SOLVING LINEAR EQUATIONS WITH QISKIT

A PREPRINT

Jaime Santos
A71739

Américo da Costa
A72121

June 11, 2019

ABSTRACT

This paper serves as an analysis of a quantum algorithm with the purpose of solving linear systems of equations. Here we attempt to implement an efficient and generic algorithm in Qiskit, the HHL algorithm. The theoretical basis of this algorithm will be explained as simply as possible as well as the detailed construction of the quantum circuits using Qiskit and Python.

Contents

1 Introduction

The quantum computer has been gaining popularity and viability over the years because more and more algorithms have been shown to have a gain in complexity over the classical counterpart. Some examples are the Shor algorithm and Grover algorithm [1].
Solving a system of linear equations with N variables is a crucial step in many fields like science or engineering. However, a lot of cases involving these systems work with Big Data, which results in a large number of variables. With this being said, and since the best known classical algorithms require a time of O(N), quantum computing came to offer an exponential speedup in solving these systems.
Harrow, Hassidim, and Lloyd (HHL) proposed an algorithm to solve linear systems exponentially faster than a classical computer in cases where one is interested in obtaining expectation values associated with the solution rather than the full solution. For an s-sparse system matrix of size $N \times N$, this algorithm can reach a desired computer accuracy within a time of O(log(N)), which provides a huge speedup for a wide range of applications that involve large-scale linear systems.

2 Theoretical Overview

Given an Hermitian $N \times N$ d-sparse matrix, the objective is to solve $A\vec{x} = \vec{b}$, where $\vec{b}$ is an unitary vector.
The following figure, obtained from [2], represents the three major stages of this algorithm:

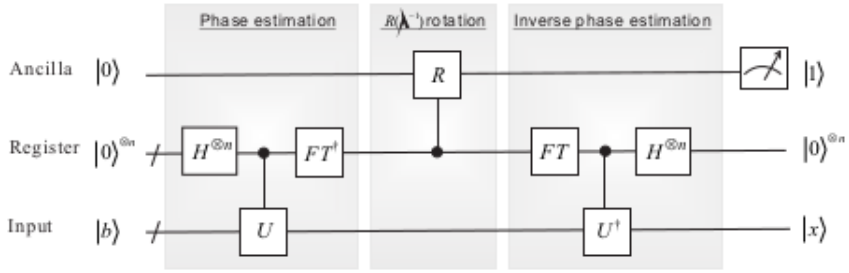

Figure 1: Three main phases of the HHL algorithm.

Both vectors $\vec{b}$ and $\vec{x}$ will be scaled to unit length to adapt this problem to quantum processing. Vector $\vec{b}$, will be represented by the quantum state $\sum_i b_i |i\rangle$ on $O(log(N))$ qubits, where $|i\rangle$ is the computational basis.
The solution, $\vec{x}$ can then be encoded in the input state as:

$$|x\rangle = cA^{-1} |b\rangle , \quad c^{-1} = \|A^{-1} |b\rangle \| \tag{1}$$

This algorithm involves three subsystems: an ancillary qubit initialized in $|0\rangle$, a working memory consisting of n qubits initialized in $|0\rangle^{\otimes n}$ and an input state $|b\rangle$.
If the circuit is successful in all stages of Figure 1, then a measurement of the ancillary qubit should return the state $|1\rangle$ and $\vec{x}$ will be encoded into $|b\rangle$.

2.1 Phase Estimation

Suppose there's an unitary operator U, operating on n-qubits, with a known eigenstate $|u\rangle$ and an unknown eigenvalue $e^{i\phi}$. In order to find the eigenvalues, which means finding the phase ϕ, one must conjure a measurement which will give information about ϕ.[3]

2.1.1 Phase Kickback

Consider the state $|\psi\rangle$ being an eigenvector of the operator U, with an eigenvalue of $e^{2\pi i\phi}$:

$$U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle \tag{2}$$

Where ϕ is the phase wanted to kick back.
In order to understand this phenomenon, the following figure is presented:

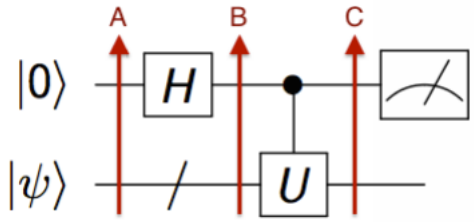

Figure 2: Phase Kickback.

Analyzing the three steps:

- At point **A** the state is $|0\rangle|\psi\rangle$
- At point **B**, the Hadamard operator transforms $|0\rangle$ into $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$. The system is therefore in the state:

$$\frac{|0\rangle+|1\rangle}{\sqrt{2}}|\psi\rangle = \frac{|0\rangle|\psi\rangle+|1\rangle|\psi\rangle}{\sqrt{2}} \tag{3}$$

- At point **C**, a controlled **U** operator is applied, transforming the state of the system to:

$$\frac{|0\rangle|\psi\rangle+|1\rangle|\psi\rangle U}{\sqrt{2}} = \frac{|0\rangle|\psi\rangle+|1\rangle|\psi\rangle e^{2\pi i\phi}}{\sqrt{2}} = \frac{|0\rangle+|1\rangle e^{2\pi i\phi}}{\sqrt{2}}|\psi\rangle \tag{4}$$

The conclusion is that state $|\psi\rangle$ remains unchanged. Applying **U** to $|\psi\rangle$ results in a "multiple" $e^{2\pi i\phi}$ gets of $|\psi\rangle$, which can be factored out. This is possible since we admit that $|\psi\rangle$ is an eigenstate of **U**.

2.1.2 Phase Estimation Algorithm

The complete phase estimation procedure is as follows:[3]

1. Prepare the t-qubit control register in state $|0\rangle$, and the target register in state $|u\rangle$ (eigenstate of **U**);
2. Perform Hadamards on the control qubits.
3. Apply a controlled-$\hat{U}^{2^j}$, from the j-th control qubit onto the n target qubits for each of the control qubits in succession.
4. Perform an inverse Fourier transform on the t control qubits and measure them in the computational basis.

Applying this to the problem of the HHL algorithm, will result in a mapping of the eigenvalues of the matrix **A** onto the working memory.
The input state $|b\rangle$, can be expanded in the basis of $|u_j\rangle$ as:

$$|b\rangle = \sum_{j=1}^{N} \beta_j |uj\rangle \tag{5}$$

Where, $|u\rangle$ is an eigenstate of A, and $\beta_j = \langle u_j|b\rangle$. This protocol applied to the input, using the working memory as a control, will result in the state:

$$\sum_{j=1}^{N} \beta_j |u_j\rangle |\lambda_j\rangle \tag{6}$$

2.2 Extracting the inverse eigenvalues

In this step the objective is to extract the eigenvalues of $\mathbf{A}^{-1}$, which means extracting λ_j^{-1} from $|\lambda_j\rangle$. This is achieved by applying an appropriate controlled-$R(\lambda^{-1})$ rotation (which will be explained later), using the working memory as a control and the ancillary qubit as a target. This transforms the system to:

$$\sum_{j=1}^{N} \beta_j |u_j\rangle |\lambda_j\rangle \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle\right) \tag{7}$$

This involves performing a controlled Y-rotation $R_y(\theta_j) = e^{-i\theta_j Y/2}$ where $\theta_j = 2arcsin(c/\lambda_j)$ and $C \leq min_j|\lambda_j|$.[4]

2.3 Uncomputing the system

This final step involves applying the phase estimation protocol in reverse.[2]
This disentangles the register, which is reset to $|0\rangle^{\otimes n}$. The system therefore ends up in a state without the eigenvalues:

$$\sum_{j=1}^{N} \beta_j |u_j\rangle \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle\right) \tag{8}$$

In this step, measuring the ancillary qubit and post-selecting an outcome of $|1\rangle$ will result in an output state:

$$\sum_{j=1}^{N} C\left(\frac{\beta_j}{\lambda_j}\right) |u_j\rangle \tag{9}$$

Which will be proportional to the expected result state $|x\rangle$.

2.4 Run time analysis

The best classical algorithm using Gaussian elimination to find a solution for $\vec{x}$ runs in $O(N^3)$ time.[7] Given an A s-sparse and positive semi-definite matrix, the Conjugate Gradient method can be used to found the solution in $O(Nsk)$ time.
The quantum algorithm proposed by Harrow et al[8] was shown to be $O(k^2 \log N)$, where *k* is the condition number of the A matrix. The HHL algorithm only maintains its logarithmic scaling for sparse or low rank matrices, which was extended by Wossnig et al[9] to dense matrices running in $O(\sqrt{N} \log N k^2)$.

3 Qiskit Implementation

In this chapter, the group will attempt to implement the HHL algorithm using the Qiskit Python package from IBM. The focus of this section will be analyzing the quantum circuits and the results, rather than the code itself, which will be sent as an attachment.
The final subsections will be used to reflect on the challenges and shortcomings associated with our implementation as well as future improvements on our code.

3.1 Solving 2×2 Linear Equation Systems

In order to obtain the experimental results of this algorithm,the following matrix A was chosen as:

$$A_1 = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \tag{10}$$

And the input vector $|b\rangle$ as $|b_1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $|b_2\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ and $|b_3\rangle = \frac{1}{\sqrt{2}} \times \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. This will also be done for different matrices in the results section.

3.1.1 Phase Estimation

The first subroutine (phase estimation) was implemented in Qiskit by applying one Hadamard gate to each register qubit, one CR_x^{n-1} gate with the n-th register qubit as control and input qubit as target, and one Inverse Quantum Fourier Transform to the register qubits. The Hadamards will create a state of superposition on the register qubits, while the CR_x^{n-1} gates will map the input matrix eigenvalues into them. The IQFT is used to codify this eigenvalues into the working memory by having the amplitudes of the basis states as k values that satisfy:

$$\lambda_k = \frac{2\pi k}{t0} \tag{11}$$

As a first step of the phase estimation algorithm, one must implement the Hamiltonian evolution, denoted as **U** in figure 1:

$$U = \sum_{k=0}^{T-1} |k\rangle \langle k| \otimes e^{i\frac{Akt0}{T}} \tag{12}$$

Where A is the input matrix, $T = 2^n$ with *n* being the number of registers, t_0 chosen as 2π and *k* being the condition number (the ratio between A's largest and smallest eigenvalues).[2]
Since the input is a 2x2 matrix, one can simply implement a CR_x gate to represent the Hamiltonian evolution operator A (which in [3] is a CU_3 gate).
This is proven by writing the Hamiltonian A_1 as a Pauli decomposition:

$$A_1 = 2I - X \tag{13}$$

The I gates can be ignored since they apply a global phase to the system, therefore, any real 2x2 matrix can be decomposed into an $Rx(\alpha\theta)$, where α is the second element of the input matrix. With this said, the group implemented the Hamiltonian evolution as a $CU_3(\alpha\theta, \phi, \lambda)$ gate, where $\phi = -\pi/2$ and $\lambda = \pi/2$ to simulate a CR_x gate. The IQFT implementation is explained in [3].
The following figure represents the phase estimation part of the quantum algorithm, as coded by the group:

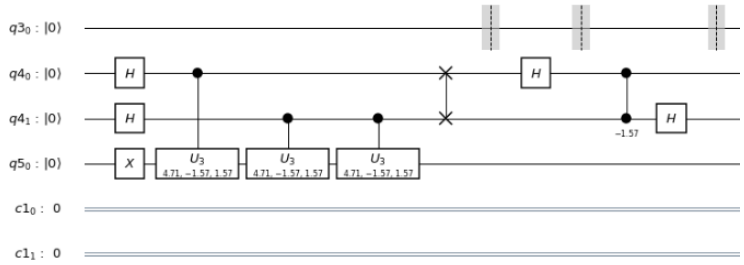

Figure 3: Phase Estimation Circuit

3.1.2 Controlled ancillary Rotation

At this point of the circuit, it is necessary to extract the reciprocal eigenvalues, which is done by performing conditional Y rotations on the ancillary qubit, which will now be in the state:

$$|Anc.\rangle = \sqrt{(1 - (C^2/\lambda^2)\,|0\rangle} + C/\lambda\,|1\rangle \tag{14}$$

This state is achieved by applying $CR_y(\frac{(n)\pi}{2^{r-1}})$ gates between every register qubit (control) and the ancillary qubit (target), where *r* is the fidelity number chosen as 4 [5] and *n* is initially the number of control registers and decrements by 1 each time we change the control register index. In this implementation, *r* was chosen to be 4, which is the optimal value when dealing with 2 qubit registers. Note that the results accuracy scales with the increase of *r*, but the probability of obtaining the result lowers, making the *r* choice for the algorithm crucial, since it must be balanced between these 2 parameters.[6]
The coded block results in:

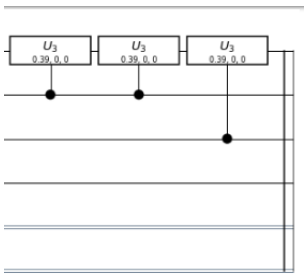

Figure 4: Controlled Y rotations

3.1.3 Uncomputation and Measuring

The only blocks left to implement at this point are the ones responsible for uncomputing the system and performing the measurements.
The uncomputation is done by implementing the inverse phase estimation, which consists in applying the phase estimation blocks (H gates, Hamiltonian evolution and QFT) in reverse order. The Hamiltonian evolution must as well be replaced by its complex conjugate. These 2 blocks are coded into the circuit as follows:

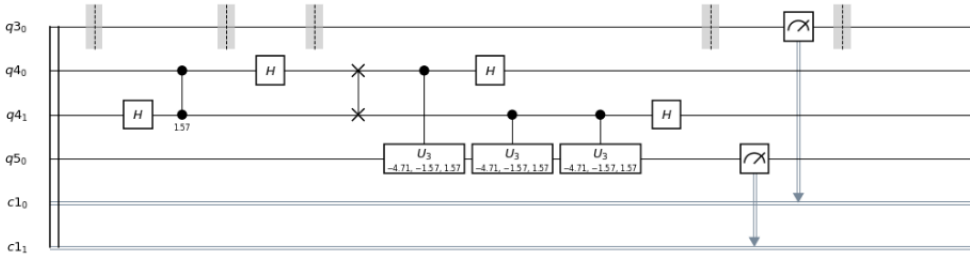

Figure 5: Uncomputation and measurement

After this, only the analysis of the measurements remains. The ancillary and the input qubits are measured in order to obtain the probabilities of the $|b\rangle$ state, which is proportional to $\vec{x}$ when the ancillary is measured as $|1\rangle$.

3.2 Solving 4×4 Linear Equation Systems

A 4x4 matrix as input demands a larger circuit with 7 qubits, and the one used in this implementation is:

$$A = \frac{1}{4}\begin{pmatrix} 15 & 9 & 5 & -3 \\ 9 & 15 & 3 & -5 \\ 5 & 3 & 15 & -9 \\ -3 & -5 & -9 & 15 \end{pmatrix} \tag{15}$$

The input vectors $|b_1\rangle$ and $|b_2\rangle$ are initialized at the state $|b_j\rangle = \frac{1}{\sqrt{2}} \times \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, which leads to the initial input state:

$$|b\rangle = \frac{1}{2} \times \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \tag{16}$$

The input register must have 2 qubits to extract all the 3 proportion values, and the working memory needs 4 qubits to encode the eigenvalues. Using the same algorithm as in the 2×2 matrix scenario results in the following circuit:

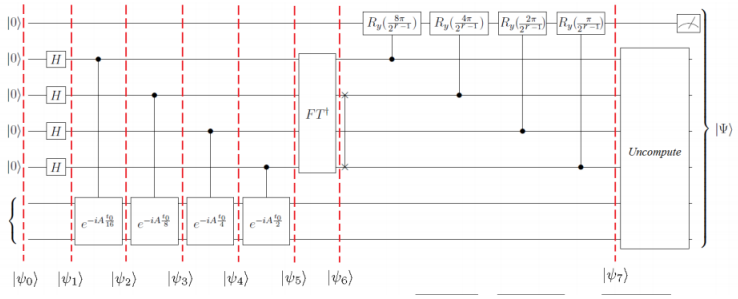

Figure 6: The circuit for a 4x4 Matrix [5]

In this case, the Hamiltonian evolution obviously won't be decomposed as a single CR_x gate. Since this decomposition is way harder to find, the group opted to implement the decomposition presented in [5], which is shown in the figure below:

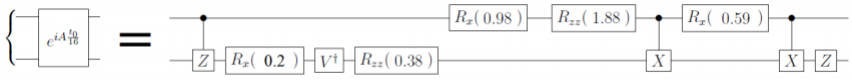

Figure 7: Hamiltonian Operator Decomposition [5]

This decomposition is implemented by giving every gate inside the Hamiltonian Evolution a control (or extra control) on the memory register qubit being used at the moment, so that the block becomes controlled as shown in Figure 6. Some of the gates inside the decomposition are decomposed again so they can be implemented in qiskit, as for the CCz, $Cv+$ and CR_zz gates which can be decomposed as follows:

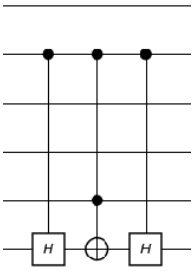

Figure 8: Decomposed $controlled^2$-Z gate

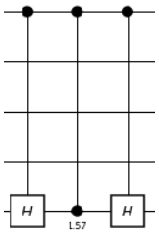

Figure 9: Decomposed controlled-v^+ gate

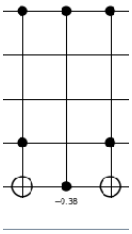

Figure 10: Decomposed controlled-Rzz gate

Note that the full circuit will not be presented here, due to the large size of the Hamiltonian decomposition in the phase estimation protocol (it will be in the notebook).
The CR_x gates were implemented by using Cu_3 gates like in the 2x2 matrix input scenario and the 2 Cx gates become toffoli gates. For the controlled-Y rotations subroutine, r was chosen as 6 [5], which is found to be the optimal value considering the parameters mentioned in section 3.1.2. Right before applying these rotations, the authors also apply a SWAP gate between the second and fourth working memory qubits to invert the eigenvalues to their reciprocals.

4 Results and Discussion

4.1 2×2 Linear Equation Systems

4.1.1 Matrix A_1

It is now presented the result of measuring the circuit, for each $|b\rangle$, 8192 times:

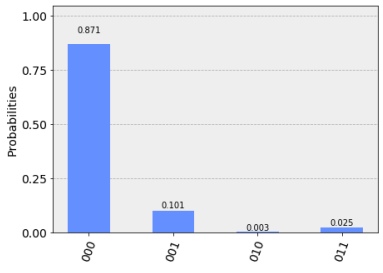

Figure 11: $|b1\rangle$

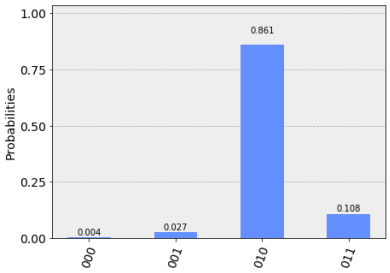

Figure 12: $|b2\rangle$

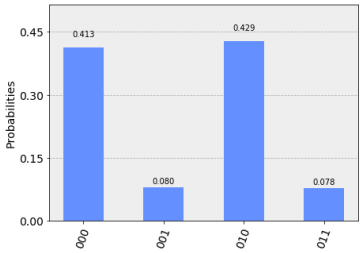

Figure 13: $|b3\rangle$

Only the probabilities of the two states where the ancillary qubit is measured as 1 matter, since those represent the probabilities of measuring the states where the solution is encoded. It is now necessary to transform these probabilities into their corresponding amplitude by doing:

$$A_j = \sqrt{P_j} \tag{17}$$

The ratio between the two amplitudes of this states represents a value k, that satisfies $|b\rangle = \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a \\ ka \end{pmatrix}$, where $k = \frac{A|0\rangle}{B|1\rangle} \otimes |1\rangle$.

Finding the value k provides the direction of the solution vector, simplifying the linear equation system into a linear equation (only a is left to find).
The resulting proportions are:

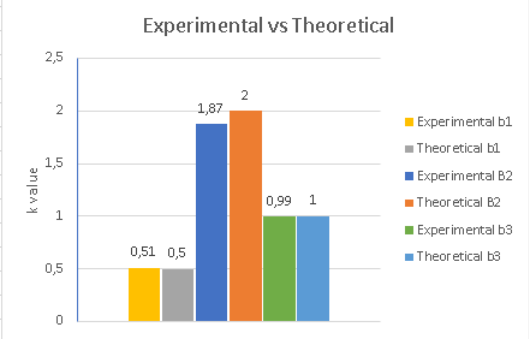

Figure 14: Experimental proportions vs Theoretical

Where each experimental value was obtained by doing several measurements of the probabilities of the states, converting them into amplitudes and plotting the average value in Excel (sent as an attachment).

4.1.2 Matrix A_2

The group now chooses the matrix A_2 to be the following:

$$A_2 = \begin{pmatrix} 1.5 & 1 \\ 1 & 1.5 \end{pmatrix} \tag{18}$$

The measurements results were as follows:

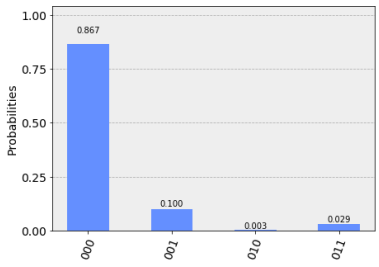

Figure 15: $|b1\rangle$

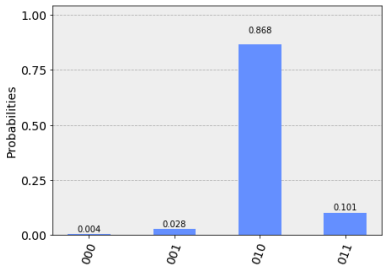

Figure 16: $|b2\rangle$

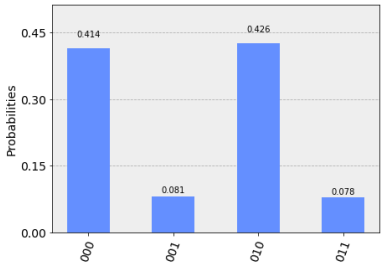

Figure 17: $|b3\rangle$

The corresponding k values calculated resulted in:

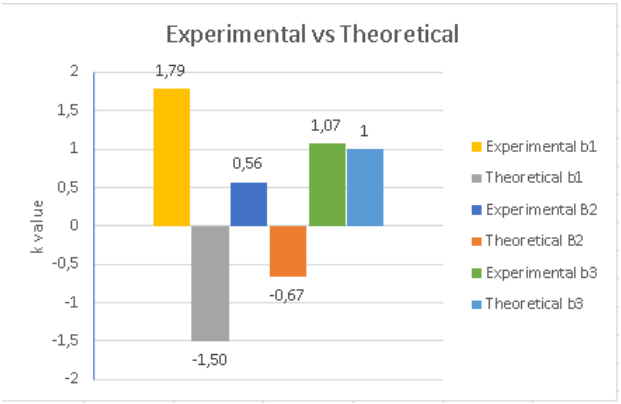

Figure 18: Experimental proportions vs Theoretical

The results of the measuring are satisfactory, since the group obtained a low deviation of the experimental results from the theoretical values. This error is probably the result of the implementation of the Hamiltonian evolution. However, the A_2 input case results show that the group could not obtain the correct sign of the k values, since the probabilities measured are always positive. We believe this happens due to the final state of $|b\rangle$ being $|b\rangle = |0\rangle - |1\rangle$ or $|b\rangle = |1\rangle - |0\rangle$, which makes k negative.

4.2 4×4 Linear Equation Systems

The measuring of 8192 shots resulted in the image below:

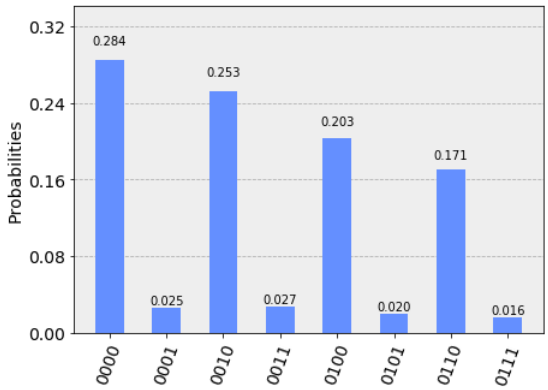

Figure 19: $|b3\rangle$

The k_j values were calculated from the probabilities measured and the results were the following:

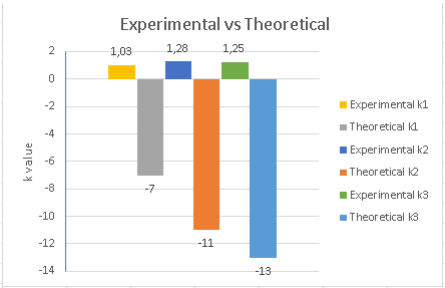

Figure 20: Experimental vs Theoretical

The experimental results are nothing like what was expected. This means the group's version of the circuit was badly implemented, since in [5] Cao et al obtain results very close to the theoretical values.
The group believes that the error resides in a bad implementation of the Hamiltonian decomposition of the circuit, since this phase has an associated error due to the approximation of the values.

4.3 Future Work

Future work on this project involves a better Hamiltonian simulation operator decomposition for the 4×4 case, since the one used in this attempt is too large, and the results were not satisfactory. Once this was achieved, the next step would be increasing the abstraction of the generation of the circuit, which means creating a function that would receive any suitable 4×4 A matrix as input (as well as any other relevant parameters) and generate the HHL circuit accordingly.This was done for the more trivial 2×2 case, but the group recognizes that this version could also be improved.
More future work would involve expanding the range of matrices that could be solved, as was done by Wossnig et al [9].

5 Conclusion

Reflecting on the results obtained, we conclude that they could be better. For the 2×2 case, we got some consistent results however when expanding into the 4×4 this was not the case. We suspect that the root cause is the implementation of the Hamiltonian simulation. This was heavily influenced by the lack of information contained in the articles referenced in the previous sections of this paper, which also presented many challenges implementing the 2×2, specifically in the result analysis.
Reflecting on the process as a whole, we evaluate it as a positive experience. The cryptic nature of the information available and the "detective work" needed to implement and analyze the results was a very valuable experience in terms of learning, even though the results weren't as good as expected.

A Matrices

$$R_x = \begin{pmatrix} \cos\theta/2 & -i\sin\theta/2 \\ -i\sin\theta/2 & \cos\theta/2 \end{pmatrix}$$

$$R_y = \begin{pmatrix} \cos\theta/2 & -\sin\theta/2 \\ \sin\theta/2 & \cos\theta/2 \end{pmatrix}$$

$$R_z = \begin{pmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{-i\phi/2} \end{pmatrix}$$

$$CU_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & e^{i\lambda} \end{pmatrix}$$

$$CU_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{-i(\phi+\lambda)/2} & 0 & -e^{-i(\phi-\lambda)/2} \\ 0 & 0 & 1 & 0 \\ 0 & e^{i(\phi-\lambda)/2} & 1 & e^{i(\phi+\lambda)/2} \end{pmatrix}$$

Where the CU_3 matrix will be used to implement CR_x and CR_y.

References

[1] M. A. Nielsen and I. L. Chuang. Quantum Computation and Quantum Information (10th Anniversary Edition). In *Cambridge University Press, 2010*.

[2] X.-D. Cai, C. Weedbrook, Z.-E. Su, M.-C. Chen, Mile Gu, M.-J. Zhu, Li Li, Nai-Le Liu, Chao-Yang Lu, Jian-Wei Pan. Experimental quantum computing to solve systems of linear equations. In *arXiv:1302.4310v2*.

[3] A. Rodrigues. Period finding. Phase kickback. Quantum phase estimation. In *arca.di.uminho.pt/quantum-computation-1819/*.

[4] Yudong Cao, Anmer Daskin, Steven Frankel and Sabre Kais. Quantum circuits for solving linear systems of equations. In *arXiv:1110.2232*.

[5] Yudong Cao, Anmer Daskin, Steven Frankel and Sabre Kais. Quantum circuits for solving linear systems of equations. In *arXiv:1110.2232v2* April 2016.

[6] Yudong Cao, Anmer Daskin, Steven Frankel and Sabre Kais. Quantum circuits for solving linear systems of equations. In *arXiv:1110.2232v3* April 2013.

[7] Wikipedia, the free enciclopedia Quantum algorithm for linear systems of equations In *en.wikipedia.org/wiki/Quantum_algorithm_for _linear_systems_of_equations*.

[8] Aram W. Harrow, Avinatan Hassidim, Seth Lloyd Quantum algorithm for solving linear systems of equations In *arXiv:0811.3171* 2009.

[9] Wossnig, Leonard; Zhao, Zhikuan; Prakash, Anupam A quantum linear system algorithm for dense matrices In *arXiv:1704.06174* 2017.