Ukrainian Catholic University

Faculty of Applied Sciences

Business Analytics & Computer Science Programmes

HHL Algorithm for Solving Linear System of Equations

Linear Algebra second interim project report

Authors:
Valihurskyi Anton
Shtohryn Oleh
Tarnavskyi Maksym-Vasyl

Mentor:
Hryniv Rostyslav

Abstract

In this report, we are going to present how quantum computing can be applied to linear algebra, using the relatively fundamental Harrow–Hassidim–Lloyd algorithm as an example. We will provide an overview of the algorithm, explain its uses and limitations, as well as try to analyze how useful it actually is.

1 Introduction

In this report we describe theoretical background for Harrow-Hassidim-Lloyd quantum algorithm for solving linear system of equations. We supply it with numerical examples and bloch-sphere system representation, to improve reader's understanding. In the very end, we elaborate on algorithm's complexity and show results of present the algorithm in quantum simulator.

2 Motivation

Quantum computing is a relatively new field that is only just starting to gain momentum because of the current technical limitations and its relative complexity. It introduces a unique conceptual framework for problem-solving, which can make it somewhat unfriendly to beginners, especially those that have only interacted with classical algorithmic paradigms before. Nonetheless, it has significant potential and, in our opinion, is worth giving a chance, which is the reason we chose this project. Therefore, our main goal is to understand and explain the core principles of quantum computing with the chosen algorithm as an example. We also aim to assess whether this field is really worth investing your time and resources in.

But why are quantum computers good?

2.1 How it works?

1. **Superposition:** Unlike classical bits, which can only be 0 or 1, qubits can exist in multiple states simultaneously. This allows a quantum computer to process an exponential number of possible states in parallel.

2. **Quantum Entanglement**: through entanglement, the state of one qubit becomes directly related to the state of another, enabling computations where the relationships between data are crucial, such as in quantum algorithms for factorization or search.

2.2 Quantum computers supremacy

Thanks to these quantum phenomena, quantum computers can solve specific tasks—like factoring large numbers or searching unstructured databases—much faster than classical algorithms. For example, Shor's algorithm offers exponential speed-up for factorization, while Grover's algorithm provides quadratic speed-up for search tasks.

2.3 Why do quantum algorithms may not be effective on classical computers?

Classical computers operate with bits that are either 0 or 1 at any given time. Attempting to simulate quantum effects (such as superposition, entanglement, and interference) on a classical architecture requires computing all possible quantum states individually, an exponentially growing challenge with each additional qubit. This makes such simulations impractical for problems involving a large number of qubits, thus negating the inherent advantage of quantum algorithms.

2.4 HHL

Harrow–Hassidim–Lloyd algorithm (hereafter referred to as HHL) is a quantum computing tool used to solve linear systems of equations (to some extent, details will be provided later). While the specific explanation is laid out in the respective section, we would like to start with some basic quantum computing concepts to establish a common foundation.

3 Basic principles

1. **Qubits**. In the conventional computing, the basic information unit is the bit, which takes the value of either 0 or 1. On the contrary, the main feature of quantum computing and the source of both its pros and cons is its usage of qubits - they, in a way, can encode both 0 and 1, to different extents, simultaneously. This, of course, has some limitations, which makes working with them somewhat more restrictive. More specifically, a qubit can be represented mathematically as

$$\psi = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \alpha \left|0\right\rangle + \beta \left|1\right\rangle$$

   Here $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ represent basic states "0" and "1", with $|0\rangle$ and $|1\rangle$ being a specific (Dirac) notation for them, it will be explained shortly. α and β represent the probabilites of the corresponding states being measured - more specifically, the probability of measuring "0" is α^2. Therefore, the squares of α and β should sum up to 1. Also, as mentioned earlier, when we measure a qubit, it falls back into one of its basic states - this is called a **collapse**.

2. **Dirac notation**. Dirac notation, also called bra-ket notation, is the conventional way to represent quantum states. Without delving into specifics, we represent quantum state and their corresponding vector v as the ket form $|v\rangle$, and the bra form $\langle v|$ is a conjugate transpose of that. In this way, the expression $\langle u|v\rangle$ amounts to an inner product between u and v.

3. **State of a quantum system and entanglement**. Let $|\xi_i\rangle = \alpha_i |0\rangle + \beta_i |1\rangle$ denote state of i'th qubit in the system. Then the state of quantum system formed by qubits $|\xi\rangle_1, \ldots, |\xi\rangle_n$ can be found as $|\xi\rangle = |\xi_1\rangle \otimes \cdots \otimes |\xi_n\rangle$, where $\otimes$ is tensor product[6]. In fact, any normed vector can be efficiently represented in quantum computer[7]. Now consider state $|\psi\rangle = \alpha |00\rangle + \beta |11\rangle$. You might notice that by measuring the first qubit as 1 we can be sure that the second one is also 1, and the same can be said about zeros. Therefore, even if we measure only one qubit of this system,

the other will collapse as well. This connection between qubits is called quantum entanglement, and dealing with it takes up a sizable portion of the algorithm.

4. **Relative phase**. Another important concept is the relative phase of the qubit, represented by $e^{i\theta}$, where θ is an angle. To illustrate its usefulness, let's compare qubits:

$$\psi_1 = \alpha\,|0\rangle + \beta\,|1\rangle = \alpha\,|0\rangle + e^{2\pi i}\beta\,|1\rangle$$
$$\psi_2 = \alpha\,|0\rangle - \beta\,|1\rangle = \alpha\,|0\rangle + e^{\pi i}\beta\,|1\rangle$$

While the absolute values of their corresponding probabilities are the same, they still differ in the multiplier of the $|1\rangle$ state. This exactly is the relative phase, and if we can influence it (indeed, the quantum computing allows us to), it gives us another degree of freedom and more space to encode information per qubit, which is one of its main advantages over the classical bits. We also present the Bloch sphere, which is a graphical way to represent the qubit: the probabilities of $|0\rangle$ and $|1\rangle$ move it along the Z axis, while its projection onto the XY plane circle shows the relative phase of the qubit.

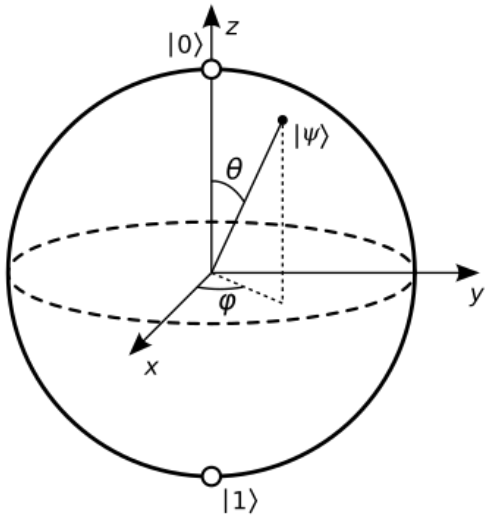


Figure: Bloch sphere representation of a qubit. [3]

Registers. We will also use the notion of a quantum register, which is basically an ordered collection of several qubits of a fixed length. For example, if the b register has all qubits in $|0\rangle$ state with probability 1, we can write it as $|0...0\rangle_b$.

Gates. Gates in quantum computing represent transformations we perform over qubits. In the linear algebra framework, they can be represented as matrices. For example, applying X gate to any qubit is tantamount to multiplying it by matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, which reverses the probabilities of 0 and 1 being measured.

4 A short review of related work and possible approaches to solutions

4.1 Literature Review

- Quantum linear systems algorithms: a primer (HHL part, pages 28-41) [9] explanation of how HHL work in terms of linear algebra;
- Quantum algorithms for solving linear systems of equations (pages 3-4)[5] brief overview of main approach to Hamiltonian simulation;
- HHL walkthrough [12] detailed algorithm walkthrough;
- Quantum curcuits for solving linear systems of equations [11] explanation on how to translate HHL to quantum computers;

4.2 What have been done

- Translated the HHL algorithm's principles of operation from the world of quantum computing to linear algebra;
- Implemented the HHL algorithm and tested it;

Our approach used the Python library `Qiskit`. Qiskit, IBM's open-source quantum computing framework, provides tools for implementing and optimizing the HHL algorithm. Key features include:

- Classes like `QuantumCircuit` and `QuantumRegister`;
- Quantum gates implemented as functions acting on the quantum register;
- Direct compatibility with IBM's quantum computers (free access via first-come, first-served basis).

The most challenging task was translating HHL's quantum principles into linear algebra. This report was greatly inspired by HHL walkthrough [12], which significantly improved our understanding.

4.3 Bonus Steps

- Launch the implementation on a real quantum computer (e.g., IBM's free quantum hardware);
- Fuse HHL with classical algorithms to improve performance;

5 Theoretical foundations

5.1 Harrow-Hassidim-Lloyd Algorithm overview

For the Linear System of equations $A\mathbf{x} = \mathbf{b}$, where A is $n \times n$ hermitian and non-singular, we want to find solution of form $\mathbf{x} = A^{-1}\mathbf{b}$. If A is not Hermitian we may consider

$\begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ 0 \end{bmatrix}$. Because A is Hermitian we can find its spectral decomposition

$$A = \sum_{j=1}^{n} \lambda_j \mathbf{u}_j \mathbf{u}_j^\top \tag{1}$$

where λ_j - is an eigenvalue (EV) of A and $\mathbf{u}_j$ respective eigenvector (EVc). Because matrix A is invertible and symmetric so its EVc's form an orthogonal basis, thus we can rewrite $\mathbf{b}$ as

$$\mathbf{b} = \sum_{j=1}^{n} \beta_j \mathbf{u}_j \tag{2}$$

Then we can easily find solution

$$\mathbf{x} = A^{-1}\mathbf{b} = \sum_{j=1}^{n} \frac{\beta_j}{\lambda_j} \mathbf{u}_j \tag{3}$$

HHL algorithm basically estimates eigenvalues and calculates solution vector in the following steps.

1. Encode A as unitary $U = e^{iAt}$, where t - time evolution [4]
2. Initialize quantum registers b, c, a for vector $\mathbf{b}$, control bits (where the EV's will be stored) and ancilla bit, that controls "correctness" of the solution
3. Normalize $\mathbf{b}$, so it can be represented as state of quantum system
4. Apply Hadamard gate $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ to each qubit in c-register
5. Apply controlled U rotation on b-register, where the controlling qubits are in c-register, starting from the most significant one to the least significant. It means to multiply $U^{2^r}\mathbf{b}$ if qubit c_r is read as 1
6. Apply Discrete Fourier Transform to c-register
7. Apply $R_y(\theta) = \begin{bmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix}$ on ancila-bit and if its state is measured as $\begin{bmatrix} 1 & 0 \end{bmatrix}^\top$ return to step 1. Otherwise proceed with an algorithm
8. Apply Inverse Discrete Fourier Transform on c-register
9. Apply inverted controlled U^{-1} rotation on b-register
10. Apply Hadamard gates on c-register

After that we will have our solution $\mathbf{x}$ in the c-register. One might notice, that we have normalized $\mathbf{b}$. In fact, with HHL, we solve not the former problem, but $A\hat{\mathbf{x}} = \frac{\mathbf{b}}{\|b\|}$, where $\hat{\mathbf{x}}$ stores relation between x_i (original solution), later we will explain why it is relevant to our former problem. We also had to calculate e^{iAt} for which we would need to know eigenvalues of A, however, we assume that U is found through Hamiltonian simulation [5] (section 5). In general, problem of simulating matrix as Hamiltonian for general case is unsolved, even though deeply studied. We add additional constraints that matrix A is sparse, and it values are near 1.

5.2 On Relevance of the Modified Problem

In fact, any normed vector can be efficiently represented in quantum computer[7].

Now, we can state, that solving "normed version" is relevant.

Proposition 1. *Solving QLSP (Quantum LSP) for vector* $|b\rangle$ *that encodes* $\mathbf{b}$ *results in a vector* $|\mathbf{x}\rangle$ *that encodes* $\mathbf{x}$*.*

Proof. We want to show that encoded version of $\mathbf{x} = A^{-1}\mathbf{b}$ as

$$|\mathbf{x}\rangle = |A^{-1}\mathbf{b}\rangle \propto |\mathbf{x}\rangle = A^{-1}|\mathbf{b}\rangle \tag{4}$$

$$|\mathbf{b}\rangle = \frac{\sum_{j=1}^{n} \beta_j |\mathbf{u_j}\rangle}{\|\mathbf{b}\|}, |\mathbf{x}\rangle = \frac{\sum_{j=1}^{n} \tilde{x}_j |\mathbf{u_j}\rangle}{\|\mathbf{x}\|} \tag{5}$$

We can write

$$A^{-1}|\mathbf{b}\rangle = \frac{1}{\|\mathbf{b}\|}\sum_{j=1}^{n} \lambda_j^{-1} |\mathbf{u_j}\rangle \langle \mathbf{u_j}| \sum_{k=1}^{n} \beta_k |\mathbf{u_k}\rangle = \frac{1}{\|\mathbf{b}\|}\sum_{j=1}^{n} \frac{\beta_j}{\lambda_j} |\mathbf{u_j}\rangle \tag{6}$$

From this, we have

$$\frac{A^{-1}|\mathbf{b}\rangle}{\|A^{-1}|\mathbf{b}\rangle\|} = \frac{1}{\frac{1}{\|\mathbf{b}\|}\sqrt{\sum_{j=1}^{n} \frac{|\beta_j|^2}{|\lambda_j|^2}}} \frac{1}{\|\mathbf{b}\|}\sum_{j=1}^{n} \frac{\beta_j}{\lambda_j} |\mathbf{u_j}\rangle = \frac{1}{\sqrt{\sum_{j=1}^{n} \frac{|\beta_j|^2}{|\lambda_j|^2}}} \sum_{j=1}^{n} \frac{\beta_j}{\lambda_j} |\mathbf{u_j}\rangle \tag{7}$$

From 6 follows that

$$\langle \mathbf{u_j}| A^{-1}|\mathbf{b}\rangle = \frac{1}{\|\mathbf{b}\|}\frac{\beta_j}{\lambda_j} \tag{8}$$

$$\langle \mathbf{u_j}|\mathbf{x}\rangle = \frac{1}{\|\mathbf{x}\|}\tilde{x}_j = \frac{1}{\|\mathbf{x}\|}(\mathbf{u}_j^\top A^{-1}\mathbf{b}) \tag{9}$$

Equation (9) holds, because $\tilde{x}_j$ is in fact is the length of orthogonal projection, onto orthonormal basis formed by EVc's of A. Rewriting it further gives

$$\frac{1}{\|\mathbf{x}\|}(\mathbf{u}_j^\top A^{-1}\mathbf{b}) = \frac{1}{\|\mathbf{x}\|}(\langle \mathbf{u_j}| A^{-1}\frac{\mathbf{b}}{\|\mathbf{b}\|}\|\mathbf{b}\|) = \frac{\|\mathbf{b}\|}{\|\mathbf{x}\|}(\langle \mathbf{u_j}| A^{-1}|\mathbf{b}\rangle) = \frac{1}{\|\mathbf{x}\|}\frac{\beta_j}{\lambda_j} \tag{10}$$

$$\Rightarrow \tilde{x}_j = \frac{\beta_j}{\lambda_j} \Rightarrow |\mathbf{x}\rangle = \frac{1}{\|\mathbf{x}\|}\sum_{j=1}^{n} \frac{\beta_j}{\lambda_j} |\mathbf{u_j}\rangle = |A^{-1}\mathbf{b}\rangle$$

And its normed version

$$\frac{|A^{-1}\mathbf{b}\rangle}{\||A^{-1}\mathbf{b}\rangle\|} = \frac{1}{\sqrt{\sum_{j=1}^{n} \frac{|\beta_j|^2}{|\lambda_j|^2}}} \sum_{j=1}^{n} \frac{\beta_j}{\lambda_j} |\mathbf{u_j}\rangle = \frac{A^{-1}|\mathbf{b}\rangle}{\|A^{-1}|\mathbf{b}\rangle\|} \Rightarrow |A^{-1}\mathbf{b}\rangle \propto A^{-1}|\mathbf{b}\rangle \tag{11}$$

□

5.3 Detailed explanation

Here we provide detailed explanation, together with primitive numerical examples, and bloch-spheres representations of the system on each step, for system $A\mathbf{x} = \mathbf{b}$

$$A = \begin{bmatrix} 1 & -\frac{1}{3} \\ -\frac{1}{3} & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Through the algorithm, c-register will store eigenvalues of A, however, it will not store exact values, but scaled to integer versions. This is due the fact, that we want to measure these values accurately, for this we need to have eigenvalues be represented in classical binary representation (i.e. $0010 = 2$). This can be easily done, if c-register is represented through elementary basis vectors. For this we introduce new notation for elementary basis vectors

$$|00\rangle = |0\rangle = \mathbf{e_1} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^\top, |10\rangle = |2\rangle = \mathbf{e_3} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^\top \tag{12}$$

To generalize, for function f that takes any binary vector and returns decimal number that it represents

$$f : \mathbb{F}_2 \to \mathbb{N} \cup \{0\} \tag{13}$$

we have

$$(\forall \mathbf{v} \in \mathbb{F}_2) \, |\mathbf{v}\rangle = |f(\mathbf{v})\rangle = \mathbf{e}_{f(\mathbf{v})+1} \tag{14}$$

Where $\mathbf{e}_i$ is standard basis vector (i.e. j'th element of $\mathbf{e}_i$ is δ_{ij}. This notation also shows, how many qubits we need to store decimal number (i.e. $|3\rangle = |11\rangle$ - requires 2 qubits, we get the system $(0\,|0\rangle + 1\,|1\rangle) \otimes (1\,|0\rangle + 0\,|1\rangle)$).

Set n_b - number of qubits in b-register, n number of qubits in c-register, $N = 2^n, N_b = 2^{n_b}$.

Algorithm starts in initial quantum system state

$$|\psi_\mathbf{0}\rangle = |\text{b-register}\rangle \, |\text{c-register}\rangle \, |\text{ancila-bit}\rangle = |0\rangle^{\otimes n_b} \, |0\rangle^{\otimes n} \, |0\rangle_a \tag{15}$$

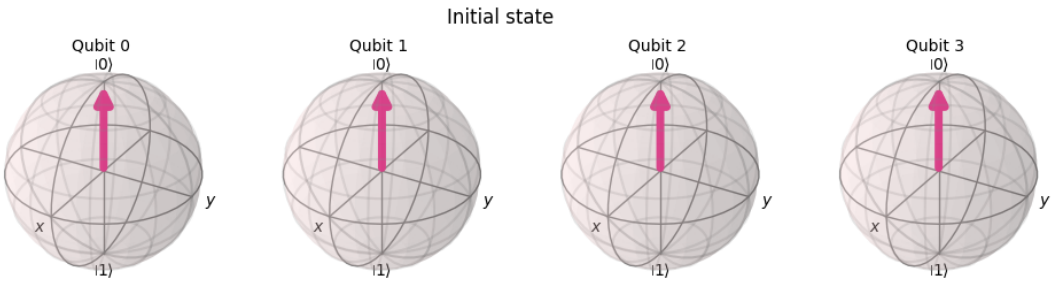


Figure 1: Qubit's initial state.

After that we encode $\mathbf{b}$ as $|\mathbf{b}\rangle$, and put this value to b-register. As was mentioned earlier, through the algorithm we will need $\mathbf{b}$'s eigen decomposition, however we may still encode $|\mathbf{b}\rangle$ through normalization, because it will store the same result as eigen decomposition.

$$|\psi_\mathbf{1}\rangle = |\mathbf{b}\rangle \, |0\rangle^{\otimes n} \, |0\rangle \tag{16}$$

We apply Hadamard 4 gates on each qubit in c-register

$$|\psi_\mathbf{2}\rangle = |\mathbf{b}\rangle \otimes \underbrace{H\,|0\rangle \otimes \cdots \otimes H\,|0\rangle}_{n \text{ times}} \otimes |0\rangle \tag{17}$$

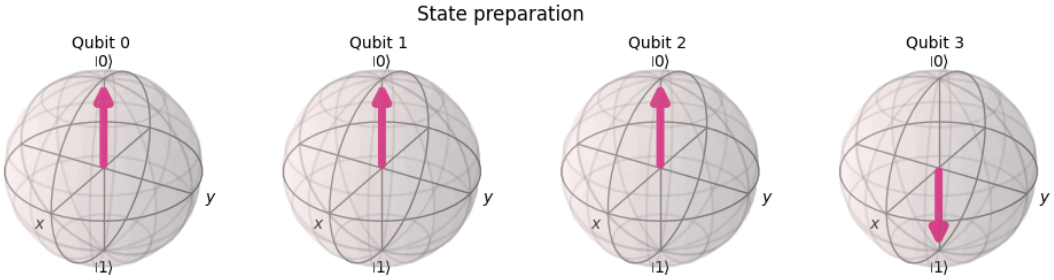


Figure 2: System's state after state preparation.

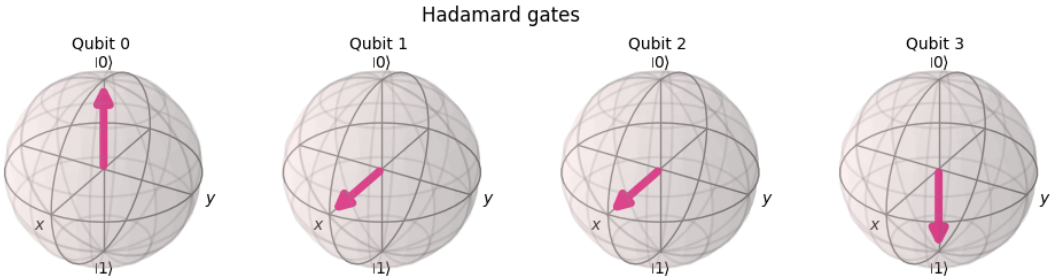


Figure 3: System's state after Hadamard gate on each control qubit.

For qubit's state $\alpha\mathbf{e}_1 + \beta\mathbf{e}_2$, α^2 and β^2 are probabilities of qubit to be measured as $|0\rangle$ or $|1\rangle$ respectively. What Hadamard in fact does, is that it sets each qubit into state

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{18}$$

where it has equal probability to be measured either as 0 or 1. Thus, our quantum system is in state

$$|\psi_{\mathbf{2}}\rangle = |\mathbf{b}\rangle \left(\frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\end{bmatrix} + \frac{1}{\sqrt{2}}\begin{bmatrix}0\\1\end{bmatrix}\right)^{\otimes n} |0\rangle \tag{19}$$

Now, we need to apply controlled 5 rotation U^{2^r}, where r - is qubit's number. We now introduce phenomena called phase kickback. If we have qubit in state $|\mathbf{a}\rangle = \begin{bmatrix}a_0 & a_1\end{bmatrix}^\top$, which is an eigenvector of some operator X, with eigenvalue λ_a. Thus, before controlled X rotation ($|\rangle_c$ is control qubit), if we have system of the following form

$$|\xi\rangle = |\mathbf{a}\rangle H|\mathbf{0}\rangle = |\mathbf{a}\rangle \otimes \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \tag{20}$$

and we apply controlled X rotation (i.e. when control qubit read as 1) our system changes to

$$|\xi_{\mathbf{1}}\rangle = X|\mathbf{a}\rangle \otimes |1\rangle = \begin{bmatrix}0\\ \lambda_a a_0\\ 0\\ \lambda_a a_1\end{bmatrix} \tag{21}$$

or if we do not apply the rotation (control qubit read as 0)

$$|\xi_{\mathbf{0}}\rangle = |\mathbf{a}\rangle \otimes |0\rangle = \begin{bmatrix}a_0\\ 0\\ a_1\\ 0\end{bmatrix} \tag{22}$$

We see, that actually we do not have to apply controlled X $(C-X)$ rotation on $|\mathbf{a}\rangle$ qubit, but we can apply it only on control qubit (s).

$$(C-X)\,|\mathbf{a}\rangle \otimes H\,|0\rangle = |\mathbf{a}\rangle \otimes \left(\frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}X\,|1\rangle\right) = \frac{1}{\sqrt{2}}\begin{bmatrix} a_0 \\ \lambda_a a_0 \\ a_1 \\ \lambda_1 a_1 \end{bmatrix} \tag{23}$$

We see, that resulting vector representation of quantum system corresponds to the cases when control qubit was read as 0 (X was not applied) and otherwise.

So, now we have more convenient representation of controlled U rotation

$$|\psi_{\mathbf{3}}\rangle = |\mathbf{b}\rangle \otimes \frac{1}{\sqrt{2^n}}\left(|0\rangle + U^{2^{n-1}}\,|1\rangle\right) \otimes \left(|0\rangle + U^{2^{n-2}}\,|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + U^{2^0}\,|1\rangle\right) \otimes |0\rangle \tag{24}$$

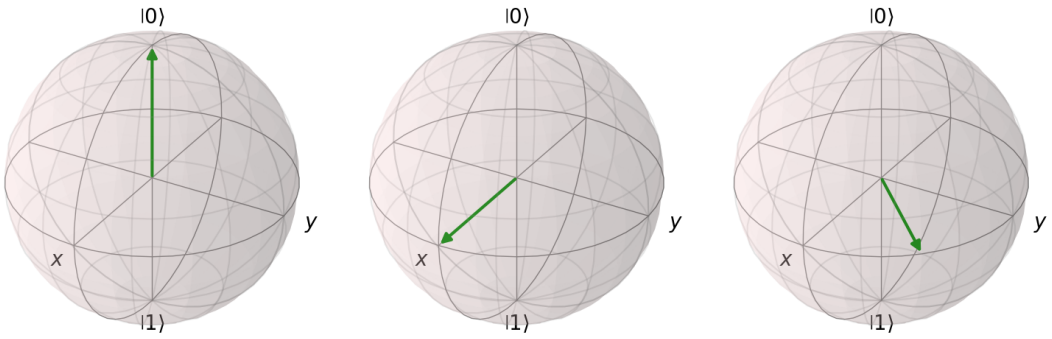


Figure 4: Qubits' state after controlled rotation.

We can now "abuse" symmetry and invertibility of matrix A, which gives us orthonormal basis formed by EVc's of A and the fact that we have applied Hadamard gates. After applying hadamard gate (second bloch sphere of Figure 4), we can change phase of the qubit, by simply multiplying its state by $e^{i2\pi\phi}$. So, let us assume that $|\mathbf{b}\rangle$ is an eigenvector of U, with respective eigenvalue $e^{i2\pi\phi}$ (i.e. $U\,|\mathbf{b}\rangle = e^{i2\pi\phi}\,|\mathbf{b}\rangle$). We need this assumption, because phase ϕ (angle of rotation) stores information of EV's of A (because for some EVc $\mathbf{u}_j, e^{i2\pi\phi}\mathbf{u}_j = U\mathbf{u}_j = e^{iAt}\mathbf{u}_j = e^{i\lambda_j t}\mathbf{u}_j$, because of diagonalization). We **can** make this assumption, because we are applying this gate on $|\mathbf{b}\rangle$ (even though using phase kickback), which will be rewritten as linear combination of EVc's. We rewrite the system as

$$|\psi_{\mathbf{3}}\rangle = |\mathbf{b}\rangle \otimes \frac{1}{\sqrt{2^n}}\left(|0\rangle + e^{i2\pi\phi 2^{n-1}}\,|1\rangle\right) \otimes \left(|0\rangle + e^{i2\pi\phi 2^{n-2}}\,|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{i2\pi\phi 2^0}\,|1\rangle\right) \otimes |0\rangle \tag{25}$$

Next, for simple 2 qubit example we have the following identity

$$\begin{aligned}\left(|0\rangle + e^{i2\pi\phi 2^{2-1}}\,|1\rangle\right) \otimes \left(|0\rangle + e^{i2\pi\phi 2^{2-2}}\,|1\rangle\right) &= \left(\begin{bmatrix}1\\0\end{bmatrix} + \begin{bmatrix}0\\e^{i4\pi\phi}\end{bmatrix}\right) \otimes \left(\begin{bmatrix}1\\0\end{bmatrix} + \begin{bmatrix}0\\e^{i2\pi\phi}\end{bmatrix}\right) \\ = \begin{bmatrix}1\\e^{i4\pi\phi}\end{bmatrix} \otimes \begin{bmatrix}1\\e^{i2\pi\phi}\end{bmatrix} = \begin{bmatrix}1\\e^{i2\pi\phi}\\e^{i4\pi\phi}\\e^{i6\pi\phi}\end{bmatrix} &= \sum_{k=0}^{2^2-1} e^{i2\pi\phi k}\mathbf{e}_{k+1} = \sum_{k=0}^{2^2-1} e^{i2\pi\phi k}\,|k\rangle\end{aligned} \tag{26}$$

Observing former identity we can rewrite our system state as

$$|\psi_{\mathbf{3}}\rangle = |\mathbf{b}\rangle \otimes \left(\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{i2\pi\phi k}\,|k\rangle\right) \otimes |0\rangle \tag{27}$$

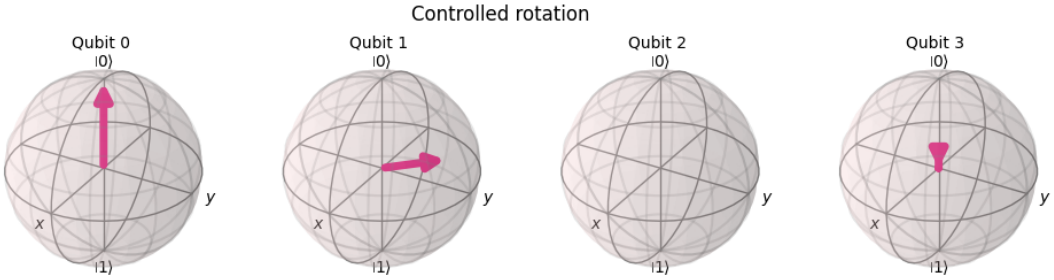


Figure 5: Qubit's state after Hadamard gate and phase rotation.

This step is crucial, as it gives all possible ϕ rotation in multidimensional circle, formed by control qubits system. This angle is now extracted applying Discrete Fourier transform[8]

Definition 1. *Discrete Fourier transform (DFT) is a linear transformation that acts on elementary basis vectors the following way*

$$\mathbf{e}_{j+1} \xrightarrow{\text{DFT}} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{-i2\pi j \frac{k}{N}} \mathbf{e}_{k+1} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{-i2\pi j \frac{k}{N}} |k\rangle \tag{28}$$

It follows that Discrete Fourier transform can be rewritten in matrix form

$$\text{DFT} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & (e^{i2\pi/N})^{-(1\cdot 1)} & \dots & (e^{e^{i}2\pi/N})^{-(1\cdot(N-1))} \\ \dots & \dots & \ddots & \dots \\ 1 & (e^{i2\pi/N})^{-((N-1)\cdot 1)} & \dots & (e^{i2\pi/N})^{-((N-1)\cdot(N-1))} \end{bmatrix} \tag{29}$$

Applying DFT to our system

$$|\psi_4\rangle = \frac{1}{2^{n/2}} |\mathbf{b}\rangle \otimes \text{DFT} \left(\sum_{k=0}^{2^n-1} e^{i2\pi\phi k} |k\rangle \right) \otimes |0\rangle \tag{30}$$

$$= \frac{1}{2^{n/2}} |\mathbf{b}\rangle \otimes \left(\sum_{k=0}^{2^n-1} e^{i2\pi\phi k} \text{DFT} \ |k\rangle \right) \otimes |0\rangle \tag{31}$$

$$= \frac{1}{2^{n/2}} |\mathbf{b}\rangle \otimes \left(\sum_{k=0}^{2^n-1} e^{i2\pi\phi k} \frac{1}{\sqrt{N}} \sum_{y=0}^{2^n-1} e^{-i2\pi k y/N} |y\rangle \right) \otimes |0\rangle \tag{32}$$

$$= \frac{1}{2^{n/2}} |\mathbf{b}\rangle \otimes \left(\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \sum_{y=0}^{2^n-1} e^{i2\pi\phi k} e^{-i2\pi k y/N} |y\rangle \right) \otimes |0\rangle \tag{33}$$

$$= \frac{1}{2^{n/2}} |\mathbf{b}\rangle \otimes \left(\frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} \sum_{y=0}^{2^n-1} e^{i2\pi k(\phi - y/N)} |y\rangle \right) \otimes |0\rangle \tag{34}$$

Because columns of matrix form of DFT are orthogonal in $\mathbb{C}^n$, follows $\phi \neq y/N \Rightarrow e^{i2\pi k(\phi - y/N)} = 0$. Otherwise, $\phi = y/N \Rightarrow e^{i2\pi k(\phi - y/N)} = 1$ (i.e. $y = N\phi$). We can further rewrite our equation

$$|\psi_4\rangle = \frac{1}{2^n} |\mathbf{b}\rangle \otimes \left(\sum_{k=0}^{2^n-1} e^{i2\pi k 0} |N\phi\rangle \right) \otimes |0\rangle = |\mathbf{b}\rangle |N\phi\rangle |0\rangle \tag{35}$$

Using our previous assumption 5.3 of $|\mathbf{b}\rangle$ being eigenvector, we can equate $i\lambda_j t = i2\pi\phi \Rightarrow \phi = \frac{\lambda_j t}{2\pi}$

$$|\psi_{\mathbf{5}}\rangle = \sum_{j=0}^{2^{n_b}-1} \beta_j |\mathbf{u_j}\rangle |N\lambda_j t/2\pi\rangle |0\rangle \tag{36}$$

Set $\tilde{\lambda}_j = \frac{N\lambda_j t}{2\pi}$. As EV's are usually not integers, we will have to choose t such that $\tilde{\lambda}_j$ is integer. One might wonder, how do we estimate such t when we do not know EV's? However, remembering our assumption on A being sparse and that U is simulated through as Hamiltonian (that requires A to be transformed to adjacency matrix of graph, for which we can easily estimate the biggest EV) we can will find t that makes even the larget EV scaled to integer.

$$|\psi_{\mathbf{5}}\rangle = \sum_{j=0}^{2^{n_b}-1} \beta_j |\mathbf{u_j}\rangle |\tilde{\lambda}_j\rangle |0\rangle \tag{37}$$

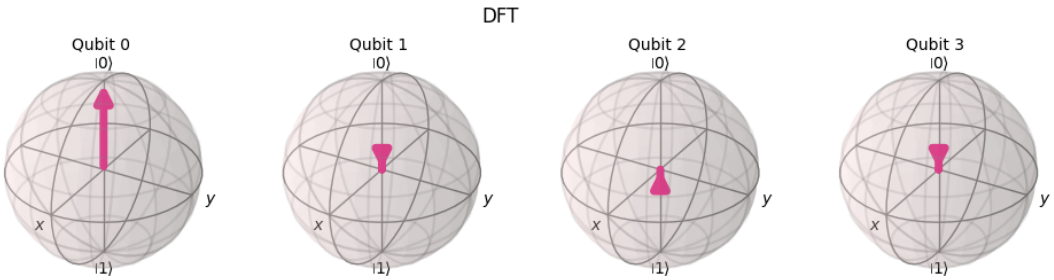


Figure 6: Qubit's state after applying DFT.

We now want to rotate ancila bit, so the system is in state

$$|\psi_{\mathbf{5}}\rangle = \sum_{j=0}^{2^{n_b}-1} \beta_j |\mathbf{u_j}\rangle |\tilde{\lambda}_j\rangle \left(\sqrt{1 - \frac{C^2}{\tilde{\lambda}_j^2}} |0\rangle + \frac{C}{\tilde{\lambda}_j} |1\rangle\right) \tag{38}$$

to this end, we need to apply $R_y(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$ of ancila qubit, choosing such $\theta = 2\arcsin(\frac{C}{\tilde{\lambda}_j})$, that we maximize probability of $R_y(\theta)|0\rangle_a$ be read as 1. Again, it requires some estimation of the smallest eigenvalue (C cannot be greater than $\tilde{\lambda}_j$ - to not break probability theory axioms!). After ancila bit is measured as 1, we have state of the form

$$|\psi'_{\mathbf{6}}\rangle = \sum_{j=0}^{2^{n_b}-1} \frac{C}{\tilde{\lambda}_j} \beta_j |\mathbf{u_j}\rangle |\tilde{\lambda}_j\rangle |1\rangle \tag{39}$$

However, such vector will again break probabilities, thus we need normalization

$$|\psi_{\mathbf{6}}\rangle = \frac{1}{\sqrt{\sum_{j=0}^{2^{n_b}-1} \left(\frac{C\beta_j}{\tilde{\lambda}_j}\right)^2}} \sum_{j=0}^{2^{n_b}-1} \frac{C}{\tilde{\lambda}_j} \beta_j |\mathbf{u_j}\rangle |\tilde{\lambda}_j\rangle |1\rangle \tag{40}$$

We already see something resembling $|\mathbf{x}\rangle$ in b and c register together, however we cannot read it yet, it cannot be separated. Such state is called entanglement. It means that we

cannot write system's state in form of tensor products of separate qubits, for example

$$|\Psi\rangle = \frac{1}{\sqrt{2}} \left(\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right) \tag{41}$$

is an entangled state, while

$$|\Psi'\rangle = \left(-\frac{1}{2} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right) \tag{42}$$

$$= \left(\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \end{bmatrix} - \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) \otimes \left(-\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) \tag{43}$$

is not. Another interpretation is if we have two 1-dimensional functions

$$f : X \to A, g : Y \to B \tag{44}$$

if another function $h : X \times Y \to M$ is not entangled, we can rewrite $h(x, y) = f(x)g(y)$.

To unentangle b-register and c-register we apply inverse procedures in reverse order. Firstly we apply inverse of Discrete Fourier transform

$$\text{IDFT} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & (e^{i2\pi/N})^{(1\cdot 1)} & \dots & (e^{e^{i}2\pi/N})^{(1\cdot(N-1))} \\ \dots & \dots & \ddots & \dots \\ 1 & (e^{i2\pi/N})^{((N-1)\cdot 1)} & \dots & (e^{i2\pi/N})^{((N-1)\cdot(N-1))} \end{bmatrix} \tag{45}$$

$$|\psi_{\mathbf{7}}\rangle = \text{IDFT}\,|\psi_{\mathbf{6}}\rangle = \frac{1}{\sqrt{\sum_{j=0}^{2^{n_b}-1} \left(\frac{C\beta_j}{\tilde{\lambda}_j}\right)^2}} \sum_{j=0}^{2^{n_b}-1} \frac{C}{\tilde{\lambda}_j} \beta_j \,|\mathbf{u_j}\rangle\, \text{IDFT}\,|\tilde{\lambda}_j\rangle\,|1\rangle \tag{46}$$

Because $\tilde{\lambda}_j$ is scaled to integer version of λ_j, it is the same as to apply IDFT to elementary basis vector

$$|\psi_{\mathbf{7}}\rangle = \frac{1}{\sqrt{\sum_{j=0}^{2^{n_b}-1} \left(\frac{C\beta_j}{\tilde{\lambda}_j}\right)^2}} \sum_{j=0}^{2^{n_b}-1} \frac{C}{\tilde{\lambda}_j} \beta_j \,|\mathbf{u_j}\rangle\, \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{i2\pi k\tilde{\lambda}_j/N}\,|k\rangle\,|1\rangle \tag{47}$$

$$= \frac{1}{2^{n/2}\sqrt{\sum_{j=0}^{2^{n_b}-1} \left(\frac{C\beta_j}{\tilde{\lambda}_j}\right)^2}} \sum_{j=0}^{2^{n_b}-1} \frac{C}{\tilde{\lambda}_j} \beta_j \,|\mathbf{u_j}\rangle \sum_{k=0}^{2^n-1} e^{i2\pi k\tilde{\lambda}_j/N}\,|k\rangle\,|1\rangle \tag{48}$$

$$\tag{49}$$

We now apply inverse of control U rotation (i.e. U^{-1}), following same logic as we did in (27)

$$|\psi_{\mathbf{8}}\rangle = \frac{1}{2^{n/2}\sqrt{\sum_{j=0}^{2^{n_b}-1} \left(\frac{C\beta_j}{\tilde{\lambda}_j}\right)^2}} \sum_{j=0}^{2^{n_b}-1} \frac{C}{\tilde{\lambda}_j} \beta_j \,|\mathbf{u_j}\rangle \sum_{k=0}^{2^n-1} e^{-i\lambda_j tk} e^{i2\pi k\tilde{\lambda}_j/N}\,|k\rangle\,|1\rangle \tag{50}$$

However, as $\tilde{\lambda}_j = N\lambda_j t/2\pi$, we can rewrite the equation and cancel out exponents

$$|\psi_{\mathbf{8}}\rangle = \frac{1}{2^{n/2}\sqrt{\sum_{j=0}^{2^{n_b}-1}\left(\frac{C\beta_j}{\tilde{\lambda}_j}\right)^2}} \sum_{j=0}^{2^{n_b}-1} \frac{C}{\tilde{\lambda}_j}\beta_j |\mathbf{u_j}\rangle \sum_{k=0}^{2^n-1} e^{-i\lambda_j tk} e^{i2\pi k\tilde{(}N\lambda_j t)/N2\pi} |k\rangle |1\rangle \tag{51}$$

$$= \frac{1}{2^{n/2}\sqrt{\sum_{j=0}^{2^{n_b}-1}\left(\frac{C\beta_j}{\tilde{\lambda}_j}\right)^2}} \sum_{j=0}^{2^{n_b}-1} \frac{C}{\tilde{\lambda}_j}\beta_j |\mathbf{u_j}\rangle \sum_{k=0}^{2^n-1} e^{-i\lambda_j tk} e^{i\lambda_j tk} |k\rangle |1\rangle \tag{52}$$

$$= \frac{1}{2^{n/2}\sqrt{\sum_{j=0}^{2^{n_b}-1}\left(\frac{C\beta_j}{\tilde{\lambda}_j}\right)^2}} \sum_{j=0}^{2^{n_b}-1} \frac{C}{\tilde{\lambda}_j}\beta_j |\mathbf{u_j}\rangle \sum_{k=0}^{2^n-1} |k\rangle |1\rangle \tag{53}$$

$$= \frac{C}{2^{n/2}\sqrt{\sum_{j=0}^{2^{n_b}-1}\left(\frac{C\beta_j}{\tilde{\lambda}_j}\right)^2}} \sum_{j=0}^{2^{n_b}-1} \frac{\beta_j}{\tilde{\lambda}_j} |\mathbf{u_j}\rangle \sum_{k=0}^{2^n-1} |k\rangle |1\rangle \tag{54}$$

$$= \frac{1}{2^{n/2}\sqrt{\sum_{j=0}^{2^{n_b}-1}\left(\frac{\beta_j}{\tilde{\lambda}_j}\right)^2}} \sum_{j=0}^{2^{n_b}-1} \frac{\beta_j}{\tilde{\lambda}_j} |\mathbf{u_j}\rangle \sum_{k=0}^{2^n-1} |k\rangle |1\rangle \tag{55}$$

$$= \frac{1}{2^{n/2}\sqrt{\sum_{j=0}^{2^{n_b}-1}\left(\frac{1}{\frac{Nt}{2\pi}}\right)^2\left(\frac{\beta_j}{\lambda_j}\right)^2}} \frac{1}{\frac{Nt}{2\pi}} \sum_{j=0}^{2^{n_b}-1} \frac{\beta_j}{\lambda_j} |\mathbf{u_j}\rangle \sum_{k=0}^{2^n-1} |k\rangle |1\rangle \tag{56}$$

$$= \frac{1}{2^{n/2}\sqrt{\sum_{j=0}^{2^{n_b}-1}\left(\frac{\beta_j}{\lambda_j}\right)^2}} |\mathbf{x}\rangle \sum_{k=0}^{2^n-1} |k\rangle |1\rangle \tag{57}$$

$$\tag{58}$$

We have already unentagled registers and can get solution from b-register. Now we apply Hadamard gates on c-registers once more, before this notice following identity

$$\sum_{y=0}^{2^2-1} |y\rangle = \begin{bmatrix}1\\1\\1\\1\end{bmatrix}, H\otimes H \begin{bmatrix}1\\1\\1\\1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} = \frac{1}{2^{2/2}}\begin{bmatrix}1\\0\\0\\0\end{bmatrix} \tag{59}$$

Observing this identity and applying Hadamard gate to each qubit in c-register our system's state vector changes to

$$|\psi_{\mathbf{9}}\rangle = \frac{1}{2^{n/2}\sqrt{\sum_{j=0}^{2^{n_b}-1}\left(\frac{\beta_j}{\lambda_j}\right)^2}} |\mathbf{x}\rangle \left(H^{\otimes n}\sum_{k=0}^{2^n-1}|k\rangle\right)|1\rangle = \frac{1}{\sqrt{\sum_{j=0}^{2^{n_b}-1}\left(\frac{\beta_j}{\lambda_j}\right)^2}} |\mathbf{x}\rangle |0\rangle^{\otimes n} |1\rangle \tag{60}$$

As $\||\mathbf{x}\rangle\| = \left\|\sum_{j=1}^{n}\frac{\beta_j}{\lambda_j}|\mathbf{u_j}\rangle\right\| = 1 \Rightarrow \sum_{j=1}^{n}\left(\frac{\beta_j}{\lambda_j}\right)^2 = 1$. Our system results in

$$|\psi_{\mathbf{9}}\rangle = |\mathbf{x}\rangle_b |0\rangle_c^{\otimes n} |1\rangle_a \tag{61}$$

We see that solution vector is stored in b-register.

5.4 Algorithm complexity

The complexity of the algorithm is $O(\log N) * k^2 * s^2/\epsilon$, where N is the number of variables in the linear system, k is the conditional variable (maximum eigenvalue divided by minimum eigenvalue in our case), s is the sparsity(maximum number of non-zero entries in any row) of the matrix, ϵ is the desired error tolerance.

1. **Dependence on** $log(N)$. The HHL algorithm encodes an N-dimensional vector in $O(log(N))$ qubits.
2. **Dependence on** k^2. The condition number (k) measures how ill-conditioned the system is. A large k means the matrix is close to singular, making the problem harder. In HHL, the runtime scales as k^2 because: the phase estimation step (key to extracting eigenvalues) has an error that scales with k.
3. **Dependence on** s. The sparsity s affects the cost of Hamiltonian simulation (e^{iAt}). If the matrix is sparse ($s << N$), Hamiltonian simulation is efficient.
4. **Dependence on** ϵ. The error tolerance ϵ appears because quantum algorithms often trade off precision for speed. Achieving a smaller ϵ requires more resources.

We must also notice, that reading whole solution vector scales to $O(N)$ complexity, which devalues main advantage of exponential scaling with quantum computing. Thus this algorithm should be used only when we know, what entries of $|\mathbf{x}\rangle$ we will need (this goes well with assumption that matrix A is sparse).

6 Simulation Results

We tried to run the algorithm for the following systems

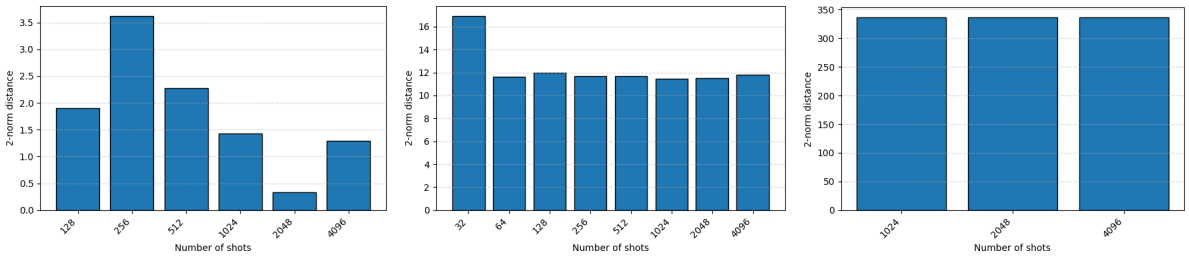
$$\begin{bmatrix} 1 & -\frac{1}{3} \\ -\frac{1}{3} & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2.6 & 0.02 & -0.7 & -0.78 \\ 0.02 & 2.69 & 0.48 & -0.48 \\ -0.7 & 0.48 & 1.46 & 0.28 \\ -0.78 & -0.48 & 0.28 & 3.27 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{62}$$

$$\begin{bmatrix} 1.34 & 0.11 & 0.08 & -0.60 & 0.06 & -0.05 & -0.17 & 0.01 \\ 0.11 & 1.19 & -0.22 & -0.23 & 0.18 & 0.16 & -0.10 & 0.12 \\ 0.08 & -0.22 & 1.66 & -0.04 & 0.22 & -0.46 & 0.32 & 0.13 \\ -0.60 & -0.23 & -0.04 & 2.06 & -0.13 & 0.07 & 0.46 & -0.06 \\ 0.06 & 0.18 & 0.22 & -0.13 & 2.11 & -0.25 & 0.24 & 0.81 \\ -0.05 & 0.16 & -0.46 & 0.07 & -0.25 & 1.54 & 0.24 & -0.25 \\ -0.17 & -0.10 & 0.32 & 0.46 & 0.24 & 0.24 & 2.47 & -0.03 \\ 0.01 & 0.12 & 0.13 & -0.06 & 0.81 & -0.25 & -0.03 & 1.63 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 2 \\ 3 \end{bmatrix} \tag{63}$$

We generated those systems by creating vector of random eigenvalues (but not to big in ratio) and random orthogonal matrices (act like matrices of orthogonal EVc's for symmetric matrix).

For each obtained solution (using classical Gauss-Jordan elimination and HHL) we created vector of pairwise squared ratios and to normed distance between two vecotrs of ratios.

Below you can find plots of previously described distances for different number of algorithm runs:



(a) 2x2 system (b) 4x4 system (c) 8x8 system

You can notice, that for 8x8 system we have not included number of shots less than 1024. This is dues the reason it refused to terminate (i.e. ancilla bit was not measured as 1). It leads us to hypothesis, that the bigger system the more times we have to run algorithm on the noisy hardware.

References

[1] Gilbert Strang, *Introduction to Linear Algebra*, 5th Edition, Wellesley–Cambridge Press, 2016.

[2] `https://github.com/TarnMaVas/LA_HHL_project/`

[3] `https://commons.wikimedia.org/wiki/File:Bloch_sphere.svg`

[4] Time evolution in quantum mechanics,y Robert G. Littlejohn. `https://bohr.physics.berkeley.edu/classes/221/1112/notes/tevolut.pdf`

[5] On Quantum Algorithms for Solving Linear Systems of Equations, Adrien Vandenbroucque `https://adrienvdb.com/wp-content/uploads/2021/11/Semester_Project_Final.pdf`

[6] The tensor product, demistified `https://www.math3ma.com/blog/the-tensor-product-demystified`

[7] Creating superpositions that correspond to efficiently integrable probability distributions, Lov Grover and Terry Rudolph `https://arxiv.org/pdf/quant-ph/0208112`

[8] THE DISCRETE FOURIER TRANSFORM, Bertrand Delgutte and Julie Greenberg. `https://web.mit.edu/~gari/teaching/6.555/lectures/ch_DFT.pdf`

[9] Quantum linear systems algorithms: a primer `https://arxiv.org/pdf/1802.08227`

[10] Efficient quantum algorithms for simulating sparse Hamiltonians `https://arxiv.org/pdf/quant-ph/0508139`

[11] Quantum circuits for solving linear systems of equations `https://arxiv.org/pdf/1110.2232`

[12] A Step-by-Step HHL Algorithm Walkthrough to Enhance Understanding of Critical Quantum Computing Concepts, Anika Zaman, Hector Jose Morrell, Hiu Yung Wong. `https://arxiv.org/pdf/2108.09004`