A flexible implementation of the Harrow-Hassidim-Lloyd quantum algorithm for 2x2 symmetric matrices with equal diagonal entries

Juan Diego Castro Miyashiro

Department of Computing Science

University of Alberta

Abstract- In recent years, due to its potential to provide an exponential speed up in many problems, Quantum Computing has become one of the most exciting and active fields of research in Physics and Computer Sciences. Among the most relevant of the tasks being studied in this field is the one concerning linear system of equations and how to solve them using Quantum Computing techniques. In this paper we develop an implementation of the quantum Harrow-Hassidim-Lloyd linear system solver algorithm for a special type of a 2 by 2 matrix. We present the circuit and test it in the IBM Quantum Computers and QASM simulators available online showing the results for two test cases and compared them to the normalize true solution computed classically.

INTRODUCTION.

For its wide range of application in various fields the problem of solving linear systems of equations is one of the most exciting tasks in which quantum computers are expected to be useful in the future. The Harrow-Hassidim-Lloyd (HHL) algorithm formulated in 2009 promises a polylogarithmic time complexity in solving systems of equations where, for whatever reason, our final goal is to find the expectation value of some operator applied to the vector x rather than x itself. A general implementation of this algorithm for arbitrary d-dimensional matrices is still currently not possible due to efficiency limitations in embedding an arbitrary system in a Hamiltonian, a step that is required for the HHL algorithm (Child & Kothari, 2010). Previously at the University of Alberta Thea Wang (2019) elaborated an implementation of the HHL algorithm for the specific case of the 2 x 2 linear system

$$A = \begin{bmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{bmatrix};\ \mathrm{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Similarly, Cao, Daskin, Frankel and Kais (2012) implemented an example for a similar fixed system of linear equations. In this project our goal

is to go a step further and implement the quantum circuit for a more flexible case. The instance in which A is not fixed but a 2x2 symmetric matrix with equal diagonals of the form

$$A = \pi \begin{bmatrix} 2^{3-a} & 2^{3-b} \\ 2^{3-b} & 2^{3-a} \end{bmatrix}$$

Where a and b are positive integers and a is strictly less than b. We will bypass the limitations by exploiting two primary facts. The first is that as pointed out by Sadeghi (2016) the exponentials of a 2 by 2 antidiagonal matrix with equal entries have a friendly closed form and the second one being that:

$$e^{\mathrm{i}[\mathrm{X}+\mathrm{Y}]t} = e^{\mathrm{iX}t}e^{\mathrm{iY}t} \leftrightarrow [X, Y] = 0$$

1. The HHL algorithm

As explained by Dervovic et al (2018) and by Yudong Cao et al (2012) the HHL algorithm can be broken down in two key concepts: The embedding of the matrix that describe the system into a Hamiltonian and the Phase Estimation algorithm. We will walk through them in order and then we will put it all together before proceeding to explain our implementation

Embedding the system

The key realization of this step of the algorithm is that if our system's matrix A is Hermitian and has eigenvalues $|\mu_j\rangle$ and eigenvectors λ_j we can make a mapping to a unitary matrix U by matrix exponentiation that conserves the eigenvectors and eigenvalues. This is needed because Quantum gates can only perform unitary matrix transformation. Under this mapping the eigenvalues for the eigenvectors will be transform as follows:

$$\lambda_j \rightarrow e^{i\lambda_j t}$$

In the case were A does not happen to be Hermitian the only relevant change we would need to do would be to embed A in bigger system of the form.

$$\begin{bmatrix} 0 & A \\ A & 0 \end{bmatrix}$$

Which clearly is Hermitian (Harrow et al, 2009).

Phase Estimation

The phase estimation algorithm is a common subroutine in several quantum algorithms in which, given a target quantum register of length b with all its qubits in the Hadamard positive state (the $|+\rangle$ superposition state), a quantum operator U, and a register in the state of one of the eigenvector of the matrix A the algorithm performs the transformation:

$$|\mu_j\rangle|+\rangle \rightarrow |\mu_j\rangle|2^n \lambda_j\rangle$$

Where λ_j is the phase of the operator U as well as the eigenvalue of the Hermitian matrix A.

Similarly, if the input is an arbitrary quantum state b, the system would be left in a superposition of the eigenvectors of the form:

$$|\Psi\rangle = \sum_j \beta_j |2^n \lambda_j\rangle\, |\mu_j\rangle$$

Where the βs would be just the weights of the linear combination of eigenvectors that spans b

Putting the algorithm together.

Taking this into consideration HHL works by putting a quantum state $|b\rangle$ (proportional to the true vector b) in the eigenvalue register of the QPE while it uses the unitary matrix produced by the exponentiation of A as the operator.

This puts the target register in a superposition of the eigenvalues of A which corresponds to the superposition that spans b (using the eigenvectors as basis).

The algorithm then makes use of an additional ancillary qubit to perform a controlled rotation applying Ry gates conditioned on the eigenvalues to the ancilla. The Ry gate is defined by the following matrix transformation

$$R_y(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$$

The system is thus put in the following state as result of the operation:

$$|\Psi\rangle = \sum_j \left(\sqrt{1 - \left(\frac{C}{\lambda_j}\right)^2}\; |0\rangle + \frac{C}{\lambda_j}|1\rangle \right) \beta_j |2^n \lambda_j\rangle |\mu_j\rangle$$

This is accomplish by setting the angle of rotation of the rotation to

$$\theta = 2 \arcsin\left(\frac{C}{\lambda_j}\right)$$

Where C is just a proportionality constant that we chose.

Then we clean the QPE target register by applying the same gates in reverse order exploiting the fact that unitary transformations are reversible. This leaves the system as follows:

$$|\Psi\rangle = \sum_j \left(\sqrt{1 - \left(\frac{C}{\lambda_j}\right)^2}\; |0\rangle + \frac{C}{\lambda_j}|1\rangle \right) \beta_j |0\rangle \left|\mu_j\right\rangle$$

For reference a schematic of this circuit can be found in Figure 1

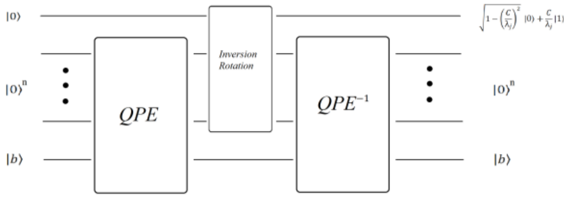

FIG 1: High level schematics of the HHL algorithm circuit

Now note that if we make measurements and post-select the states where the ancillary qubit collapses to $|1\rangle$ we are left with the state

$$|\Psi\rangle = \mathrm{C}\sum_j \frac{\beta_j}{\lambda_j}|\mu_j\rangle\,|0\rangle|1\rangle$$

And if we look carefully, we will realize that we have accomplish our goal since

$$A^{-1}|b\rangle \propto \mathrm{C}\sum_j \frac{\beta_j}{\lambda_j}\left|\mu_j\right\rangle$$

IMPLEMENTATION.

Implementing our operator

To implement the HHL algorithm for our case we rewrite our target matrix as the sum of commuting diagonal and anti-diagonal matrices.

$$\pi\begin{bmatrix}0 & 2^{3-b}\\ 2^{3-b} & 0\end{bmatrix}+\begin{bmatrix}2^{3-a} & 0\\ 0 & 2^{3-a}\end{bmatrix}) = \pi\begin{bmatrix}2^{3-a} & 2^{3-b}\\ 2^{3-b} & 2^{3-a}\end{bmatrix}$$

$$\begin{bmatrix}0 & 2^{3-b}\\ 2^{3-b} & 0\end{bmatrix}\begin{bmatrix}2^{3-a} & 0\\ 0 & 2^{3-a}\end{bmatrix} = \begin{bmatrix}2^{3-a} & 0\\ 0 & 2^{3-a}\end{bmatrix}\begin{bmatrix}0 & 2^{3-b}\\ 2^{3-b} & 0\end{bmatrix}$$

This means that if we take the matrix exponential of A, we can rewrite it as the product of the exponential of its diagonal component (X) with the exponential of antidiagonal component (Y):

$$e^{\mathrm{i}[\mathrm{X}+\mathrm{Y}]t} = e^{\mathrm{iX}t}e^{\mathrm{iY}t}$$

Now we are going to use the fact from Sadeghi (2016) that the exponential of a 2 by 2 antidiagonal matrices with repeated entries can in general be written as.

$$e^{\mathrm{i}\pi\begin{bmatrix}0 & 2^{3-b}\\ 2^{3-b} & 0\end{bmatrix}t} = \begin{bmatrix}\cosh(i\pi 2^{3-b}t) & \sinh(i\pi 2^{3-b}t)\\ \sinh(i\pi 2^{3-b}t) & \cosh(i\pi 2^{3-b}t)\end{bmatrix}$$

Which we can further simplify using the identities

$$\sinh(ix) = i\sin(x)$$

$$\cosh(ix) = \cos(x)$$

And assign

$$\theta = -\pi 2^{3-b}$$

to rewrite the matrix as:

$$e^{\mathrm{iY}t} = \begin{bmatrix}\cos(\theta t) & -\mathrm{i}\sin(\theta t)\\ -\mathrm{i}\sin(\theta t) & \cos(\theta t)\end{bmatrix}$$

Note that is no more than the Rx-gate

$$R_X(2\theta t) = \begin{bmatrix}\cos(2\theta t/2) & -\mathrm{i}\sin(2\theta t/2)\\ -\mathrm{i}\sin(2\theta t/2) & \cos(2\theta t/2)\end{bmatrix}$$

Setting t=1/4 this can be easily implemented by

$$R_X(\tfrac{\theta}{2}) = \begin{bmatrix}\cos(\theta/4) & -\mathrm{i}\sin(\theta/4)\\ -\mathrm{i}\sin(\theta/4) & \cos(\theta/4)\end{bmatrix}$$

Likewise, the exponential of the diagonal matrix X

$$e^{\mathrm{iX}t} = \begin{bmatrix}e^{\mathrm{i}\pi 2^{3-a}t} & 0\\ 0 & e^{\mathrm{i}\pi 2^{3-a}t}\end{bmatrix}$$

can also be simply implemented by a combination of X-gates and R-phi

$$\begin{bmatrix}e^{\mathrm{i}\pi 2^{3-a}t} & 0\\ 0 & e^{\mathrm{i}\pi 2^{3-a}t}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & e^{\mathrm{i}\pi 2^{3-a}t}\end{bmatrix}\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & e^{\mathrm{i}\pi 2^{3-a}t}\end{bmatrix}$$

$$= XR_{(\emptyset)}XR_{(\emptyset)}$$

Where we make

$$\emptyset = i\pi 2^{3-a}t$$

The IBM Qiskit Python package conveniently has available its own implementation for this transformation.

So, our final operator for the QPE is just the controlled version of this succession of gates.

$$\exp\left(\mathrm{i}\pi\begin{bmatrix}2^{3-a} & 2^{3-b}\\ 2^{3-b} & 2^{3-a}\end{bmatrix}t\right) = CnotCR_{(\emptyset)}CnotCR_{(\emptyset)}R_{y\left(\frac{\theta}{2}\right)}$$

Where we are going to use t=4.

Implementing the inversion rotation

In the most general case for this step we would be required to implement a unitary gate U able to take the input in the register of the eigenvalues and map it to their inverses. However, in this case we are going to make use of our knowledge of the format of the input matrix and solve it in an ad-hoc manner.

It is easy to check that the eigenvalues of the antidiagonal matrix Y are b and -b and thus the eigenvalues of its exponential correspond to $e^{\mathrm{i}2\pi 2^{-b}}$ and $e^{-\mathrm{i}2\pi 2^{-b}}$ with t=4. The diagonal matrix simply scales the input vector by the entry in its diagonal. Hence, the eigenvalues for the product of both matrices are

$$e^{\pm \mathrm{i}2\pi 2^{-b}} e^{\mathrm{i}2\pi 2^{-a}} = e^{2\pi i(2^{-a} \pm 2^{-b})}$$

In our matrix |a| < |b| so to apply the correct angle of rotation to invert each eigenvalue we only need to condition the rotation in the most significant qubit of the QPE registry.

This means that we can implement the inversion by simply applying two control rotation in the most significant qubit. If it's 1 we rotate by an angle of $\theta_1 = 2\arcsin\left(\frac{C}{2\pi(2^{-a}+2^{-b})}\right)$ and if 0 we rotate by $\theta_2 = 2\arcsin\left(\frac{C}{2\pi(2^{-a}-2^{-b})}\right)$.

Therefore, we can replace the whole rotation inversion by the circuit shown in figure 2 where the control reaches to the first most significant qubit in the QPE registry.

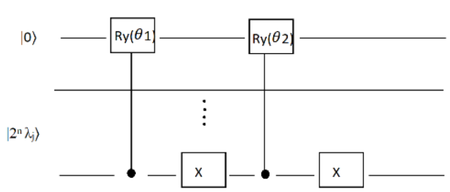

FIG 2: Circuit for the rotation inversion of the eigenvalues for our family of matrices. The rotation is condition on the first most significant qubit of the QPE registry.

TEST CASES.

We are now going to proceed to present the results we obtained for the two test cases ran in the IBM Cloud Quantum Computer and QASM simulator.

For all cases we will be using

$$C = 2\pi(2^{-a} - 2^{-b})$$

Test Case 1

$$A = \begin{bmatrix} 4\pi & 2\pi \\ 2\pi & 4\pi \end{bmatrix};\ b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

The result of the linear system solved conventionally is:

$$x = \left(\frac{1}{\pi}\right)\begin{bmatrix} 1/3 \\ -1/6 \end{bmatrix}$$

which means that we expect the square amplitude of the corresponding quantum state to be measure to:

$$\begin{bmatrix} (C_1)^2 \\ (C_2)^2 \end{bmatrix} = \begin{bmatrix} \left(\frac{1/9}{1/9+1/36}\right) \\ \left(\frac{1/36}{1/9+1/36}\right) \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}$$

So, when we read the results, we expect the states that end in 1 to have the same ratio.

The circuit generated using QISKIT as well as the results from running it in the QASM simulator can be found in Figure 3 and 4 underneath

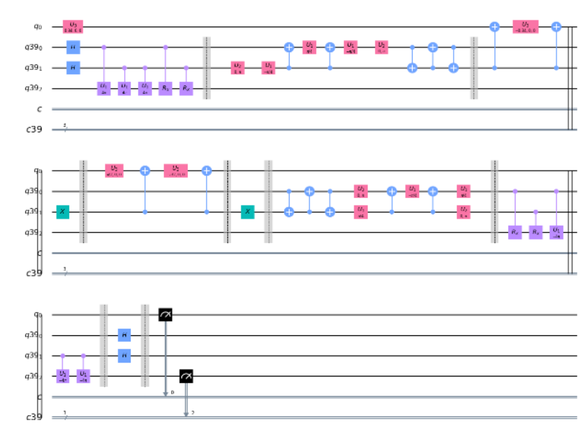

FIG 3: Quantum Circuit generated by Qiskit using the IBM subroutines for QFT and QFT^{-1}

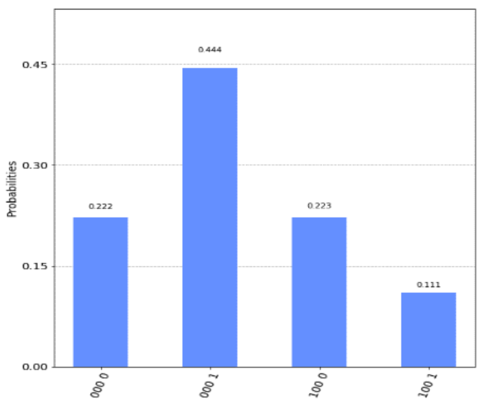

FIG 4: Histogram of results counts of the simulation of test case 1 with 8000 shots from the IBM-QASM simulator

We note from FIG 4 that

$$c_{1qasm} = \frac{0.444}{0.444 + 0.111} = 0.8$$

$$c_{2qasm} = \frac{0.111}{0.444 + 0.111} = 0.2$$

So indeed, the normalize vectors are the same. Now if we run the same circuit in the ibmq-16_melbourne quantum computer available online we get the results depicted in figure 5.

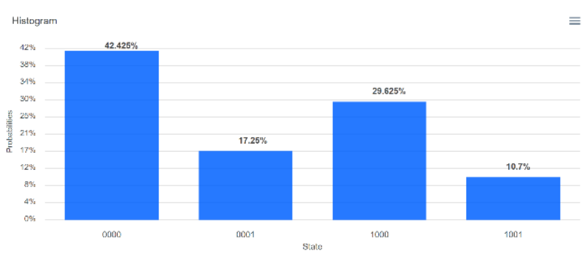

FIG 5: Histogram of counts of the simulation of test case 1 with 8000 shots from the ibmq-16_melbourne quantum computer available online

Normalizing the cases where the ancilla collapsed to 1 we get

$$c_{1melb} = \frac{29.625}{29.625 + 10.7} = 0.734$$

$$c_{2melb} = \frac{10.7}{29.625 + 10.7} = 0.265$$

Thus, the L1 norm of the difference between the amplitudes squares is

$$\overrightarrow{||c_{qasm} - c_{melb}||_1} = 0.0785 + 0.0784 = 0.1569$$

Test Case 2

$$A = \begin{bmatrix} 4\pi & 1\pi \\ 1\pi & 4\pi \end{bmatrix};\ \mathrm{b} = \begin{bmatrix} -\sin(\pi/8) \\ \cos(\pi/8) \end{bmatrix}$$

The solution to 4 significant digits is

$$\mathrm{x} = \begin{bmatrix} -0.05208 \\ 0.08654 \end{bmatrix}$$

Thus, we expect

$$\begin{bmatrix} (C_1)^2 \\ (C_2)^2 \end{bmatrix} = \begin{bmatrix} \dfrac{0.05208^2}{0.05209^2 + 0.08654^2} \\ \dfrac{0.08654^2}{0.05209^2 + 0.08654^2} \end{bmatrix} = \begin{bmatrix} 0.2658 \\ 0.7341 \end{bmatrix}$$

Likewise, we present the circuit generated using Qiskit as well as the results from running it in the QASM simulator in Figure 5 and 6.

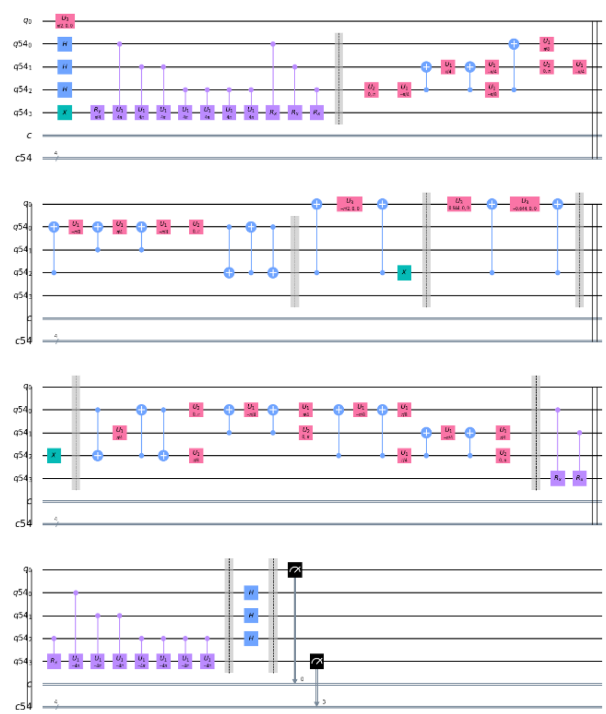

FIG 5: Quantum Circuit for test case 2 generated using Qiskit subroutines for QFT and QFT[-1]

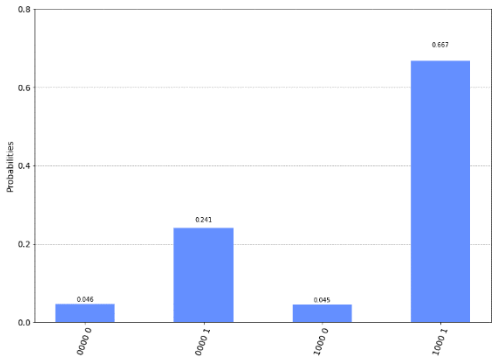

FIG 6: Histogram of results counts of the simulation of test case 1 with 8000 shots from the IBM-QASM simulator

And again, we note that:

$$c_{1qasm} = \frac{0.241}{0.241 + 0.667} = 0.2654$$

$$c_{2qasm} = \frac{0.667}{0.241 + 0.667} = 0.7345$$

The L1 difference between the two normalized vectors is

$$||\overrightarrow{c_{qasm} - c_{real}}||_1 = 0.004 + 0.004 = 0.008$$

Similarly, running the same circuit with the same number of shots in the Melbourne back-end we get the following output:

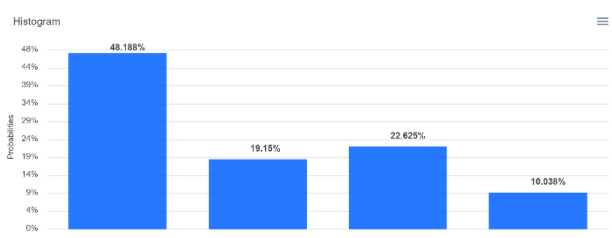

FIG 5: Histogram of results counts of the simulation of test case 1 with 8000 shots from the ibmq-16_melbourne quantum computer available online

Looking at the states that end in 1 again we see that

$$c_{1melb} = \frac{10.038}{19.150 + 10.038} = 0.3439$$

$$c_{2melb} = \frac{19.150}{19.150 + 10.038} = 0.6561$$

And the L1 norm of the difference between the amplitude square of the states gotten by the real device and the QASM is

$$||\overrightarrow{c_{qasm} - c_{melb}}||_1 = 0.0785 + 0.1087 = 0.1872$$

LIMTATIONS AND FURTHER WORK.

Even though we have accomplished our goal and have been able to extract information about x with reasonable precision using the currently noisy quantum computers. The procedure we have implemented is still no more than an example with an ad-hoc solution.

Throughout the implementation we have relied heavily in the format of our input. The prescribed format of our matrix made it possible to easily factorized it into unitary operators and readily gave up information about its eigenvalues. We relied upon this information during the rotation inversion which is the key step of HHL. It remains to be implemented a general unitary operation that takes the result of the QPE algorithm and maps directly to its inverse, so we do not need to change the angle of rotations with every input in our instance of HHL.

In addition to this every other limitation of the algorithm described by Cao et al (2012) and Childs and Kothari (2010) such as the problem of the preparation of an arbitrary state $|b\rangle$ applies as well to this instance

In conclusion even though we have been able to put together a working example of the HHL algorithm this is still a didactical implementation.

References

Cao, Y., Daskin, A., Frankel, S., & Kais, S. (2012). Quantum circuit design for solving linear systems of equations. *Molecular Physics*, 110(15-16), 1675-1680.

Childs, A.M., & Kothari, R. (2010). Limitations on the simulation of non-sparse Hamiltonians. *Quantum Inf. Comput.*, *10*, 669-684.

Dervovic, D., Herbster, M., Mountney, P., Severini, S., Usher, N., & Wossnig, L. (2018). Quantum linear systems algorithms: a primer. *ArXiv, abs/1802.08227.*

Harrow, A., Hassidim, A. & Lloyd, S. (2009). Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15), 150502.

Sadeghi, A. (2016). On the Function of Block Anti Diagonal Matrices and Its Applications. *International Journal of Mathematical Modelling & Computations*, 6 (2), 105- 117

Wang, T. (2019). Implementing Quantum Computing on Solving Linear Systems of Equations. Unpublished manuscript. Department of Computing Science University of Alberta