.-W. Pan, "Experimental quantum computing to solve systems of linear equations," *Physical Review Letters*, vol. 110, no. 23, Jun. 6, 2013, ISSN: 0031-9007, 1079-7114. DOI: `10.1103/PhysRevLett.110.230501`. arXiv: `1302.4310`. [Online]. Available: `http://arxiv.org/abs/1302.4310` (visited on 11/05/2017).

[8] H.-L. Huang, Y.-W. Zhao, T. Li, F.-G. Li, Y.-T. Du, X.-Q. Fu, S. Zhang, X. Wang, and W.-S. Bao, "Homomorphic encryption experiments on IBM's cloud quantum computing platform," *ARXIV:1612.02886 [physics]*, Dec. 8, 2016. arXiv: `1612.02886`. [Online]. Available: `http://arxiv.org/abs/1612.02886` (visited on 09/25/2017).

[9] Y. Cao, A. Daskin, S. Frankel, and S. Kais, "Quantum circuit design for solving linear systems of equations," *Molecular Physics*, vol. 110, no. 15, pp. 1675–1680, Aug. 10, 2012, ISSN: 0026-8976, 1362-3028. DOI: `10.1080/00268976.2012.668289`. arXiv: `1110.2232`. [Online]. Available: `http://arxiv.org/abs/1110.2232` (visited on 11/12/2017).

[10] (). IBM q experience, [Online]. Available: `https://quantumexperience.ng.bluemix.net/qx/devices` (visited on 12/31/2017).

[11] J. Harakas, *Quantum-qiskit: Quantum computing scratchpad and recipes for IBM q*, original-date: 2017-10-27T04:23:53Z, Dec. 7, 2017. [Online]. Available: `https://github.com/harakasj/quantum-qiskit` (visited on 02/03/2018).

A Appendix

A.1 Basis States

A qubit is in a superposition if it's components have two or more distinct states. Most common is the Hadamard, or diagonal, basis $\{|+\rangle, |-\rangle\}$ where

$$|+\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\end{bmatrix} + \frac{1}{\sqrt{2}}\begin{bmatrix}0\\1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$
$$|-\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\end{bmatrix} - \frac{1}{\sqrt{2}}\begin{bmatrix}0\\1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

A.2 Operators

General unitary operators

: Arbitrary unitary operators can be effectively implemented in IBMQX with parametric gates:

$$U_1(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix} \qquad U_2(\varphi, \theta) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -e^{(i\theta)} \\ e^{i\varphi} & e^{i(\theta+\varphi)} \end{bmatrix} \qquad U_3(\rho, \varphi, \theta) = \begin{bmatrix} \cos(\theta/2) & -e^{i\rho}\sin(\theta/2) \\ e^{i\varphi}\sin(\theta/2) & e^{i(\rho+\varphi)}\cos(\theta/2) \end{bmatrix}$$

where ρ, φ, and θ are angles corresponding to rotations in $\mathbb{C}^3$. The general controlled unitary operator is given by:

$$CU_3(\theta, \phi, \lambda) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\theta/2)e^{-i(\phi+\lambda)/2} & -\sin(\theta/2)e^{-i(\phi-\lambda)/2} \\ 0 & 0 & \sin(\theta/2)e^{i(\phi-\lambda)/2} & \cos(\theta/2)e^{i(\phi+\lambda)/2} \end{bmatrix}$$

Pauli Operators

X gate (Bit flip) :

$$X = R_x(\pi) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Z gate (Phase flip):

$$Z = R_z(\pi) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Y gate(Bit and phase flip):

$$Y = R_y(\pi) = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

$$Y = XZ$$

Parameterized Quantum Gates

$$X = U_3(\pi, 0, \pi) \qquad S = U_1(\pi/2)$$

$$Y = U_3(\pi, \pi/2, \pi/2) \qquad S^\dagger = U_1(-\pi/2)$$

$$Z = U_1(\pi) \qquad T = U_1(\pi/3)$$

$$H = U_2(0, \pi) \qquad T^\dagger = U_1(\pi/4)$$

Parameterized Rotation Gates

$$R_x(\theta) = U_3(\theta, -\pi/2, \pi/2)$$

$$R_y(\theta) = U_3(\theta, 0, 0)$$

$$R_z(\theta) = U_1(\varphi)$$

A.3 Iterative Phase Estimation

Given $|\psi\rangle$ and U, find φ such that $U|\psi\rangle = e^{i2\pi\varphi}|\psi\rangle$. Let φ has a binary expansion of the form:

$$\begin{aligned}\varphi &= \sum_{k=1}^{m} \frac{\varphi_k}{2^k} = \frac{\varphi_1}{2} + \frac{\varphi_2}{4} + \ldots + \frac{\varphi_m}{2^m} \\ &= 0.\varphi_1\varphi_2\ldots\varphi_m\end{aligned}$$

Given two qubits q_0 and q_1,

$$q_0 \mapsto |+\rangle \qquad q_1 \mapsto |\psi\rangle$$

Using $|+\rangle$ as the control and applying U to the state $|\psi\rangle$ 2^k times transforms

$$q_0 \to |0\rangle + e^{2\pi i 2^t \varphi}|1\rangle$$

Looking at each step to see the phase in q_0:

$$\begin{aligned}
k &= 0 \\
e^{2\pi i 2^0 \varphi} &= e^{2\pi i\varphi} = \exp\Big(2\pi i\Big(\frac{\varphi_1}{2} + \frac{\varphi_2}{2^2} + \ldots + \frac{\varphi_m}{2^m}\Big)\Big) \\
&= e^{2\pi i 0.\varphi_1\varphi_2\ldots\varphi_m} \\
k &= 1 \\
e^{2\pi i 2^1 \varphi} &= \exp\Big(2\pi i\Big(2\Big(\frac{\varphi_1}{2}\Big) + 2\Big(\frac{\varphi_2}{2^2} + \ldots + \frac{\varphi_m}{2^m}\Big)\Big)\Big) \\
&= \exp(2\pi i\varphi_1)\exp\Big(2\pi i\Big(\frac{\varphi_2}{2} + \ldots + \frac{\varphi_m}{2^{m-1}}\Big)\Big) \\
&= e^{2\pi i\varphi_1} e^{2\pi i 0.\varphi_2\varphi_3\ldots\varphi_m} \\
k &= 2 \\
e^{2\pi i 2^2 \varphi} &= \exp\Big(2\pi i\Big(4\Big(\frac{\varphi_1}{2}\Big) + 4\Big(\frac{\varphi_2}{2^2}\Big) + 4\Big(\frac{\varphi_3}{2^3}\ldots + \frac{\varphi_m}{2^m}\Big)\Big)\Big) \\
&= \exp(2\pi i 2\varphi_1)\exp(2\pi i\varphi_2)\exp\Big(2\pi i\Big(\frac{\varphi_3}{2} + \ldots + \frac{\varphi_m}{2^{m-2}}\Big)\Big) \\
&= e^{2\pi i 2\varphi_1} e^{2\pi i\varphi_2} e^{2\pi i 0.\varphi_3\ldots\varphi_m} \\
k &= m-1 \\
e^{2\pi i 2^{m-1} \varphi} &= \exp(2\pi i 2^{m-1}\varphi_1)\exp(2\pi i 2^{m-2}\varphi_2)\ldots\exp\Big(2\pi i\Big(\frac{\varphi_m}{2^1}\Big)\Big) \\
&= e^{2\pi i 2^{m-1}\varphi_1} e^{2\pi i 2^{m-2}\varphi_2}\ldots e^{2\pi i 2^{-1}\varphi_m} \\
&= e^{2\pi i 0.\varphi_m}
\end{aligned}$$

The state after t controlled unitary operations is $|\psi'\rangle|0\rangle + e^{2\pi i 0.\varphi_m}|1\rangle$. Measurement of $|\psi'\rangle$ in the Hadamard basis produces the following based on the value of φ_m,

$$\begin{aligned}\varphi_m = 0 &\implies |0\rangle \\ \varphi_m = 1 &\implies |1\rangle\end{aligned}$$

A.4 Multi-Qubit Inversion

Assume the eigenvalues $\{1, 2\}$ were already calculated from iterative phase estimation for

$$A = \begin{bmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$$

giving $\theta = 2 \arcsin\left(\frac{1}{2}\right)$. The circuit diagram depicted in Figure 5 proceeds as follows:

$$\begin{aligned}
|\psi_0\rangle &= \bigwedge X(|b\rangle \otimes |0\rangle) = \bigwedge X(|+\rangle \otimes |0\rangle) \\
&= \frac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big) \\
|\psi_1\rangle &= (I \otimes U)|\psi_0\rangle \\
&= \frac{1}{2}\big(|00\rangle + |01\rangle + |10\rangle + |11\rangle\big) \\
|\psi_2\rangle &= (U^\dagger \otimes I)|\psi_1\rangle \\
&= \frac{1}{\sqrt{2}}\big(|01\rangle - |10\rangle\big) \\
|\psi_3\rangle &= |\psi_2\rangle \otimes |1\rangle \\
&= \frac{1}{\sqrt{2}}\big(|011\rangle - |101\rangle\big) \\
|\psi_4\rangle &= \bigwedge R(\theta)|\psi_3\rangle \\
&= \frac{-1}{2\sqrt{2}}|010\rangle + \frac{\sqrt{\frac{3}{2}}}{2}|011\rangle + \frac{1}{\sqrt{2}}|101\rangle \\
|\psi_5\rangle &= \bigwedge R(\theta)|\psi_4\rangle \\
&= \frac{-1}{2\sqrt{2}}|010\rangle + \frac{\sqrt{\frac{3}{2}}}{2}|011\rangle - \frac{1}{2\sqrt{2}}|100\rangle + \frac{\sqrt{\frac{3}{2}}}{2}|101\rangle
\end{aligned}$$

The two eigenvalue registers are entangled using a controlled-NOT gate ($\bigwedge X$). Applying the operators U and $U^\dagger$ to $|\psi_0\rangle$ and $|\psi_1\rangle$ produces a state $|\psi_2\rangle$ in the eigenbasis of A. The state $|\psi_4\rangle$ is the result of applying the first rotation to the ancillary register conditioned on the first eigenvalue register. The final state $|\psi_5\rangle$ is the second rotation conditioned on the second eigenvalue register. If we chose to specify a measurement with respect to the Hadamard basis, the resulting state would be $|1\rangle$ 75% of the time. From this, we can deduce that the solution vector has components $|0\rangle$ and $|1\rangle$. Although the result is a normalized solution, the scaling coefficient can be calculated from the probability of failure.