HHL Algorithm for Linear Systems of Equations

Danial Imam (22120009), Amber Riaz (22120010)
Introduction to Quantum Information - PHY 612

April 2023

Abstract

The HHL algorithm, proposed by Aram Harrow, Avinatan Hassidim, and Seth Lloyd in 2009, is used for solving linear systems of equations using the principles of quantum computing. To solve such a system, we cast our problem in the form $A|x\rangle = |b\rangle$, where $|x\rangle$ and $|b\rangle$ are normalized vectors and A is a hermitian matrix. The process involves finding the eigenvalues of the matrix by making use of the Quantum Phase Estimation (QPE) sub-routine. This in turn makes use of the inverse Quantum Fourier Transform (QFT). The determined eigenvalues are then used to implement a controlled rotation to effectively find the inverse of the matrix A. This allows us to calculate $|x\rangle = A^{-1}|b\rangle$. The final step is to uncompute the phase estimation. We next discuss the step by step implementation of this algorithm on physical hardware and simulate the results on IBM's quantum computers. Finally, we compare the operation counts of the classical algorithms with the HHL algorithm which promises an exponential boost to computation speed.

Keywords: HHL, quantum computing, linear systems, quantum phase estimation.

Contents

1 Introduction

Linear systems of equations are widespread in many sub-fields of physics, engineering, mathematics, finance, and computer science. Classically, the best algorithms for solving an $N \times N$ system, such as Gaussian elimination which incorporates pivoting, take polynomial time, i.e., $\mathcal{O}(N^3)$ [4]. But with advances in these fields, we encounter ever larger systems of equations where even polynomial time is too costly.This is where the strength of quantum computing shines, which can offer exponential speed ups to computation time. The HHL algorithm is one such implementation of quantum computing principles. Introduced in 2009 by Harrow, Hassidim and Lloyd, it can be used to solve the matrix equation $A|x\rangle = |b\rangle$, where A is a hermitian matrix. This quantum advantage, however, comes at the cost of detail in the output information. To wit, the algorithm does not provide the complete solution vector. Instead, it generates a function of the solution vector, which would still enable the user to extract useful information about the solution vector. This can be thought of as finding the expectation value of an operator M acting on the solution vector as $\langle x|M|x\rangle$ and measuring the outcome [5, 3].

This algorithm serves as a subroutine of more advanced algorithms such as those pertaining to Machine Learning and the modeling of quantum systems [4]. Indeed, with advances in quantum computing it may gain further importance as its limitations are overcome.

2 Mathematical Formulation

2.1 Formulating the problem

Our problem is expressed as:

$$A|x\rangle = |b\rangle, \tag{1}$$

where A is an $N \times N$ hermitian matrix, and $|x\rangle$ and $|b\rangle$ are normalized vectors. Using the spectral decomposition theorem, we can express A as:

$$A = \sum_{j=0}^{N-1} \lambda_j |u_j\rangle\langle u_j|, \tag{2}$$

Since A is hermitian, its eigenbasis is orthonormal. We can also calculate the inverse of A as:

$$A^{-1} = \sum_{j=0}^{N-1} \lambda_j^{-1} |u_j\rangle\langle u_j|. \tag{3}$$

Further, we express $|b\rangle$ in the eigenbasis of A as:

$$|b\rangle = \sum_{i=0}^{N-1} b_i |u_i\rangle. \tag{4}$$

Now, the solution vector for Eq. (1) is given by:

$$|x\rangle = A^{-1}|b\rangle.$$

By substituting in equations (3) and (4) into the above, we get:

$$\begin{aligned} |x\rangle &= \left(\sum_{j=0}^{N-1} \lambda_j^{-1} |u_j\rangle\langle u_j|\right)\left(\sum_{i=0}^{N-1} b_i |u_i\rangle\right) \\ &= \sum_{j=0}^{N-1}\sum_{i=0}^{N-1} \lambda_j^{-1} b_i |u_j\rangle\langle u_j|u_i\rangle. \end{aligned}$$

Hence:

$$|x\rangle = \sum_{j=0}^{N-1} \lambda_j^{-1} b_j |u_j\rangle. \tag{5}$$

Note that $|x\rangle$ is normalized, hence:

$$\sum_{j=0}^{N-1} |\lambda_j^{-1} b_j|^2 = 1 \tag{6}$$

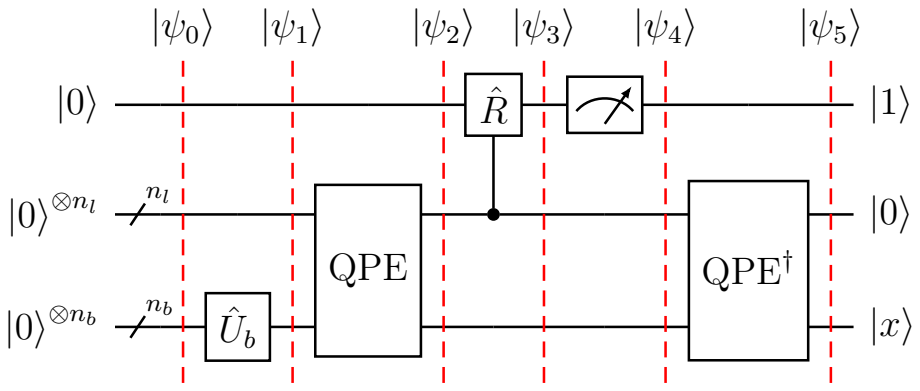

Figure 1: Circuit diagram.

2.2 Tracing the evolution of the state

We can trace the evolution of the state in a step-wise fashion with reference to Fig. 1. The circuit contains three registers: the first register is based on n_b qubits, so that it can be encoded with the value of $|b\rangle$. The size of the $|b\rangle$ vector is thus $N \times 1$ where $N = 2^{n_b}$. The second register, also known as the clock register, is based on n_l qubits. This is meant to hold the eigenvalues of A. The final register is a single ancilla qubit. In the diagram, the convention followed is that the most significant bit is at the bottom and the least significant bit is at the top.

We initialize the state as [4]:

$$|\psi_0\rangle = |0\rangle^{\otimes n_b}|0\rangle^{\otimes n_l}|0\rangle.$$

We encode the information for $|b\rangle$ onto the n_b register using a unitary operation $\hat{U}_b$ such that $\hat{U}_b|0\rangle = |b\rangle$ to obtain the following:

$$|\psi_1\rangle = |b\rangle^{\otimes n_b}|0\rangle^{\otimes n_l}|0\rangle.$$

Next, we apply the Quantum Phase Estimation subroutine to encode the eigenvalues on to the n_l register. This results in the following state:

$$|\psi_2\rangle = |b\rangle^{\otimes n_b}|\tilde{\lambda_j}\rangle^{\otimes n_l}|0\rangle.$$

We now want to find the inverse of the eigenvalues. To this end, we apply a controlled rotation on the ancilla qubit in the top register, based on the value of the n_l register [1]. Thus we obtain following state:

$$|\psi_3\rangle = \sum_{j=0}^{N-1} b_j|u_j\rangle|\tilde{\lambda_j}\rangle \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}}|0\rangle + \frac{C}{\lambda_j}|1\rangle\right).$$

In the above equation, C is a normalization constant for the third register. The next step is to perform a measurement on the third register. The desired outcome is $|1\rangle$, which will result in the following state:

$$|\psi_4\rangle = \frac{1}{\sqrt{\sum_{j=0}^{N-1}\left|\frac{b_j C}{\lambda_j}\right|^2}} \sum_{j=0}^{N-1} b_j|u_j\rangle|\tilde{\lambda_j}\rangle \left(\frac{C}{\lambda_j}|1\rangle\right),$$

where the pre-factor is there to ensure normalization post measurement. If the measurement had resulted in the $|0\rangle$ state, we would have repeated the

procedure until we obtained the desired state. With a little manipulation, the state $|\psi_4\rangle$ becomes:

$$|\psi_4\rangle = \frac{C}{\sqrt{\sum_{j=0}^{N-1} \left|\frac{b_j C}{\lambda_j}\right|^2}} \sum_{j=0}^{N-1} b_j \lambda_j^{-1} |u_j\rangle |\tilde{\lambda_j}\rangle |1\rangle.$$

If C is real, we can cancel it out; secondly, we make use of the normalization condition from Eq. (6) to obtain the following simplified expression:

$$|\psi_4\rangle = \sum_{j=0}^{N-1} b_j \lambda_j^{-1} |u_j\rangle |\tilde{\lambda_j}\rangle |1\rangle.$$

We then enter the uncompute stage, where we apply the inverse phase estimation, hence returning the n_l register to the $|0\rangle$ state. The state now becomes:

$$|\psi_5\rangle = \sum_{j=0}^{N-1} b_j \lambda_j^{-1} |u_j\rangle |0\rangle^{\otimes n_l} |1\rangle.$$

Now we know from Eq. (5) that $|x\rangle = \sum_{j=0}^{N-1} \lambda_j^{-1} b_j |u_j\rangle$, so:

$$|\psi_5\rangle = \sum_{j=0}^{N-1} |x\rangle |0\rangle^{\otimes n_l} |1\rangle.$$

We can see that the final state $|\psi_5\rangle$ is proportional to our solution vector.

3 Implementation Methodology

In this section, we explain the gate level implementation of the HHL algorithm and present a worked example simulated on python.

Here we have chosen a 2×2 hermitian matrix A such that:

$$A = \begin{pmatrix} \frac{3}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{3}{4} \end{pmatrix},$$

and the following $|b\rangle$ vector:

$$|b\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

The solution vector for this system is $x = \begin{pmatrix} -\frac{1}{2} \\ \frac{3}{2} \end{pmatrix}$, and the ratio of the squares of the magnitudes of its components is $1 : 9$ which we will later compare with the results of the simulation to verify its accuracy.

The eigenvectors for A are $|u_0\rangle = \begin{pmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$ and $|u_1\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$, whereas its eigenvalues are $\lambda_0 = \frac{1}{2}$ and $\lambda_1 = 1$. Using this information, we can diagonalize A as follows:

$$A = P\Lambda P^{-1}.$$

$$\therefore A = \frac{1}{2}\begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} \frac{1}{2} & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix} \tag{7}$$

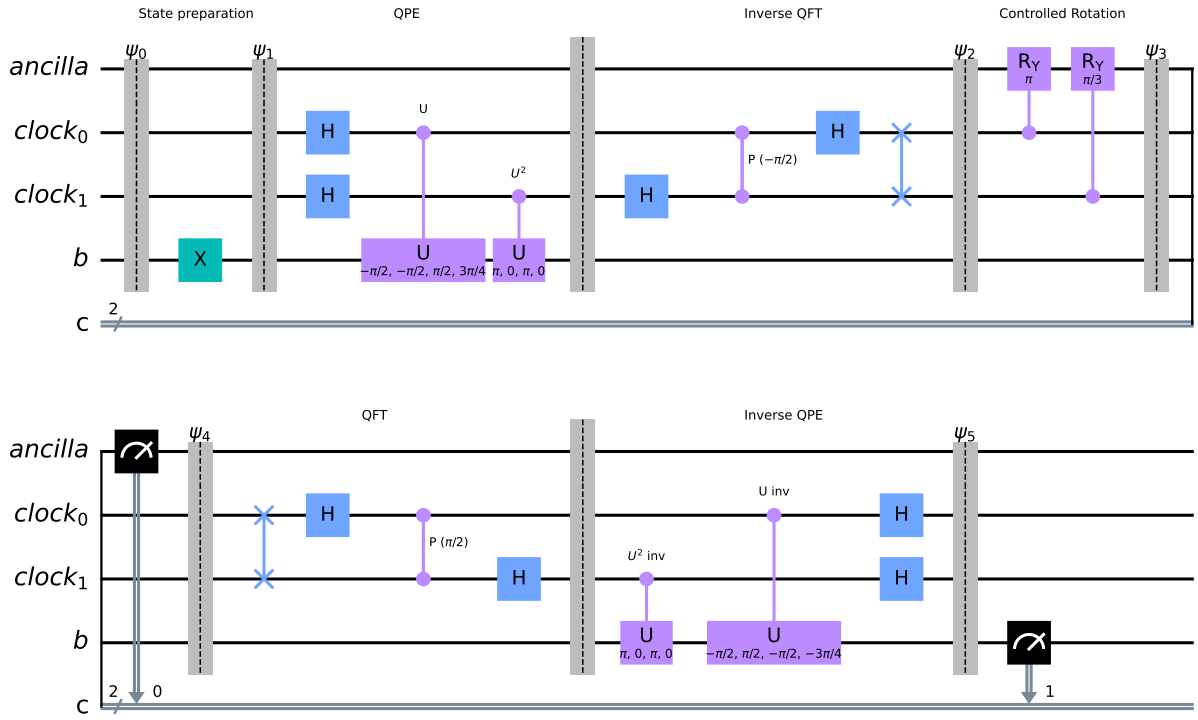

Figure 2: Simulated circuit on python.

Fig. 2 shows the python implementation of the algorithm for the above chosen example. The state labels correspond to those used in Fig. 1 for consistency. The code used to generate this circuit is provided in Appendix A. Detailed calculations for state evolution for our specific example are presented in Appendix B.

We will now go through the various stages of the algorithm, first providing a

general explanation of the procedure and then its application to our chosen example.

3.1 Encoding $|b\rangle$

From a practical stand point, this can be one of the more challenging parts of the HHL algorithm implementation. We assume that there exists an operator that can efficiently encode the vector $|b\rangle$ onto the first register, or that $|b\rangle$ is the output of a different algorithm which feeds into our circuit.

In our example, where $b = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, the application of a simple NOT gate will suffice.

3.2 Hamiltonian Simulation

In quantum mechanics, when given a Hamiltonian H, we can write the time evolution of a quantum state as $|\psi(t)\rangle = e^{-iHt/\hbar}|\psi(0)\rangle$, where $e^{-iHt/\hbar}$ is a unitary operator.

We can treat matrix A which is one of the main inputs of the HHL algorithm as a Hamiltonian, the only precondition being that A is hermitian, as is the case for a true Hamiltonian. However, quantum gates can only implement unitary operations. Thus, we modify our hermitian matrix to form $U = e^{iAt}$, which is unitary. We can now use U for our Quantum Phase Estimation subroutine. There are multiple ways to implement this operator at a gate level, depending on the complexity of the Hamiltonian. For our numerical example involving a single qubit vector $|b\rangle$, U can be implemented on Qiskit using the following four-parameter gate, with the appropriate choice of input parameters [4]:

$$U_{\theta\phi\gamma\lambda} = \begin{pmatrix} e^{i\gamma}\cos\frac{\theta}{2} & -e^{i(\gamma+\lambda)}\sin\frac{\theta}{2} \\ e^{i(\gamma+\phi)}\sin\frac{\theta}{2} & e^{i(\gamma+\lambda+\phi)}\cos\frac{\theta}{2} \end{pmatrix}. \tag{8}$$

Any single-qubit unitary operation can be carried out using the above operator, and it is trivial to modify it to form a control U gate for use in QPE:

$$U_c = U \otimes |1\rangle\langle 1| + I \otimes |0\rangle\langle 0|.$$

3.3 Quantum Phase Estimation

A unitary matrix acting on its eigenvector is represented as follows:

$$U|u\rangle = e^{i2\pi\phi}|u\rangle. \tag{9}$$

If the unitary matrix is of the form $U = e^{iAt}$, with A being a hermitian matrix bearing the eigenvalues λ_i, then Eq. (9) can be written as:

$$e^{iAt}|u\rangle = e^{i\lambda_i t}|u\rangle. \tag{10}$$

Comparing Eq. (9) and (10), we obtain:

$$\phi_i = \frac{\lambda_i t}{2\pi}. \tag{11}$$

If the input vector is not an eigenvector of the unitary operator, it can be expressed as a linear combination of the eigenvectors, and due to the linearity of operators, the transformation acts on each of these eigenvectors.

The Quantum Phase Estimation (QPE) process acts on the clock register as $|0\rangle^{\otimes n_l} \rightarrow |\phi_i 2^{n_l}\rangle = |\tilde{\lambda_i}\rangle$. Using Eq. (11), we can write the output of the QPE as:

$$|\tilde{\lambda_i}\rangle = \frac{\lambda_i t 2^{n_l}}{2\pi}.$$

In our example, $n_l = 2$ and, in order to obtain integer values, t is chosen to be π. Hence we obtain:

$$\begin{aligned} |\tilde{\lambda_0}\rangle &= |1_{10}\rangle = |01_2\rangle, \\ |\tilde{\lambda_1}\rangle &= |2_{10}\rangle = |10_2\rangle. \end{aligned} \tag{12}$$

This is an instance of Basis Encoding, where the eigenvalue is mapped onto the computational basis of the qubits, which involves the use of the binary representation of the eigenvalues [1].

Next, given that A is a hermitian matrix we can exponentiate it to create a unitary operator $U = e^{iAt}$. We can determine the exact form of U by comparing with Eq. (7):

$$U = \frac{1}{2}\begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} i & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}$$

where the diagonal elements of the second matrix are obtained by:

$$\begin{aligned} e^{i\lambda_0 t} &= e^{i\frac{\pi}{2}} = i, \\ e^{i\lambda_1 t} &= e^{i\pi} = -1. \end{aligned}$$

Hence:

$$U = \frac{1}{2}\begin{pmatrix} -1+i & -1-i \\ -1-i & -1+i \end{pmatrix}, \tag{13}$$

and

$$U^2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{14}$$

In order to implement U and U^2 in the simulated circuit in Fig. 2, we make use of the four parameter unitary gate from Eq. (8).

By setting $\theta = -\frac{\pi}{2},\ \phi = -\frac{\pi}{2},\ \lambda = \frac{\pi}{2}$, and $\gamma = \frac{3\pi}{4}$ in Eq. (8), we can implement U. Next, by setting $\theta = \pi,\ \phi = 0,\ \lambda = \pi$, and $\gamma = 0$, we can implement U^2.

Applying the inverse of QPE at the end of the algorithm is a simple matter of finding the inverse of each unitary gate and applying it in reverse order. We can clearly see from Eq. (14) that U^2 is real and symmetric, hence $(U^2)^{-1} = (U^2)^\dagger = U^2$. Next, we can easily find the inverse of U by conjugating and transposing it:

$$U^{-1} = U^\dagger = \frac{1}{2}\begin{pmatrix} -1-i & -1+i \\ -1+i & -1-i \end{pmatrix}.$$

We can implement this using the four parameter arbitrary unitary gate from Eq. (8) by setting $\theta = -\frac{\pi}{2},\ \phi = \frac{\pi}{2},\ \lambda = -\frac{\pi}{2}$, and $\gamma = -\frac{3\pi}{4}$.

3.4 Ancilla Rotation

At this stage of the algorithm, we wish to encode the inverse of the calculated eigenvalue onto the amplitude of the ancilla register as follows:

$$|0\rangle \to \frac{1}{\tilde{\lambda}}|0\rangle.$$

Such a transformation, however, is not possible. Instead, we apply the following change on the ancilla qubit:

$$|0\rangle \to \sqrt{1 - \left(\frac{C}{\tilde{\lambda}}\right)^2}\,|0\rangle + \frac{C}{\tilde{\lambda}}|1\rangle, \tag{15}$$

In order to affect the above change, we must apply the controlled rotation $R_y(\theta_i)$. C must be chosen such that it is not larger than the smallest eigenvalue, so as not to disturb normalization. In our example, we set $C = 1$.

By comparing Eq. (15) with the representation of an arbitrary quantum state $|\psi\rangle = \cos(\frac{\theta}{2}) + e^{i\phi}\sin(\frac{\theta}{2})$, we can determine the required angle of rotation:

$$\theta_i = 2\arcsin\frac{1}{\tilde{\lambda}_i}. \tag{16}$$

Hence, for $\tilde{\lambda_0} = 1$, we find that $\theta_0 = \pi$, and the required rotation is $R_y(\pi)$. For $\tilde{\lambda_1} = 2$, we find that $\theta_1 = \frac{\pi}{3}$, and the required rotation is $R_y(\frac{\pi}{3})$.

In effect, we want to apply [4]:

$$\Theta(c_1 c_0) = \frac{\pi}{3} c_1 + \pi c_0,$$

where $\Theta(c_1 c_0)$ is a function that takes the values encoded in the clock register in binary form, and outputs the required rotation angle θ. This is achieved using two controlled rotation gates as shown in Fig. 2.

In contrast with QPE, the ancilla rotation is an instance of Amplitude Encoding, where the relevant information, i.e., the inverse of the eigenvalue, is encoded into the *amplitude* of a qubit [1].

3.5 Time and resource efficiency

As stated in the introduction, classical algorithms such as Gaussian elimination can solve a system of equations in $\mathcal{O}(N^3)$ time. If the matrix is sparse and well-conditioned, and we are only interested in a summary statistic of the solution vector rather than the exact solution, the run time is reduced to $\mathcal{O}(N\sqrt{\kappa})$, [2] where κ, the condition number, is the ratio of the smallest eigenvalue to the largest one. The condition number indicates how sensitive the system is to perturbations.

Using the HHL algorithm, we can get a run time of $\mathcal{O}(\kappa^2 \log \frac{N}{\epsilon})$ [2]. Here we assume that the matrix is s-sparse, which means that there are at most s non zero elements in a given row, and ϵ is the error that we are willing to tolerate in the solution.

Focusing on individual sections of the algorithm, the quantum phase estimation stage is one of the dominant sources of error. The computational complexity depends upon the extent of accuracy we require in eigenvalues and the number of qubits required to encode eigenvalues. Computational complexity scales with the factor $\mathcal{O}(1/\epsilon)$, where ϵ is the desired accuracy of the estimated eigenvalues [2]. The QPE part of the HHL is comparatively resource intensive as it uses a large number of qubits and gates, which increases with the size of the system.

The eigenvalue inversion stage of the algorithm can also be resource intensive, as it it a non-unitary transformation, and may require multiple attempts to produce the desired $|1\rangle$ state.

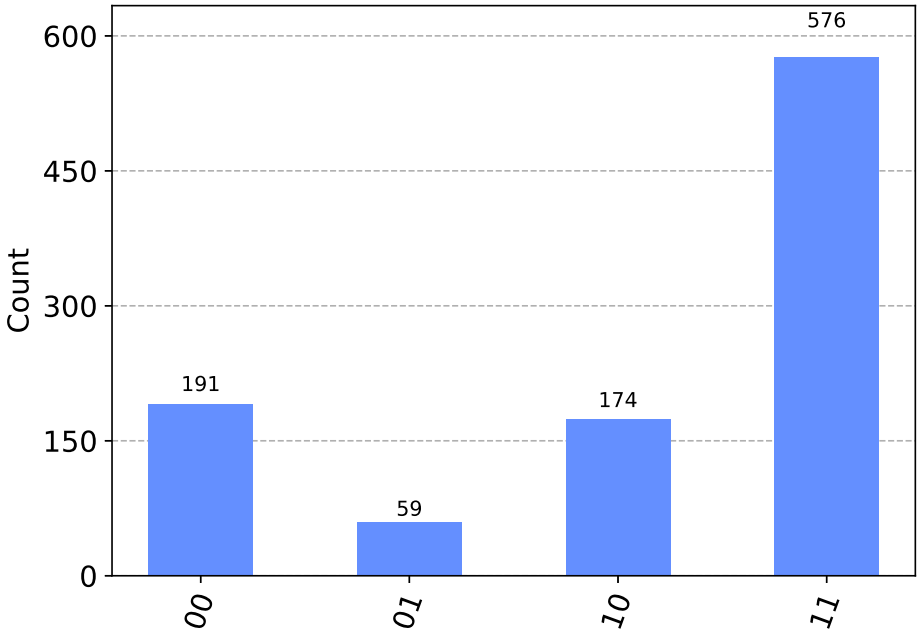

Figure 3: Simulator results for 1000 shots.

4 Results

We ran the circuit shown in Fig. 2, with our chosen example as input, first on a high performance simulator, and then proceeded to run the same circuit on real quantum hardware.

The results of the simulation are presented in Fig. 3 with 1000 shots fired. Recall that we are only interested in case where the measurement of the ancilla results in $|1\rangle$. Therefore we look at the bins marked 01 and 11, whose heights correspond to the magnitude squared of the first and second coefficients of the solution vector $|x\rangle$ respectively.

The calculated ratio thus obtained is $1 : 9.762$, which is very close to the true ratio of the solution vector based on the classical solution, i.e., $1 : 9$ from Section 3.

Running the same circuit on actual quantum hardware generated the results presented in Fig. 4. The corresponding ratio in this case is $1 : 1.234$ which is significantly different from the true value. This difference is due to noise from the environment and decoherence arising from the usage of a real quantum computer as opposed to a simulator [5].

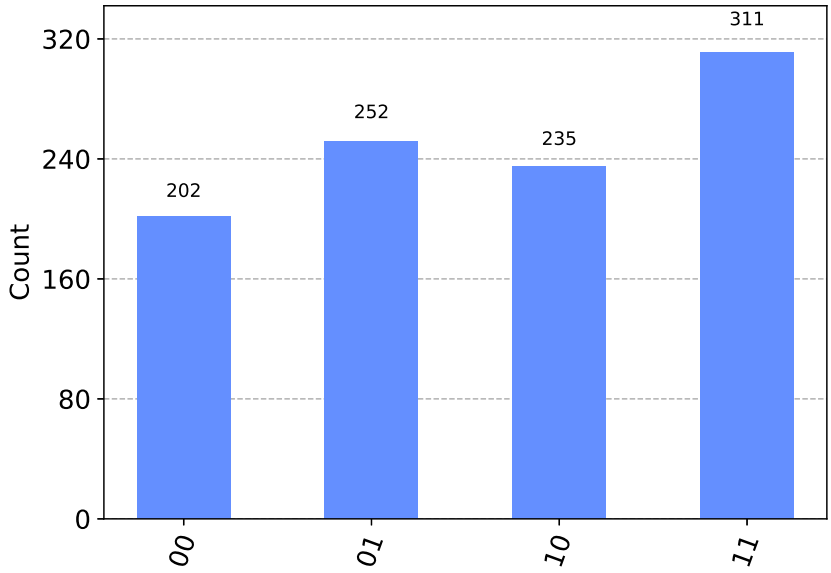

Figure 4: Quantum computer results for 1000 shots.

The circuit we have implemented can be generalized to some extent, as we can input any 2×1 vector $|b\rangle$ without needing to change other elements of the circuit. (Note that solving for a different matrix A would require significant changes to the circuit with our current implementation methodology).

Hence we input several different vectors and compared the solutions from the simulator to the classical solutions. A summary of the results is presented in Table 1.

Input state	**Probability Ratio of $\|x\rangle$**		**Relative Error**
	Classical	Simulation	
$\|+\rangle$	$1:1$	$1:1.315$	31.5%
$\|-\rangle$	$1:1$	$1:0.992$	0.8%
$\|0\rangle$	$1:0.111$	$1:0.119$	7.2%
$\|1\rangle$	$1:9$	$1:9.762$	8.5%

Table 1: Results for different inputs of vector $|b\rangle$

5 Conclusion

We have explained the working and implementation of the HHL algorithm and demonstrated its usage with an example. We then compared the solution thus obtained with the classically obtained solution to check the accuracy of the algorithm. Further, we contrasted the solution obtained via a high performance quantum simulator against results from a real 7-qubit quantum computer and explained the discrepancy in the results. Finally we subjected our circuit to a range of inputs to verify its flexibility and found the results to be accurate.

References

[1] Bojia Duan, Jiabin Yuan, Chao-Hua Yu, Jianbang Huang, and Chang-Yu Hsieh. A survey on HHL algorithm: From theory to application in quantum machine learning. *Physics Letters A*, 384(24):126595, 2020.

[2] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009.

[3] Abhijith J., Adetokunbo Adedoyin, John Ambrosiano, Petr Anisimov, William Casper, Gopinath Chennupati, Carleton Coffrin, Hristo Djidjev, David Gunter, Satish Karra, Nathan Lemons, Shizeng Lin, Alexander Malyzhenkov, David Mascarenas, Susan Mniszewski, Balu Nadiga, Daniel O'malley, Diane Oyen, Scott Pakin, Lakshman Prasad, Randy Roberts, Phillip Romero, Nandakishore Santhi, Nikolai Sinitsyn, Pieter J. Swart, James G. Wendelberger, Boram Yoon, Richard Zamora, Wei Zhu, Stephan Eidenbenz, Andreas Bärtschi, Patrick J. Coles, Marc Vuffray, and Andrey Y. Lokhov. Quantum algorithm implementations for beginners. *ACM Transactions on Quantum Computing*, 3(4):1–92, jul 2022.

[4] Hector Jose Morrell Jr, Anika Zaman, and Hiu Yung Wong. Step-by-step hhl algorithm walkthrough to enhance the understanding of critical quantum computing concepts. *arXiv preprint arXiv:2108.09004*, 2021.

[5] Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2023.

A Python Code

```
#################### importing ####################################
from qiskit import QuantumCircuit, Aer, execute
from qiskit.visualization import plot_histogram
from qiskit import QuantumRegister, ClassicalRegister
from qiskit.quantum_info import Statevector
from matplotlib import pyplot as plt
from pylatexenc import *
from qiskit.circuit.library import QFT
import numpy as np
from numpy import pi
from matplotlib import pyplot as plt

#################### initializing ##################################
#################### initializing  (part (a)) ######################
######## no of qubits in each register#########

n_a = 1  ## ancilla
n_l = 2  ## clock register
n_b = 1  ## input vector

#################### initializing  (part (b)) ######################
######## initializing all registers ##########
b = QuantumRegister(n_b, name='b')
clock = QuantumRegister(n_l, name='clock')
ancilla = QuantumRegister(n_a, name='ancilla')
measurement = ClassicalRegister(2, name='c')

hhl = QuantumCircuit(ancilla, clock, b, measurement)
hhl.barrier(label='$\psi_0$')

#################### Encoding b ####################################
hhl.x(b)
hhl.barrier(label='$\psi_1$')

#################### Quantum Phase Estimation ######################
######## Applying controlled rotations ######
hhl.h(clock)
hhl.cu(-pi/2, -pi/2, pi/2, 3/4*pi, clock[0], b, label='U')
hhl.cu(pi, 0, pi, 0, clock[1], b, label=r'$U^2$')
hhl.barrier()

####### Inverse Quantum Fourier Transform ###
hhl.h(clock[1])
```

```
hhl.cp(-np.pi/2, clock[0], clock[1])
hhl.h(clock[0])
hhl.swap(clock[0], clock[1])
hhl.barrier(label='$\psi_2$')

#################### ancilla rotation ##############################
hhl.cry(pi, clock[0], ancilla)
hhl.cry(pi/3, clock[1], ancilla)
hhl.barrier(label='$\psi_3$')

#################### ancilla measurement ###########################
hhl.measure(ancilla, measurement[0])
hhl.barrier(label='$\psi_4$')

#################### inverse QPE ###################################
######## Quantum Fourier Transform ###########
hhl.swap(clock[0], clock[1])
hhl.h(clock[0])
hhl.cp(np.pi/2, clock[0], clock[1])
hhl.h(clock[1])
hhl.barrier()

######## Applying controlled rotations ######
hhl.cu(pi, 0, pi, 0, clock[1], b, label=r'$U^2$ inv')
hhl.cu(-pi/2, pi/2, -pi/2, -3/4*pi, clock[0], b, label='U inv')
hhl.h(clock)
hhl.barrier(label='$\psi_5$')

#################### measure x #####################################
hhl.measure(b,measurement[1])

#################### labelling #####################################
fig = plt.figure(figsize=(16, 8))
ax = fig.add_subplot()
height = 0.8
ax.text(0.5, height, 'State preparation', size=8)
ax.text(4.5, height, 'QPE', size=8)
ax.text(10, height, 'Inverse QFT', size=8)
ax.text(14, height, 'Controlled Rotation', size=8)
ax.text(4, -5.5, 'QFT', size=8)
ax.text(10, -5.5, 'Inverse QPE', size=8)

# hhl.draw('mpl', ax=ax, fold=17, filename='python_circuit.eps')
```

B State evolution for chosen example

We initialize the state as follows:

$$|\psi_0\rangle = |0\rangle|00\rangle|0\rangle.$$

After state preparation, we get:

$$|\psi_1\rangle = |1\rangle|00\rangle|0\rangle.$$

Expressing $|b\rangle$ in the eigenbasis of A we obtain:

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}\left(|u_0\rangle + |u_1\rangle\right)|00\rangle|0\rangle = \frac{1}{\sqrt{2}}|u_0\rangle|00\rangle|0\rangle + \frac{1}{\sqrt{2}}|u_1\rangle|00\rangle|0\rangle.$$

After performing phase estimation, the eigenvalues are encoded into the clock register as follows:

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}|u_0\rangle|01\rangle|0\rangle + \frac{1}{\sqrt{2}}|u_1\rangle|10\rangle|0\rangle.$$

The ancilla rotation produces the following state:

$$|\psi_3\rangle = \frac{1}{\sqrt{2}}|u_0\rangle|01\rangle\left(\sqrt{1-\left(\frac{1}{1}\right)^2}|0\rangle + \frac{1}{1}|1\rangle\right) + \frac{1}{\sqrt{2}}|u_1\rangle|10\rangle\left(\sqrt{1-\left(\frac{1}{2}\right)^2}|0\rangle + \frac{1}{2}|1\rangle\right),$$

$$\therefore |\psi_3\rangle = \frac{1}{\sqrt{2}}|u_0\rangle|01\rangle|1\rangle + \frac{\sqrt{3}}{2\sqrt{2}}|u_1\rangle|10\rangle|0\rangle + \frac{1}{2\sqrt{2}}|u_1\rangle|10\rangle|1\rangle.$$

Assuming the ancilla measurement produces the desired state of $|0\rangle$, we obtain:

$$|\psi_4\rangle = \frac{\sqrt{8}}{5}\left(\frac{1}{\sqrt{2}}|u_0\rangle|01\rangle|1\rangle + \frac{1}{2\sqrt{2}}|u_1\rangle|10\rangle|1\rangle\right),$$

where the pre-factor is the normalization factor.

$$\Rightarrow |\psi_4\rangle = \frac{2}{\sqrt{5}}|u_0\rangle|01\rangle|1\rangle + \frac{1}{2\sqrt{5}}|u_1\rangle|10\rangle|1\rangle.$$

Next, the inverse of the phase estimation is performed to return the clock register to its default state as follows:

$$|\psi_5\rangle = \frac{2}{\sqrt{5}}|u_0\rangle|00\rangle|1\rangle + \frac{1}{2\sqrt{5}}|u_1\rangle|00\rangle|1\rangle,$$

$$\therefore |\psi_5\rangle = \left(\frac{2}{\sqrt{5}}|u_0\rangle + \frac{1}{2\sqrt{5}}|u_1\rangle\right)|00\rangle|1\rangle.$$

Finally, we convert the first register back into the Zeeman basis from the eigenbasis as follows:

$$|\psi_5\rangle = \left(-\frac{1}{\sqrt{10}}|0\rangle + \frac{3}{\sqrt{10}}|1\rangle\right)|00\rangle|1\rangle.$$

From the above, we can see that the ratio of the probability of obtaining $|0\rangle$ to that of $|1\rangle$ is $1:9$, in accordance with the classical solution.