Implementation of the HHL algorithm

Sameen Aziz
Roll number: 2023-10-0231

April 7, 2023

1 Abstract

Quantum computers have proven their superiority over classical computers in several problems. Many quantum algorithms have been developed which significantly reduce the computation time as compared to classical algorithms. One such problem is that of solving a system of linear equations. The HHL algorithm, developed to solve a system of linear equations, provides an exponential speedup over classical methods. The HHL algorithm can only provide an approximate solution, yet it has proven to be practical in fields such as machine learning and differential equations. In the following sections, the mathematical formulation, as well as the implementation of the HHL algorithm, is discussed.

2 Introduction

Quantum computers employ the principles of quantum mechanics, such as superposition and entanglement, to perform computations much more efficiently than their classical counterparts. For some problems, such as finding the prime factors of a large integer, quantum computers have been able to provide exponential speedups over classical computers [3]. One such domain where quantum computers have proven their superiority over classical computers is that of linear algebra. Solving a system of linear equations [3], quantum verification of matrix product [2], and commutativity testing of a set of matrices [4] are some of the problems that have been quite efficiently tackled with the help of quantum computers. In this report, the algorithm developed to solve a system of linear equations, known as the HHL algorithm, will be discussed.

Linear equations are of prime importance and find their applications in several fields, such as science, engineering, economics, finance, etc. They are used to solve differential equations and partial differential equations and are also used in regression analysis. Several classical techniques have been developed to solve systems of linear equations. However, as the size of the data set grows, so does

the computation power and time needed to solve the system of linear equations corresponding to that data on a classical computer [3].

In 2009, Harrow, Hassidim, and Llyod proposed the HHL algorithm to solve a system of linear equations on a quantum computer. On a classical computer, solving a system of N linear equations in N variables takes time of order N. The HHL algorithm can cut down this time to the order $\log(N)$ in some cases. The HHL algorithm can prove to be useful when the user is not interested in the actual solution vector but is rather interested in the result of applying some operator to it. The said algorithm cannot, however, provide an exact solution to the system of linear equations since that would require time of order N as explained below, which eliminates the advantage it has over classical algorithms, and that may be considered one of its limitations [3].

Let A be an $N \times N$ matrix and let $\vec{b}$ be an N-dimensional vector. Then, the task is to find some vector $\vec{x}$ such that $A\vec{x} = \vec{b}$. Classically, we are able to find the exact elements of the solution vector $\vec{x}$; however, as mentioned above, the problem becomes quite an arduous one when N is large. With the help of the HHL algorithm, we are not able to find the vector $\vec{x}$ exactly since the algorithm stores $\vec{x}$ in the form of a quantum state $|x\rangle$; therefore, to find all the elements of $\vec{x}$ would require us to perform the experiment at least N times. However, if we want to find, say, the expectation value of some operator M, that is, we want to find $\langle x|M|x\rangle$, then that can be accomplished using the HHL algorithm. Moreover, normalization, weights in different parts of space, moments, etc., are some other functions on $\vec{x}$ that can be obtained using the HHL algorithm [3].

Furthermore, the HHL algorithm imposes a few restrictions on the system of linear equations, which might limit its applications, such as the matrix A should be Hermitian, should be a sparse matrix, meaning that most of its elements should be zero, and should have a low condition number κ, where κ is a measure of how sensitive a matrix is to perturbations [3].

Despite the seemingly slightly hopeless state of affairs, the HHL algorithm has found its applicability in machine learning, such as solving the problem of least-squares fitting efficiently [6]. The HHL algorithm was also applied to the finite element method, which finds numerical approximations to the solutions of boundary value problems for partial differential equations [5].

3 Mathematical Formulation

Let us begin by writing the equation $A\vec{x} = \vec{b}$ in the language of quantum mechanics, where A, x, and b are used as defined above.

$$A|x\rangle = |b\rangle.$$

Applying A^{-1} on both sides gives the following

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

A can be written in terms of its eigenvalues and eigenvectors as

$$A = \sum_{i=0}^{N-1} \lambda_i |u_i\rangle\langle u_i|,$$

where $|u_i\rangle$ is an eigenvector of A corresponding to the eigenvalue λ_i and $\lambda_i \in \mathbb{R}$ for all values of i. Similarly,

$$A^{-1} = \sum_{i=0}^{N-1} \lambda_i^{-1} |u_i\rangle\langle u_i|.$$

Let us say we are given the quantum state $|b\rangle = \sum_i b_i |i\rangle$. Then $|b\rangle$ can be written in the eigenbasis of A as

$$|b\rangle = \sum_{j=0}^{N-1} b_j |u_j\rangle,$$

where $b_j \in \mathbb{C}$. Then,

$$|x\rangle = \left(\sum_{i=0}^{N-1} \lambda_i^{-1} |u_i\rangle\langle u_i| \right) \sum_{j=0}^{N-1} b_j |u_j\rangle,$$

$$|x\rangle = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \lambda_i^{-1} b_j |u_i\rangle\langle u_i|u_j\rangle,$$

but $\langle u_i|u_j\rangle = \delta_{ij}$, and $\delta_{ij} = 1$ only when $i = j$. Thus, we are left with

$$|x\rangle = \sum_{i=0}^{N-1} \lambda_i^{-1} b_i |u_i\rangle.$$

which is our desired solution vector [1].

An overview of how this algorithm is implemented is given below. However, the actual implementation and a detailed explanation of the methodology have not been made a part of this report.

To implement the HHL algorithm on a quantum computer, we begin by creating two quantum registers, both of which are initialized to $|0\rangle$. One of these is used to store the solution vector $|x\rangle$ and is denoted by n_b and also, $N = 2^{n_b}$. The other one is used to store the eigenvalues of A and is denoted by n. There is also an auxiliary qubit denoted by $|0\rangle_a$, which will be measured at the end [1].

Now since A and A^{-1} are not unitary, A^{-1} cannot be directly applied to $|b\rangle$. A can, however, be made unitary in the following manner

$$U = e^{iAt}.$$

Here, U is a unitary matrix whose eigenvalues and eigenvectors are the same as those of A. Thus, we can apply quantum phase estimation on U to estimate the eigenvalues of A [3].

Let's say the initial state of the system looks like

$$|\psi\rangle_0 = |0\rangle_{n_b}|0\rangle_n|0\rangle_a,$$

Then, the state $|b\rangle$ is fed into the register labeled as n_b.

$$|\psi\rangle_1 = |b\rangle_{n_b}|0\rangle_n|0\rangle_a.$$

Then, Quantum Phase Estimation (QPE) is applied, after which the state looks like

$$|\psi\rangle_2 = \sum_{j=0}^{N-1} b_j|u_j\rangle_{n_b}|\tilde{\lambda_j}\rangle_n|0\rangle_a.$$

After this, the ancilla qubit is rotated, and we get

$$|\psi\rangle_3 = \sum_{j=0}^{N-1} b_j|u_j\rangle_{n_b}|\tilde{\lambda_j}\rangle_n \left(\sqrt{1-\frac{C^2}{\tilde{\lambda_j}^2}}|0\rangle_a + \frac{C}{\tilde{\lambda_j}}|1\rangle_a\right),$$

where C is some normalization constant. The next step is to apply inverse QPE, after which we obtain the following state

$$|\psi\rangle_4 = \sum_{j=0}^{N-1} b_j|u_j\rangle_{n_b}|0\rangle_n \left(\sqrt{1-\frac{C^2}{\tilde{\lambda_j}^2}}|0\rangle_a + \frac{C}{\tilde{\lambda_j}}|1\rangle_a\right).$$

Then, as the final step, the auxiliary qubit is measured. If the outcome is $|0\rangle_a$, the computation is repeated. If we get $|1\rangle_a$ as our measurement outcome, then the final state of the register is

$$|\psi\rangle_5 = \frac{1}{\sqrt{\sum_{j=0}^{N-1}|\frac{b_jC}{\tilde{\lambda_j}}|^2}} \sum_{j=0}^{N-1} \frac{b_jC}{\tilde{\lambda_j}}|u_j\rangle_{n_b}|0\rangle_n.$$

which corresponds to the solution vector$|x\rangle$ [1].

References

[1] Hector Jose Morrell Jr au2, Anika Zaman, and Hiu Yung Wong. Step-by-step hhl algorithm walkthrough to enhance the understanding of critical quantum computing concepts, 2023.

[2] Harry Buhrman and Robert Spalek. Quantum verification of matrix products, 2005.

[3] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15), oct 2009.

[4] Yuki Kelly Itakura. Quantum algorithm for commutativity testing of a matrix set, 2005.

[5] Ashley Montanaro and Sam Pallister. Quantum algorithms and the finite element method. *Physical Review A*, 93(3), mar 2016.

[6] Nathan Wiebe, Daniel Braun, and Seth Lloyd. Quantum algorithm for data fitting. *Phys. Rev. Lett.*, 109:050505, Aug 2012.