Quantum Algorithm for Linear Systems of Equations

Alfred Nguyen
Department of Computer Science
Technical University of Munich
05748 Garching, Bavaria
alfred.nguyen@outlook.com

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Ease of Use

A. Maintaining the Integrity of the Specifications

The IEEEtran class file is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

II. Prepare Your Paper Before Styling

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not number text heads—LaTeX will do that for you.

A. Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

B. Units

- Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as "3.5-inch disk drive".
- Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
- Do not mix complete spellings and abbreviations of units: "Wb/m^2" or "webers per square meter", not "webers/m^2". Spell out units when they appear in text: ". . . a few henries", not ". . . a few H".
- Use a zero before decimal points: "0.25", not ".25". Use "cm^3", not "cc".)

C. Equations

Number equations consecutively. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

$$a + b = \gamma \tag{1}$$

Be sure that the symbols in your equation have been defined before or immediately following

D. LaTeX-Specific Advice

Please use "soft" (e.g., `\eqref{Eq}`) cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please don't use the `{eqnarray}` equation environment. Use `{align}` or `{IEEEeqnarray}` instead. The `{eqnarray}` environment leaves unsightly spaces around relation symbols.

III. Template

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BibTeX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BibTeX to produce a bibliography you must send the .bib files.

IV. The HHL Algorithm

The next section will give a more detailed walkthrough of the HHL algorithm. Thereby, it will go through all the 5 phases namely, state preperaration, quantum phase estimation, inversion of eigenvalues, inverse quantum phase estimation and lastly the measurement.

A. State Preparation

In total we have $n_b + n + 1$ qubits. In the beginning, they are all initialized in their zero state as

$$\begin{aligned} |\Psi_0\rangle &= |0\dots0\rangle_b \ |0\dots0\rangle_c \ |0\rangle_a \\ &= |0\rangle_b^{\otimes n_b} \ |0\rangle_c^{\otimes b} \ |0\rangle_a \end{aligned} \tag{2}$$

We now have to load the vector $\vec{b}$ into the b-register. This is achieved by amplitude encoding

$$\vec{b} = \begin{pmatrix} b_0 \\ \vdots \\ b_{N-1} \end{pmatrix} \Leftrightarrow |b\rangle = \sum_{i=0}^{N-1} b_i \, |i\rangle \tag{3}$$

Thus, $|b\rangle$ is loaded into the b-register as such

$$|\Psi_1\rangle = |b\rangle_b \ |0\dots0\rangle_c \ |0\rangle_a \tag{4}$$

We have successfully encoded the $\vec{b}$ into our b-register. We now continue with the QPE.

B. Quantum Phase Estimation

We will only briefly go through the specifics of the QPE and will not discuss each step in detail, as this is not the main topic of this paper. For further explanations refer to this paper. Todo

Inser paper here

As already mentioned, the QPE is a procedure to evalute an estimate of eigenvalues. It consists of three phases namely, creating superpositions of the clock-bits via Hadamard gates, controlled rotation via the unitary U and the IQFT. After QPE we will have an estimate of the eigenvalues of the unitary U. As we have encode A as as a Hamiltonian $U = e^{iAt}$, the phase of the eigenvalue of U is proportional to the eigenvalue of A. We have to define a scaled version of our eigenvalues λ_j.

$$\widetilde{\lambda_j} = \frac{N\lambda_j t}{2\pi} \tag{5}$$

where t can be choosen freely so that the scaled eigenvalues $\widetilde{\lambda_j}$ are integers.

Thus, the eigenvalues of A will be stored in the c-register after QPE as

$$\begin{aligned} |\Psi_2\rangle &= |b\rangle_b \, |\widetilde{\lambda}\rangle_c \, |0\rangle_a \\ &= \sum_{j=0}^{N-1} b_j \, |u_j\rangle_b \, |\widetilde{\lambda_j}\rangle_c \, |0\rangle_a \end{aligned} \tag{6}$$

Notice, that the b-register is in the same form as we have discussed in the previous section. The b-register represents the $|b\rangle$ state in the eigenbasis $|u_j\rangle$ of A like this

$$|b\rangle = \sum_{j=0}^{N-1} b_j \, |u_j\rangle \tag{7}$$

Now that we have achieved the eigenvalues and have represented our $|b\rangle$ in the eigenbasis of A, we invert the eigenvalues u_i.

C. Inversion of the eigenvalues

In the next step we invert the eigenvalues. This is achieved by the rotation of the ancilla qubit in the a-register by the eigenvalues in the c-register. The state of the registers after the rotation looks like this

$$|\Psi_3\rangle = \sum_{j=0}^{N-1} b_j \, |u_j\rangle_b \, |\widetilde{\lambda_j}\rangle_c \left(\sqrt{1 - \frac{C^2}{\widetilde{\lambda_j^2}}} \, |0\rangle_a + \frac{C}{\widetilde{\lambda_j}} \, |1\rangle_a \right) \tag{8}$$

where C is a constant that should be chosen to be as large as possible to increase the success probability. Currently, the state of the a-register can either collapse into $|1\rangle$ or $|0\rangle$. The probability of the a-register collapsing to $|1\rangle$ is $\left| \frac{C}{\widetilde{\lambda_j}} \right|^2$. If the ancilla qubit collapses to $|0\rangle$, the whole procedure has to be repeated from the beginning. As mentioned earlier this has to be done, because the rotation process is not unitary and has a probability to fail.

We assume that our rotation process was successfull and the ancilla bit collapses to $|1\rangle$. The registers will look as such

$$|\Psi_4\rangle = \frac{1}{\sqrt{\sum_{j=0}^{N-1} \left| \frac{b_j C}{\widetilde{\lambda_j}} \right|^2}} \sum_{j=0}^{N-1} b_j \, |u_j\rangle_b \, |\widetilde{\lambda_j}\rangle_c \, \frac{C}{\widetilde{\lambda_j}} \, |1\rangle_a \tag{9}$$

Note that term in front of the sum is just a factor to normalize the state. Lets call this normalization factor D. We see that we now have a term $\frac{C}{\widetilde{\lambda_j}}$ that represents the inverted eigenvalues. This term can be moved freely as it is just a scalar, such that the scalar is applied to the b-register

$$|\Psi_4\rangle = D \sum_{j=0}^{N-1} \frac{C}{\widetilde{\lambda_j}} b_j \, |u_j\rangle_b \, |\widetilde{\lambda_j}\rangle_c \, |1\rangle_a \tag{10}$$

If we go back to our idea from the mathematical overview section, our b-register is in the same form as our solution state

$$|x\rangle = \sum_{i=0}^{N-1} \lambda_i^{-1} b_j \, |u_j\rangle \tag{11}$$

That means our solution is already encoded in our registers. The problem here is, that we cannot read the solution out yet. This has to with the states in the b-register and c-register being entangled with each other. That means, we cannot convert the b-register into a $|0\rangle / |1\rangle$ measurement. In the following, we have to undo all operations to unentangle the state in the b-register and c-register, to achieve the correct result.

D. Inverse Quantum Phase estimation.

The unentangling of the registers is achieved throught the IQPE which backtracks all calculations of the QPE. We are left with the following state

$$\begin{aligned} |\Psi_5\rangle &= \frac{1}{\sqrt{\sum_{j=0}^{N-1}\left|\frac{b_j C}{\lambda_j}\right|^2}} \sum_{j=0}^{N-1} \frac{C}{\lambda_j} b_j |u_j\rangle_b |0\rangle_c^{\otimes n} |1\rangle_a \\ &= \frac{C}{\sqrt{\sum_{j=0}^{2^{n_b}-1}\left|\frac{b_j C}{\lambda_j}\right|^2}} |x\rangle_b |0\rangle_c^{\otimes n} |1\rangle_a \end{aligned} \tag{12}$$

The a-register is untouched and still holds the $|1\rangle$ state as before. The c-register however is reset to the zero state $|0\rangle_c^{\otimes n}$ as in the beginning of the process at $|\Psi_0\rangle$. It is now unentangled from the b-register. The b-register now contains the solution $|x\rangle$ after the uncomputation and can be measuremed correctly.

We can furthermore simplify the term as we assume that C is real and that the eigenvectors $|u_i\rangle$, and the $|b\rangle$ state are normalized. Then we can simplify to

$$\begin{aligned} |\Psi_5\rangle &= \frac{1}{\sqrt{\sum_{j=0}^{N-1}\left|\frac{b_j}{\lambda_j}\right|^2}} |x\rangle_b |0\rangle_c^{\otimes n} |1\rangle_a \\ &= |x\rangle_b |0\rangle_c^{\otimes n} |1\rangle_a \end{aligned} \tag{13}$$

We can now read out the result $|x\rangle$ in the b-register.

E. Measurement

As mentioned earlier, we cannot obtain the whole solution for the $\vec{x}$ as reading out all the entries would cost us $\mathcal{O}(N)$ steps. This would ommit our speedup of $\mathcal{O}(logN)$. That means, by measuring we will only obtain an estimate of specific features of $|x\rangle$. Using a linear operator M we can perform various measurements on $|x\rangle$ by calculating the inner product as such

$$\langle x| M |x\rangle \tag{14}$$

With this we can extract various statistical features of $|x\rangle$ like the norm of the vector, the average of the weight of the components, moments, probability distributions, localization and concentration in specific regions, etc.

V. Evaluation

In this section, we evaluate the performance of the HHL algorithm and will shortly look at resource requirements.

The most common known method to solve a system of linear equations is Gaussian Elimination. The time complexity of the Gaussian is $\mathcal{O}(N^3)$, which is drastically slower than other classical methods. As we are only interested in an estimate of an expectation value $|x\rangle M |x\rangle$, we can take a look of similar classical methods, considering other constraints.

In the following we will compare the HHL algorithm, to the classical conjugate gradient descent algorithm, focusing on time complexity and other factors that impact its efficiency.

A. Conjugate Gradient Descent

The classical conjugate gradient descent algorithm is very common method for solving linear systems of equations. It uses the conjugate direction method, which can locate the minimum of a function. By iteratively looking for solution vectors, the procedure converges towareds the solution of the linear system. This will provide us a suitable benchmark for analyzing the efficiency of the HHL algorihtm. Not only is it one of the fastest classical algorithms, it has very similar constraints set and calculates similar results (eg. $\vec{x}^\dagger M \vec{x}$).

B. Comparison of Time Complexity

TABLE I
Time complexity comparison

Conjugate Gradien Descent	HHL Algorithm
$\mathcal{O}(\kappa s \log\left(\frac{1}{\epsilon}\right) N)$	$\mathcal{O}\left(\frac{\kappa^2 s^2}{\epsilon} \log N\right)$
$\Rightarrow \mathcal{O}(N)$	$\Rightarrow \mathcal{O}(log(N))$

As shown in the table, the HHL algorithm offers an exponential improvement compared to the conjugate gradient descent method. Conjugate gradient descent. Conjugate gradient descrent achieves a time complexity of $\mathcal{O}(N)$, whereas the HHL algorihtm runs in $\mathcal{O}(log(N))$.

Now we will take a more detailed look a the other factors involved in the timecomplexity, where:

- s is the s-sparsness
- κ is the condition number
- ϵ is the accuracy

The sparsness s, describes to number of non-zero entries per row (e.g. a 2-sparse matrix only contains 2 non-zero entries per row). The condition number κ, describes how sensitiv the output of a function is on the error of the input. That means, a function is well conditioned if the output of a function does not change a lot for bigger errors in the input. These two factors are related to the QPE phase as, a unitary e^{iAt} has to be generated. This step heavily relies on the sparsity s and conditioning κ of the matrix A to be performed efficiently. If not, this process would grow $\mathcal{O}(N^c)$ for some constant c. Again, this would ommit our speedup of $\mathcal{O}(log(N))$.

Lastly, ϵ describes the accuracy of our desired solution. We can observer, that the sparsity s and condition number κ are quadratic in the HHL algorihtm, whereas in the conjugate gradient descent algorithm, both factors operate linearly. Furthermore, the accuracy ϵ is exponentially worse in the HHL algorithm. Thus, in terms of sparsity, accuracy and and condition number, the HHL algorithm a worse runtime over the conjugate gradient descent algorithm. If these prefactors where not worse than in the classical solution, this would have bizarre implications. One can show that if the dependency is better or equal than in the classical solution, one could solve *NP-Complete* problems in exponential faster times on quatum computers, which seems unlikely. All in all, these constraints are important to consider when choosing applications, as they have great effect on the runtime.

C. Resource requirements

The resource requirements to implement the HHL algorithm are substantial. Looking at the structure of the algorithm is has very similar structure to the Shor's algorithm. We can use Shor's procedure as a lower bound to estimate the resource requirements. Shor's algorithm for 2048 RSA, needs roughly 4000 logical qubits to work properly. To compensate for error correction and other quantum effects in the circuit, this would take millions of qubits. Although, these numbers have been dropped many times over the past couple of years.

VI. Outlook

In this section we will take a look at the future outlooks of the HHL algorithm We will explore potential applications given the constraints set of the HHL algorithm. Then we will delve into different variations and optimizations which can improve the performance. Lastly, we will discuss some broad perspective, that the HHL algorithm offers in the field of quantum computing.

A. Applications

The main problem wih the HHL algorithm is, that it does not output the full solution vector, but rather the solutino state $|x\rangle$. Not to be forgotten, is that the matrix A has to be sparse for the HHL algorithm to work efficiently. But in some cases these constraints are not a problem and can therefore be solved by the HHL algorithm. We will now discuss some examples that utilize the algorithm.

1) Analyzing Large Sparse Electrical Networks: The HHL algorithm can be applied to analyze large sparse electrical networks, where we are interested in an estimate of a specific feature. Electrical networks consist of vast numbers of interconnected components, like generators, transformers, transmission lines, etc. Though, these networks have a high number of components, they have a relative low amount of connections between the each other. Thus, systems can be modeld as a graph which correspond to large sparse matrices. By being able to calculate the inverse of a matrix A^{-1} efficienly, we can deduct many network properties, such as resistance. This can help us to further optimize the electrical powersystem. Conventionally, we would have to rely on iterative algorithms, which are as we saw earlier, exponentially slower in this specific case.

2) Machine Learning (Least-Square Estimation): In the field of machine learning the HHL algorithm can also help improve calculations. The HHL algorithm also plays a significant role in the field of machine learning, as it can improve computational intensive calculations. For example in least-square estimation, the HHL algorithm can be used to estimates of inverse matrices. This can be very helpful in data fitting tasks, enabling efficient parameter estimation and model optimization.

All in all, finding more applications for the HHL algorithm is crucial, given its specific constraints and promising capabilities.

B. Variations and Optimizations

Current research and developments have been able to find variations and optimizations to the HHL algorithm. These changes have brought more efficient ways to perform the HHL algorithm and expand its applicability. We will now discuss some examples of improvements done to the HHl algorithm.

1) Ancilla Bit Requirement: Certain variants allow the HHL algorithm to operate without an ancilla bit. This eliminates the probability of failure in the rotation step of the eigenvalues, which leads to a faster walkthrough, as one must not repeat the process mutliple times at failure. Additionally, this increases the reliability of the algorithm.

2) QRAM: Quantum Random Access Memory (QRAM) is the quantum computer equivalent of the classical ram in a computer. It enables the the quantum computer to directly access quantum states for computation without having to encode the input. This would provide us with a very efficient way to load our state $|b\rangle$ into our quantum circuit.

Exploring these variations and identifying novel optimizations can contribute to the advancement and practical implementation of the HHL algorithm.

C. IT Security

Although, the HHL algorithm is primarily designed for solving linear system, there are potential applications in the field of IT security. We will now discuss some examples how one could utilize the algorithm IT security.

1) Solving Large-scale Linear Systems: The HHL algorithm can contribute to the efficient calculation of large-scale linear systems. These are required in many IT security protocols, such as secure multi-party computation or zero-knowledge proofs.

Multi-party computation is as protocol that enables multiple different machines to computer a function, whilst not revealing their data to each other. The protocol achieves this by using linear systems as a subroutine to achieve secrecy.

Zero knowledge proofs are methods that enable a party to demonstrate certain features to another party without revealing any other information except the validity of that feature. This can be usefull to verify your identity without giving out any personal data. Oftentimes these zero knowledge proofs can be modeled in linear systems, which then can be processed by the HHL algorithm.

2) Pattern recognition for fraud dection: Pattern recognition for fraud detection is also a possible application for the HHL algorithm. Again, this plays in the field of machine learning. Here various big data analysis methods are used to detect fraudulant activites, which can be accelerated by the HHL algorithm.

D. Perspectives

The HHL algorithm has made significant contributions, especially in the field of quantum machine learning. Though, there are no ground breaking applications so far, like the Shor's algorithm that has the ability to break the RSA encryption. Nonetheless, the ongoing discussions and research

on the algorithm enable us to discover new optimizations and applications. The advances in the quantum algorithms, including the HHL algorithm, have laid the foundation for future developments in the field. But because of its generality of the HHL algorithm, it has the potential to be used as a subroutine in various domains.

While experiments have demonstrated that the HHL algorithm works on a small quantum computers with high fidelity, more advancements have to be done in the hardware. Better error correction and scaling is need to unlock the full potential of quantum computing

***Abstract*—Linear systems are a fundamental problem in math or in subroutines in more complex tasks. Linear systems are in the form of $A\vec{x} = \vec{b}$, where A is a given matrix, $\vec{b}$ is a given vector and $\vec{x}$ is the unknown we to be solved. The HHL (Harrow, Hassidim, and Lloyd) algorithm is a quantum algorithm, that is able to solve these linear systems of equations exponentially faster than its classical counter part. Though, there are a few caveats to consider. We assume that we only want to solve for an expectation value of some operator on $\vec{x}$, e.g. $\vec{x}^{\dagger} M \vec{x}$ for some matrix M. That means we are not interested in the whole solution of $\vec{x}$. Also, we assume that the matrix A is sparse and is in the size of $N \times N$. Given these requirements, classical algorithms can solve this problem in $\mathcal{O}(N)$, whereas the HHL algorithm can solve this problem in $\mathcal{O}(log(N))$. This gives us an exponential speedup over the classical method.**

***Index Terms*—Harrow-Hassidim-Lloyd (HHL) quantum algorithm, Quantum Fourier transform (QFT), Inverse Quantum Fourier transform (IQFT), Quantum Phase Estimation (QPE), is that everything?**

VII. Introduction

Quantum computing has a lot of potential in many various domains.

It leverages the power of quantum superposition and entanglement to perform computations that a classical compuer cannot similate. In some cases, quantum computer algorithms can provide an exponential speed up to common classical methods. The most famous case being the Shor's algorithm for factoring large numbers, possibly cracking our current RSA encryption.

This paper will focus on an algorithm, which can estimate features of the solution vector of a linear system of equations. The HHL algorithm, designed by Aram Harrow, Avinatan Hassidim and Seth Lloyd, was introduced in 2009. It is one of the groundbreaking algorithms in quantum computing alongside Shor's factoring algorithm, Grover's search algorithm, and the quantum fourier transform (QFT). It is expected to provide exponential speed up over the commonly used classical methods. Given an $N \times N$ sparse matrix A, the HHL algorithm runs in $\mathcal{O}(log(N))$, whilst the classical counterpart runs in $\mathcal{O}(N)$. This promises us an exponential speed up over the classical method.

Among these problems, solving linear systems of equations plays a fundamental role in various scientific, engineering, and mathematical disciplines. Traditional classical methods for solving linear systems often face scalability issues, as the computational resources required grow exponentially with the problem size. However, the advent of quantum algorithms, such as the HHL (Harrow, Hassidim, and Lloyd) algorithm, offers the potential for exponential speedup in solving linear systems on a quantum computer.

The HHL algorithm, introduced in 2009 by Aram Harrow, Avinatan Hassidim, and Seth Lloyd, is a groundbreaking quantum algorithm that aims to address the challenge of efficiently solving linear systems. By harnessing the principles of quantum mechanics, the HHL algorithm takes advantage of quantum parallelism and interference to provide a potentially significant improvement over classical approaches.method.

At its core, the HHL algorithm employs quantum phase estimation to estimate the eigenvalues of the matrix involved in the linear system. By encoding the problem as a quantum state and applying quantum operations, the algorithm can extract the desired solution efficiently. This approach offers the potential for a quantum advantage by providing exponential speedup compared to classical methods.

While the HHL algorithm demonstrates the potential of quantum computing for solving linear systems, its practical implementation poses challenges. Quantum technologies face inherent noise and errors due to factors such as decoherence and imperfect gate operations. Additionally, the HHL algorithm requires high precision and control over quantum operations, making it particularly sensitive to these errors. As a result, the algorithm is currently more suited for small-scale problem instances and specialized applications.

Nonetheless, the HHL algorithm has generated significant interest and research efforts in the quantum computing community. Its potential impact spans various fields, including optimization, machine learning, cryptography, and simulation. As quantum technologies continue to advance and overcome the current limitations, the HHL algorithm holds the promise of revolutionizing the field of linear systems solving and contributing to the broader quest for harnessing the power of quantum computing to tackle complex computational problems.