# Quantum Algorithm for Linear Systems of Equations

Seminar on Applications of Quantum Computing in IT Security

Alfred Nguyen
*Department of Computer Science*
*Technical University of Munich*
85748 Garching, Bavaria
alfred.nguyen@tum.de

***Abstract*—Linear systems are a fundamental problem in math and can be found as 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 to be solved. The HHL (Harrow, Hassidim, and Lloyd) algorithm is a quantum algorithm, that can solve these linear systems of equations exponentially faster than its classical counterpart. Though, there are a few caveats to consider. We assume that we are only interested in solving 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 has the size $N \times N$. 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).**

## I. 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 computer cannot simulate. In some cases, quantum computer algorithms can provide an exponential speed-up in comparison to common classical methods. The most famous case is Shor's algorithm for factoring large numbers, possibly cracking current RSA encryption.

Solving linear systems is a fundamental subproblem in many scientific, engineering, and mathematical disciplines. Currently, the demand for processing data sets is increasing. The sizes of these datasets which make up the equations, are growing well into terabytes and petabytes of data. Obtaining solutions for these kinds of problems can be very computationally expensive. Even an approximation of these solutions with $N$ unknowns takes at least $N$ timesteps, as even just outputting the solution vector takes at least N steps. Though, oftentimes we are not interested in the full solution vector. For example, we can assume that we are only interested in a function, that determines the weights of some subsets in a specific region of the solution vector. Then we can achieve an exponential faster approximation by using quantum algorithms.

Especially in quantum machine learning, these kinds of problems arise very frequently. Using large datasets, finding patterns, classifying and clustering are very common procedures, demanding a lot of resources. Many of the quantum machine learning algorithms extend or use the following algorithm as a subroutine.

This paper will focus on explaining the algorithm, that can estimate features of the solution vector of a linear system of equations. The HHL algorithm, designed by Aram **H**arrow, Avinatan **H**assidim and Seth **L**loyd, was introduced in 2009, solving linear systems [1]. 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).

The HHL algorithm promises an exponential speed-up over commonly used classical methods.

### *A. Outline*

Firstly, this paper will discuss relevant quantum mechanical concepts for the HHL algorithm. After that, we will give a rough overview of the HHL algorithm, specifying the problem in detail, and giving a mathematical idea of the algorithm, followed by an explanation of the quantum circuit. Then, we will discuss the HHL algorithm, going through all the phases of the algorithm in detail. To increase the understanding of the HHL algorithm, a numerical example will be discussed with a 4-qubit quantum circuit. Next, we evaluate and analyze its runtime by comparing the HHL algorithm to its classical counterpart. Afterward, we will take an outlook at future works, exploring potential applications, and variations of the HHL algorithm and will relate it to IT security. In the end, we will give a conclusion, summarizing our findings.

## II. Quantum computing concepts

In this section, we will define common quantum computing concepts, relevant for understanding the HHL algorithm.

### A. Bra-ket Notation

The *bra-ket* notation is commonly used to represent quantum states, operators and measurements. To begin with, a qubit can be represented by a 2d-vector $\vec{v}$ which can be written in its *ket* form $|v\rangle$. The most common *kets* are the 0 and 1 *kets*, which are written as $|0\rangle$ and $|1\rangle$ respectively. These are the basis states that represent the 0 and 1 bits on a classical computer. As already mentioned, we can think of a *ket* as a vector. In vector form $|0\rangle$ and $|1\rangle$ are defined by these vectors as such

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{1}$$

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2}$$

Graphically we can think of these two qubits $|0\rangle$ and $|1\rangle$ as two orthonormal vectors one pointing upwards and the other to the right, whilst having the length of 1.

Multiple qubits can be combined into a quantum state often referred to as $|\Psi\rangle$, describing the state of a quantum computer. For example, if we have a quantum computer with 4 qubits which are all initialized to the 0 states, where the state $|\Psi\rangle$ would look as such

$$|\Psi\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle = |0000\rangle \tag{3}$$

where $\otimes$ is the tensor product, which is often times simplified into one state.

The inner product of two vectors $\vec{a}$ and $\vec{b}$ is represented in braket notation

$$\langle a|b\rangle \tag{4}$$

where $\langle a|$ as the *bra* and $|b\rangle$ is the *ket*. The *bra* vector is defined as the complex conjugate of the *ket* vector. For example

$$|a\rangle = \begin{pmatrix} 1 \\ -i \end{pmatrix} \Rightarrow \langle a| = \begin{pmatrix} 1 & i \end{pmatrix} \tag{5}$$

Note that the vectors contain complex numbers.

### B. Superposition and Quantum Gate

As we already know, qubits of a quantum computer can be in states *between* 0 and 1. These states are superpositions where a qubit is neither really 0 nor 1. Mathematically, superpositions can be represented by a linear combination of basis states. The most common superposition is created by the Hadamard gate, which puts a basis state into an equal superposition. This means, the probability of the qubit collapsing to either 0 or 1 is $\frac{1}{2}$. The Hadamard gate $H$ looks like this

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{6}$$

Note that a quantum gate is represented by a matrix. That means, if we want to apply quantum gates on qubits, we calculate transformations of vectors by matrices. Let us apply the Hadamard gate $H$ to the basis state $|0\rangle$

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle \end{aligned} \tag{7}$$

Graphically, the vector of the Hadamard state $|+\rangle$ lies exactly in the middle between the basis vectors $|0\rangle$ and $|1\rangle$. That means the state has an equal probability to collapse to the $|0\rangle$ or $|1\rangle$ state.

### C. Hermitian matrix

A Hermitian matrix is a matrix, which is equal to its adjoint matrix. That means a matrix containing complex entries, is equal to itself after transposing and taking the complex conjugate of itself. Thus, a hermitian matrix $A^\dagger$ is defined as such

$$A = \overline{A^T} \Leftrightarrow A^\dagger \tag{8}$$

For example, the following matrix is Hermitian

$$\begin{aligned} A &= \begin{bmatrix} 2 & 1-i \\ 1+i & 3 \end{bmatrix} \\ \overline{A} &= \begin{bmatrix} 2 & 1+i \\ 1-i & 3 \end{bmatrix} \\ \overline{A^T} &= \begin{bmatrix} 2 & 1-i \\ 1+i & 3 \end{bmatrix} = A \\ \Rightarrow A &= \overline{A^T} = A^\dagger \end{aligned} \tag{9}$$

Hermitian matrices are important as we can easily encode them into quantum systems. More details will follow in the subsection types of encoding II-E.

### D. Spectral Decomposition

Let's say we have a Matrix $A$ that we can diagonalize. Then we can describe it by its spectrum which is a matrix with the eigenvalues in its diagonal

$$\begin{aligned} A &= UDU^\dagger \\ &= \begin{pmatrix} U_1 & \dots & U_n \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_n \end{pmatrix} \begin{pmatrix} U_1^\dagger \\ \vdots \\ U_n^\dagger \end{pmatrix} \end{aligned} \tag{10}$$

where $D$ is the diagonal matrix consisting of the eigenvalues and U consisting of the eigenvectors of $A$. With this, we can describe a matrix only by its eigenvectors and eigenvalues. Similarly, we can also determine the inverse $A^{-1}$ by simply inverting the eigenvalues as such

$$\begin{aligned} A^{-1} &= U^\dagger D^{-1} U \\ &= \begin{pmatrix} U_1^\dagger \\ \vdots \\ U_n^\dagger \end{pmatrix} \begin{pmatrix} \lambda_1^{-1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_n^{-1} \end{pmatrix} \begin{pmatrix} U_1 & \dots & U_n \end{pmatrix} \end{aligned} \tag{11}$$

Classically calculating the inverse matrice by inverting its eigenvalues is comparably slow. Though in the quantum version, we will see, that this will play a big role in making our computation faster, as computing eigenvalues can be done efficiently by Quantum Phase Estimation (QPE).

### *E. Types of Encoding*

Here we will discuss three common kinds of encodings namely, Hamiltonian encoding, Basis encoding, and Amplitude encoding.

*1) Hamiltonian Encoding:* One type of Hamiltonian encoding is to encode a matrix into a unitary gate which can be implemented into a logical quantum gate. That means we can encode a matrix $A$ into our quantum system. For example, a possible unitary gate of the matrix $A$ is

$$U = e^{iAt} \tag{12}$$

if $A$ is a Hermitian matrix.

*2) Basis Encoding:* Basis encoding converts classical numbers into quantum states. By using the binary representation of a number, we can directly write it into our quantum state.

$$x \rightarrow |x\rangle \tag{13}$$

where $x$ is a binary number. Let's say we have $x = 3$ as a decimal number which can be represented as a binary number as $x_b = 11$. Then we can encode it in our quantum state as such

$$x = 3 \xrightarrow{binary} x_b = 11 \xrightarrow{quantum\ state} |x_b\rangle = |11\rangle \tag{14}$$

*3) Amplitude Encoding:* Amplitude encoding encodes a vector $\vec{x}$ into a quantum state $|x\rangle$. We assume, that the vector $\vec{x}$ is normalized. Then we can use the elements of the vector $\vec{x}$ as coefficients of the states $|0\rangle \dots |N-1\rangle$ as such

$$\vec{x} = \begin{pmatrix} x_0 \\ \vdots \\ x_{N-1} \end{pmatrix} \Leftrightarrow |x\rangle = \sum_{i=0}^{N-1} x_i |i\rangle \tag{15}$$

### *F. Kronecker Delta*

The Kronecker delta $\delta_{ij}$ is often used to simplify calculations with inner products $\langle\psi|\phi\rangle$. Basically, $\psi$ and $\phi$ can be handled like normal vectors. The inner product calculates the angle between vectors which means that, for two orthogonal kets, the inner product is 0 as such

$$\langle\psi|\phi\rangle = 0, \text{ if } \psi \perp \phi \tag{16}$$

Let's say we have an orthonormal basis $|E_i\rangle$, meaning all basis vectors are orthonormal to each other. Then, we have the following relation

$$\begin{aligned} \langle E_i|E_i\rangle &= 1 \\ \langle E_i|E_j\rangle &= 0,\ i \neq j \end{aligned} \tag{17}$$

We can see that if the indices are the same, which means the vectors are pointing in the same direction, the inner product is 1. If they are orthogonal to each other, meaning every other vector is an orthonormal basis, then the inner product is $0$. This can be summarized in the Kronecker delta

$$\delta_{ij} = \langle E_i|E_j\rangle = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{18}$$

This can be quite useful in calculations with orthonormal bases. When calculating inner products of an orthonormal basis, whilst the indices are not equal, terms are multiplied by $0$ and will be cancle out, simplifying the equation.

### *G. Entangled states*

Entanglement is a physical property of quantum particles. When particles are entangled, they cannot be described without considering the state of the other particle. Mathematically, entangled states can not be represented by a tensor product of individual states and are defined as such,

$$|\Phi\rangle \neq |\phi\rangle |\psi\rangle \tag{19}$$

where $|\Phi\rangle$ is the quantum state of the system, while $|\phi\rangle$ and $|\psi\rangle$ are individual states.

As an example here we have a state $|\Phi_1\rangle$ which is not entangled, as we can represent it by two other states

$$\begin{aligned} |\Phi_1\rangle &= \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle) \\ &= |1\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ &= |1\rangle |+\rangle \end{aligned} \tag{20}$$

where $|+\rangle$ is the Hadmard state $H|0\rangle$. On the other hand, the following state $|\Phi_2\rangle$ is entangled

$$\begin{aligned} |\Phi_2\rangle &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\ &\neq |\phi\rangle |\psi\rangle \end{aligned} \tag{21}$$

as it cannot be represented by other states.

## III. Overview of the Algorithm

In the following section, we will describe how the algorithm works in general. Firstly, we will specify the problem statement in detail. After that, we will take a look at a mathematical summary of the algorithm. Lastly, we will discuss the quantum circuit, explaining the phases of the circuit [2] [3].

### *A. Problem statement*

Given an $N \times N$ hamiltonian matrix $A$ and vector $\vec{b}$, we want to solve for the vector $\vec{x}$, such that

$$A\vec{x} = \vec{b} \tag{22}$$

To solve for x the equation can be rewritten as

$$\vec{x} = A^{-1}\vec{b} \tag{23}$$

As described earlier, the Hermitian matrix $A^{-1}$ can be split into its spectral decomposition. $A^{-1}$ can be represented in terms of its eigenvectors $U_1 ... U_n$ and inverted eigenvectors $\lambda_1^{-1} ... \lambda_n^{-1}$.

$$A^{-1} = \begin{pmatrix} U_1^\dagger \\ \vdots \\ U_n^\dagger \end{pmatrix} \begin{pmatrix} \lambda_1^{-1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_n^{-1} \end{pmatrix} \begin{pmatrix} U_1 & \dots & U_n \end{pmatrix} \tag{24}$$

This means if we can find the eigenvalues and eigenvectors of $A$ we can then solve the linear equation quite easily. Classical solutions involving spectral decomposition are not faster than other standard algorithms, such as Gaussian Elimination. Though, estimating eigenvalues and eigenvectors can be performed quite efficiently by quantum methods. Via amplitude amplification, QPE can be accelerated to generate the eigenvalues and eigenvectors in $\mathcal{O}(log_2 N)$ steps.

In the quantum version, the linear equation looks like this

$$|x\rangle = A^{-1} |b\rangle \tag{25}$$

where $|b\rangle$ and $|x\rangle$ are the quantum states of the $\vec{b}$ and $\vec{x}$ vectors respectively.

To encode $\vec{b}$ into a quantum state $|b\rangle$ we only need $\mathcal{O}(log_2 N)$ qubits. Hence, we are able to perform everything in $\mathcal{O}(log_2 N)$ so far. Comparing that to the fastest classical methods, which run in $\mathcal{O}(N)$, this promises us an exponential speed up. However, there are some caveats to consider. Reading out the whole solution vector, meaning just reading out every entry of $\vec{x}$ would take $\mathcal{O}(N)$ steps, which would destroy our speed up. But, as we are only interested in an approximation, we can compute an expectation value $\langle x| M |x\rangle$, where $M$ is some linear operator. With this method, we can extract many statistical features like normalization, distribution of weights, moments, etc, without extracting all entries of the solution vector.

### B. Mathematical Overview

We will now look at a mathematical overview of what is happening in the quantum circuit. We assume that the matrix $A$ is Hermitian. If $A$ is not hermitian, we can write $A$ as a hermitian like this

$$A^\dagger = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \tag{26}$$

As already mentioned the matrix can now be described as a linear combination of its outer products of its eigenvectors and its eigenvalues. In the quantum version, the formula looks like this

$$A = \sum_{i=0}^{N-1} \lambda_i |u_i\rangle \langle u_i| \tag{27}$$

where $|u\rangle_i \langle u|_i$ are the outer products of $A$, $\lambda_i$ are the eigenvalues of $A$ and $N$ is the size of the matrix $A$. For the inverse $A^{-1}$, we can rewrite the formula in the following

$$A^{-1} = \sum_{i=0}^{N-1} \lambda_i^{-1} |u_i\rangle \langle u_i| \tag{28}$$

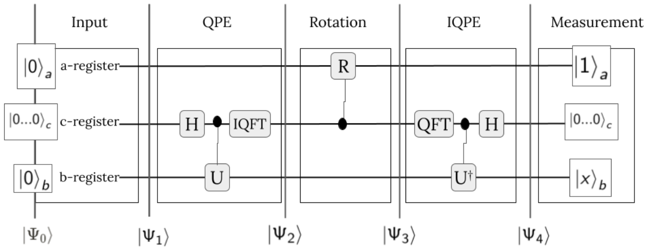

Fig. 1. Example Circuit

Similarly, $|b\rangle$ can be expressed in the eigenbasis of $A$ as

$$|b\rangle = \sum_{j=0}^{N-1} b_j |u_j\rangle \tag{29}$$

With the help of the Kronecker delta $\delta_{ij}$, we have all the tools to solve the equation, by inserting the definition of $A^{-1}$ and $|b\rangle$ into our original equation,

$$\begin{aligned} |x\rangle &= A^{-1} |b\rangle \\ &= \left( \sum_{i=0}^{N-1} \lambda_i^{-1} |u_i\rangle \langle u_i| \right) \left( \sum_{j=0}^{N-1} b_j |u_j\rangle \right) \\ &= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \lambda_i^{N-1} |u_i\rangle \langle u_i| b_j |u_j\rangle \\ &= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \lambda_i^{-1} b_j |u_i\rangle \langle u_i|u_j\rangle \\ &= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \lambda_i^{-1} b_j |u_i\rangle \delta_{ij} \\ &= \sum_{i=0}^{N-1} \lambda_i^{-1} b_j |u_j\rangle \end{aligned} \tag{30}$$

As seen, the solution vector $|x\rangle$ can be calculated only by determining the eigenvectors and eigenvalues of A. Using QPE, calculating the eigenvalues and eigenvectors can be very efficient.

### C. Quantum Circuit

Now we will take a look at how the algorithm is implemented as a quantum circuit. Firstly, we will look at the registers of the quantum circuits. Then, we will describe the phases of the procedure.

*1) Registers:* Fig. 1 shows the scheme of a simple quantum circuit for the HHL algorithm. We have three registers that describe three different sets of qubits in the quantum circuit.

The a-register contains the ancilla qubit. It is used for the inversion of the eigenvalues and will be explained in detail later on.

The c-register, oftentimes referred to as the clock-register, is used for the QPE part. It is related to the time (clock) of the controlled rotation of the qubits and will store the eigenvalues after performing QPE.

The b-register contains the $\vec{b}$ vector which is encoded into a quantum state $|b\rangle$. After the whole HHL procedure is done, the b-register will contain the solution state $|x\rangle$.

*2) Phases:* The procedure of the quantum circuit can be split into 5 phases:

- State preparation
- Quantum phase estimation (QPE)
- Inversion of eigenvalues
- Inverse quantum phase estimation (IQPE)
- Measurement of $|x\rangle$

In the state preparation phase, the vector $\vec{b}$ will be encoded into a quantum state $|b\rangle$ and the $A$ matrix will be encoded as a Hamiltonian, which is a unitary operator $U = e^{iAt}$ into the QPE and IQPE operations.

Then, the QPE will calculate the eigenvalues and eigenvectors of the A matrix.

Afterward, we perform the inversion of the eigenvalues through rotary operations. These operations have a probability to fail, as they are not unitary operators. The ancilla will detect whether the rotation was successful or not and will either collapse to $|0\rangle$ or $|1\rangle$ for failure and success. If the rotation is not successful, the procedure has to be repeated from the beginning. If the rotation was successful we can continue the procedure. The problem is, that the qubits in the b-register and c-register are entangled. This means that we cannot factorize the result into a tensor product of the c-register and b-register. As a result, we cannot convert the b-register into the $|0\rangle$ / $|1\rangle$ measurement basis with the desired amplitudes. We will need to uncompute the state, thus, we have to undo all operations up until now, to unentangle the states whilst keeping the inverted eigenvalues.

This is achieved by the IQPE which undos all steps we performed in the QPE phase, leaving us with the $|0...0\rangle$ state in the c-register and the $|x\rangle$ state in the b-register.

Lastly, the $|x\rangle$ state is to be measured. As mentioned earlier, we can only read out an approximation of an expectation value $\langle x| M |x\rangle$.

## IV. The HHL Algorithm

In the next section, we will give a more detailed walk-through of the HHL algorithm. Thereby, we will go through all 5 phases namely, state preparation, quantum phase estimation, inversion of eigenvalues, inverse quantum phase estimation and lastly the measurement [4].

### *A. State Preparation*

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

$$\begin{aligned} |\Psi_0\rangle &= |0\ldots0\rangle_b \;\; |0\ldots0\rangle_c \;\; |0\rangle_a \\ &= |0\rangle_b^{\otimes n_b} \;\; |0\rangle_c^{\otimes b} \;\; |0\rangle_a \end{aligned} \tag{31}$$

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

$$\vec{b} = \begin{pmatrix} b_0 \\ \vdots \\ b_{N-1} \end{pmatrix} \Leftrightarrow |b\rangle = \sum_{i=0}^{N-1} b_i \,|i\rangle \tag{32}$$

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

$$|\Psi_1\rangle = |b\rangle_b \;\; |0\ldots0\rangle_c \;\; |0\rangle_a \tag{33}$$

### *B. Quantum Phase Estimation*

We will only briefly go through the specifics and will not discuss each step of the QPE in detail, as this is not the main topic of this paper. For further explanations refer to [4] explaining the steps of QPE in detail.

In summary, the QPE is a procedure to evaluate 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 IQFT. After QPE we will have an estimate of the eigenvalues of the unitary $U$. As we have encoded $A$ as a Hamiltonian into $U = e^{iAt}$, the phase of the eigenvalue of U is proportional to the eigenvalue of $A$. Thus, we have to define a scaled version of our eigenvalues $\lambda_j$

$$\widetilde{\lambda_j} = \frac{N\lambda_j t}{2\pi} \tag{34}$$

where $t$ can be chosen 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

$$\begin{aligned} |\Psi_2\rangle &= |b\rangle_b \, |\widetilde{\lambda}\rangle_c \, |0\rangle_a \\ &= \sum_{j=0}^{N-1} b_j \, |u_j\rangle_b \, |\widetilde{\lambda_j}\rangle_c \, |0\rangle_a \end{aligned} \tag{35}$$

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 $A$ as

$$|b\rangle = \sum_{j=0}^{N-1} b_j \, |u_j\rangle \tag{36}$$

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 is

$$|\Psi_3\rangle = \sum_{j=0}^{N-1} b_j \, |u_j\rangle_b \, |\widetilde{\lambda_j}\rangle_c \left( \sqrt{1 - \frac{C^2}{\widetilde{\lambda}_j^2}} \, |0\rangle_a + \frac{C}{\widetilde{\lambda_j}} \, |1\rangle_a \right) \tag{37}$$

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.

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

$$|\Psi_4\rangle = \frac{1}{\sqrt{\sum_{j=0}^{N-1}\left|\frac{b_j C}{\widetilde{\lambda_j}}\right|^2}} \sum_{j=0}^{N-1} b_j |u_j\rangle_b |\widetilde{\lambda_j}\rangle_c \frac{C}{\widetilde{\lambda_j}} |1\rangle_a \tag{38}$$

Note that term in front of the sum is just a factor to normalize the state. Let's 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

$$|\Psi_4\rangle = D \sum_{j=0}^{N-1} \frac{C}{\widetilde{\lambda_j}} b_j |u_j\rangle_b |\widetilde{\lambda_j}\rangle_c |1\rangle_a \tag{39}$$

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

$$|x\rangle = \sum_{i=0}^{N-1} \lambda_i^{-1} b_j |u_j\rangle \tag{40}$$

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 do with the states in the b-register and c-register being entangled with each other. Thus, we cannot convert the b-register into a $|0\rangle$ / $|1\rangle$ measurement.

### *D. Inverse Quantum Phase estimation.*

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

$$\begin{aligned} |\Psi_5\rangle &= \frac{1}{\sqrt{\sum_{j=0}^{N-1}\left|\frac{b_j C}{\lambda_j}\right|^2}} \sum_{j=0}^{N-1} \frac{C}{\lambda_j} b_j |u_j\rangle_b |0\rangle_c^{\otimes n} |1\rangle_a \\ &= \frac{C}{\sqrt{\sum_{j=0}^{2^{n_b}-1}\left|\frac{b_j C}{\lambda_j}\right|^2}} |x\rangle_b |0\rangle_c^{\otimes n} |1\rangle_a \end{aligned} \tag{41}$$

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 measured correctly.

We can furthermore simplify the expression 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 it to

$$\begin{aligned} |\Psi_5\rangle &= \frac{1}{\sqrt{\sum_{j=0}^{N-1}\left|\frac{b_j}{\lambda_j}\right|^2}} |x\rangle_b |0\rangle_c^{\otimes n} |1\rangle_a \\ &= |x\rangle_b |0\rangle_c^{\otimes n} |1\rangle_a \end{aligned} \tag{42}$$

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 omit 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

$$\langle x| M |x\rangle \tag{43}$$

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. Easy Example

In this section, we will go through an easy numerical example of the HHL algorithm. The quantum circuit that we will be using contains 4 qubits. The scheme of the quantum circuit is shown Fig. 2. Firstly, we will have to set some preliminary values to make our calculations easier. Then, we will go through all 5 phases of the HHL algorithm [2].

### *A. Preliminaries*

We are given the following matrix $A$ and the vector $\vec{b}$

$$A = \begin{pmatrix} 1 & -\frac{1}{3} \\ -\frac{1}{3} & 1 \end{pmatrix} \tag{44}$$

$$\vec{b} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{45}$$

If we calculate the solution classically we will obtain the solution vector

$$\vec{x} = \begin{pmatrix} \frac{3}{8} \\ \frac{9}{8} \end{pmatrix} \tag{46}$$

The HHL algorithm doesn't solve for the whole solution vector $\vec{x}$, as we are only interested in some estimation value $\langle x| M |x\rangle$. To compare the results of both methods we will take the probability ratio of the entries of our solution vector $\vec{x}$ and $|x\rangle$. To calculate the probability ratio, we have to divide by the square of each entry of $\vec{x}$. Thus, we obtain a ratio of

$$\frac{|x_0|^2}{|x_1|^2} = \frac{\frac{9}{64}}{\frac{81}{64}} = \frac{1}{9} \tag{47}$$

where $x_0$ and $x_1$ are the entries of $\vec{x}$.

To make our calculations, we will cheat by determining the eigenvectors and eigenvalues beforehand. Normally, the QPE would calculate these numbers, but as the main focus of this paper is the HHL algorithm, we will not calculate the eigenvalues and eigenvector manually through QPE. The eigenvectors of the matrix $A$ are

$$\vec{u_0} = \begin{pmatrix} \frac{-1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} \tag{48}$$

$$\vec{u_1} = \begin{pmatrix} \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \tag{49}$$

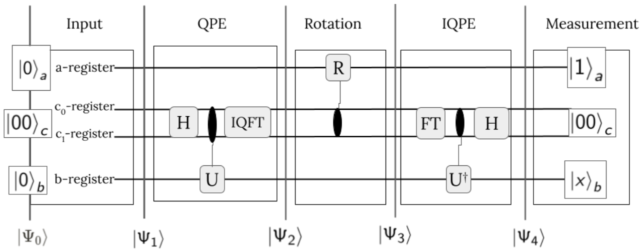

Fig. 2. 4-Qubit Circuit

These can be represented through amplitude encoding in our quantum circuit

$$|u_0\rangle = \frac{-1}{\sqrt{2}}|0\rangle + \frac{-1}{\sqrt{2}}|1\rangle \tag{50}$$

$$|u_1\rangle = \frac{-1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{51}$$

Additionally, we also have to determine the eigenvalues. The eigenvalues $\lambda_j$ of the matrix $A$ are

$$\lambda_0 = \frac{2}{3} \tag{52}$$

$$\lambda_1 = \frac{4}{3} \tag{53}$$

We then have to encode the eigenvalues into a quantum state $|\widetilde{\lambda_j}\rangle$. In the encoding scheme $\widetilde{\lambda_j} = N_q \lambda_j t/2\pi)$, we can freely choose a $t$ such that we obtain an easy encoding of the eigenvalues. Here $N_q$ refers to the number of qubits in our system, which means $N_q = 4$. If we choose $t = 3\pi/4$, then we will obtain

$$\widetilde{\lambda_0} = \frac{4 * \frac{2}{3} * \frac{3\pi}{4}}{2\pi} = \frac{4 * 2 * 3\pi}{3 * 4 * 2\pi} = 1 \tag{54}$$

$$\widetilde{\lambda_1} = \frac{4 * \frac{4}{3} * \frac{3\pi}{4}}{2\pi} = \frac{4 * 4 * 3\pi}{3 * 4 * 2\pi} = 2 \tag{55}$$

which can be encoded through basis encoding into the quantum circuit. Thus, we are left with these states

$$|\widetilde{\lambda_0}\rangle = |01\rangle \tag{56}$$

$$|\widetilde{\lambda_1}\rangle = |10\rangle \tag{57}$$

Now we can start with our procedure.

### B. Numerical walkthrough

In Fig. 2 we can see that we have 1 qubit each for the a-register and b-register, corresponding to the ancilla bit and the input $\vec{b}$. The c-register has 2 qubits, as we have to store the two eigenvalues $\lambda_1$, $\lambda_2$ we calculated just now. Our starting state looks like this

$$|\Psi_0\rangle = |0\rangle_b \;\; |00\rangle_c \;\; |0\rangle_a = |0000\rangle \tag{58}$$

all registers being initalized with the zero state.

*1) State preparation:* Here we have to encode our vector $\vec{b}$ into the quantum state $|b\rangle$, using amplitude encoding. We obtain

$$\vec{b} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \Leftrightarrow |b\rangle = 0\,|0\rangle + 1\,|1\rangle = |1\rangle \tag{59}$$

which makes sense as the $|1\rangle$ state is defined as the $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ vector. After loading $|b\rangle$ into our quantum circuit, the state of the circuit is

$$|\Psi_1\rangle = |1\rangle_b \;\; |00\rangle_c \;\; |0\rangle_a = |1000\rangle \tag{60}$$

*2) Quantum Phase Estimation:* The QPE determines the eigenvalues $\lambda_j$ of the $A$ matrix. These are stored in the c-register. The b-register now stores $|b\rangle$ in the eigenbasis state of $A$ leading us to the following state

$$|\Psi_2\rangle = |b\rangle_b \, |\widetilde{\lambda}\rangle_c \, |0\rangle_a \tag{61}$$

$$|\Psi_2\rangle = \left( -\frac{1}{\sqrt{2}} |u_0\rangle_b |01\rangle_c + \frac{1}{\sqrt{2}} |u_1\rangle_b |10\rangle_c \right) |0\rangle_a \tag{62}$$

*3) Inversion of the eigenvalues:* Now we have to invert the eigenvalues. This is achieved by rotating the a-registers by the eigenvalues in the c-register. This leaves us in this state

$$\begin{aligned} |\Psi_3\rangle &= \sum_{j=0}^{2^1-1} b_j |u_j\rangle |\widetilde{\lambda}_j\rangle \left( \sqrt{1 - \frac{C^2}{\widetilde{\lambda}_j^2}}\, |0\rangle + \frac{C}{\widetilde{\lambda}_j} |1\rangle \right) \\ &= -\frac{1}{\sqrt{2}} |u_0\rangle |01\rangle \left( \sqrt{1 - \frac{1}{1^2}}\, |0\rangle + \frac{1}{1} |1\rangle \right) \\ &\quad + \frac{1}{\sqrt{2}} |u_1\rangle |10\rangle \left( \sqrt{1 - \frac{1}{2^2}}\, |0\rangle + \frac{1}{2} |1\rangle \right) \end{aligned} \tag{63}$$

The ancilla bit is in a superposition and can either collapse to $|0\rangle$ or $|1\rangle$. We assume that our rotation was successful and we measure $|1\rangle$ in the ancilla qubit, then

$$\begin{aligned} |\Psi_3\rangle &= \sqrt{\frac{8}{5}} \left( \frac{1}{\widetilde{\lambda_0}} * -\frac{1}{\sqrt{2}} |u_0\rangle_b |01\rangle_c + \frac{1}{\widetilde{\lambda_1}} * \frac{1}{\sqrt{2}} |u_1\rangle_b |10\rangle_c \right) |1\rangle_a \\ &= \sqrt{\frac{8}{5}} \left( -\frac{1}{\sqrt{2}} |u_0\rangle_b |01\rangle_c + \frac{1}{2\sqrt{2}} |u_1\rangle_b |10\rangle_c \right) |1\rangle_a \end{aligned} \tag{64}$$

Note that the b-register and c-register are entangled. We have to unentangle the registers.

### C. Inverse Quantum Phase Estimation

We now backtrack all the steps in the QPE through IQPE. The c-register is reset to the zero state again and the b-register contains our solution state $|x\rangle$. The state of the circuit looks like this

$$|\Psi_4\rangle = |x\rangle_b \, |00\rangle_c \, |1\rangle_a \tag{65}$$

$$|\Psi_4\rangle = \frac{1}{2}\sqrt{\frac{2}{5}} \left( |0\rangle + 3\,|1\rangle \right) |00\rangle_b \, |1\rangle_a \tag{66}$$

### D. Measurement

Now we have to measure the solution state $|x\rangle$. In the beginning, we decided to take the probability ratio as our solution. To get the probability of $|u_0\rangle$ and $|u_1\rangle$ we have to square their coefficients

$$
\begin{aligned}
c_0 &= \left|\frac{1}{2}\sqrt{\frac{2}{5}} * 1\right|^2 = \frac{1}{20} \\
c_1 &= \left|\frac{1}{2}\sqrt{\frac{2}{5}} * 3\right|^2 = \frac{9}{20}
\end{aligned}
\tag{67}
$$

As expected our ratio is

$$
\frac{\frac{1}{20}}{\frac{9}{20}} = \frac{1}{9} \tag{68}
$$

## VI. Evaluation

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

The most commonly known method to solve a system of linear equations is Gaussian Elimination. The time complexity of the Gaussian is $\mathcal{O}(N^3)$, being drastically slower than other classical methods. As we are only interested in an estimate of an expectation value $\langle x| M |x\rangle$, we can take a look at similar classical methods, considering the 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 [1].

### A. Conjugate Gradient Descent

The classical conjugate gradient descent algorithm is a 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 toward the solution of the linear system. This will provide us with a suitable benchmark for analyzing the efficiency of the HHL algorithm since it also estimates a solution vector. Not only is it one of the fastest classical algorithms, but also has a very similar constraint set and calculates similar results (eg. $\vec{x}^\dagger M \vec{x}$).

### B. Comparison of Time Complexity

TABLE I
Time complexity comparison

| Conjugate Gradient 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 achieves a time complexity of $\mathcal{O}(N)$, whereas the HHL algorithm runs in $\mathcal{O}(log(N))$.

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

- $\epsilon$ is the accuracy
- $s$ is the s-sparseness
- $\kappa$ is the condition number

Firstly, $\epsilon$ describes the accuracy of our desired solution. The sparseness $s$ 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 $\kappa$ describes how sensitive the output of a function is at the error of the input. That means, a function is well-conditioned if the output of a function does not change by much if the errors in the input are big. These two factors are related to the QPE phase since a unitary $e^{iAt}$ has to be generated. This step heavily relies on the sparsity $s$ and conditioning $\kappa$ 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 omit our speedup of $\mathcal{O}(log(N))$.

We can observe, that the sparsity $s$ and condition number $\kappa$ are quadratic in the HHL algorithm, whereas in the conjugate gradient descent algorithm, both factors operate linearly. Furthermore, the accuracy $\epsilon$ is exponentially worse in the HHL algorithm. Thus, in terms of sparsity, accuracy and condition number, the HHL algorithm has a worse runtime than the conjugate gradient descent algorithm. If these prefactors were not worse than in the classical solution, this would have bizarre implications. We are able to show that if the dependency of these factors is better or equal as in the classical solution, we could solve *NP-Complete* problems exponentially faster on quantum computers, which seems unlikely. All in all, these constraints are important to consider when choosing applications, as they have a great effect on the runtime [5].

### C. Resource requirements

The resource requirements to implement the HHL algorithm are substantial. Looking at the structure of the algorithm it has a very similar structure to Shor's algorithm, having QPE as its main routine. Hence, 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 dropped many times over the past couple of years.

## VII. Applications and Variations

In this section, we will take a look at the future outlooks of the HHL algorithm and summarize our results. 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 perspectives, that the HHL algorithm offers in the field of quantum computing.

### A. Applications

The main problem with the HHL algorithm is, that it does not output the full solution vector, but rather the solution 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 each other. Thus, systems can be modeled as a graph which corresponds to large sparse matrices. By being able to calculate the inverse of a matrix $A^{-1}$ efficiently, we can deduct many network properties, such as resistance. This can help us to further optimize the electrical power system. Conventionally, we would have to rely on iterative algorithms, which are as we saw earlier, exponentially slower in this specific case [6].

*2) Machine Learning (Least-Square Estimation):* In the field of machine learning the HHL algorithm can also massively improve calculations. The HHL algorithm also plays a significant role in the field of machine learning, as it can improve computationally intensive calculations. For example in least-square estimation, the HHL algorithm can be used to determine estimates of inverse matrices. This can be very helpful in data fitting tasks, enabling efficient parameter estimation and model optimization [7].

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 multiple times at failure. Additionally, this increases the reliability of the algorithm [8].

*2) QRAM:* Quantum Random Access Memory (QRAM) is the quantum computer equivalent of the classical ram in a computer. It enables 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 [9].

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 of 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 a 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 useful 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 detection:* 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 fraudulent activities, which can be accelerated by the HHL algorithm.

## VIII. CONCLUSION

In conclusion, the HHL algorithm offers a significant improvement over its classical counterpart, considering its constraints. Sparsity, a low condition number and error are needed for the HHL algorithm to work efficiently. Additionally, ways to load the input vector faster, for example with Qram, are needed. In the end, we observed, that the sparsity and conditioning of the problem matrices $A$, have a comparable dependency on the runtime, but the error dependency is exponentially worse.

Despite these constraints, the HHL algorithm has made significant contributions, especially in the field of quantum machine learning. Since there are no ground-breaking applications so far, like Shor's algorithm that has the ability to break the RSA encryption, further use cases need to be found. Nonetheless, the ongoing discussions and research on the algorithm enable us to discover new optimizations and applications. The advances in quantum algorithms, including the HHL algorithm, have laid the foundation for future developments in the field. Because of the 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 small quantum computers with high fidelity, more advancements, as better error correction and scaling are needed to unlock the full potential.

# References

[1] A. Harrow, A. Hassidim, and S. Lloyd, ”Quantum algorithm for linear systems of equations,” Phys. Rev. Lett. 103, 150502, 2009. [Online]. Available: https://arxiv.org/abs/0811.3171

[2] Qiskit Community, ”Solving Linear Systems of Equations using HHL and its Qiskit Implementation”, [Online]. Available: https://learn.qiskit.org/course/ch-applications/solving-linear-systems-of-equations-using-hhl-and-its-qiskit-implementation, Accessed June 30, 2023.

[3] D. Dervovic, M. Herbster, P. Mountney, S. Severini, N. Usher, L. Wossnig, ”Quantum linear systems algorithms: a primer”, 2018. [Online]. Available: https://arxiv.org/abs/1802.08227

[4] H. Morrell Jr, A. Zaman, H. Wong, ”Step-by-Step HHL Algorithm Walkthrough to Enhance the Understanding of Critical Quantum Computing Concepts”, 2021. [Online]. Available: https://arxiv.org/abs/2108.09004

[5] S. Aaronson, ”Read the fine print”, Nature Phys 11, 291-293, 2015. [Online]. Available: https://inspirehep.net/literature/1834407

[6] G. Wang, ”Efficient Quantum Algorithms for Analyzing Large Sparse Electrical Networks”, Quantum Information & Computation 17 (11 & 12): 987-1026, 2017. [Online]. Available: https://arxiv.org/abs/1311.1851

[7] N. Wiebe, D. Braun, S. Lloyd, ””Quantum Data Fitting”, Phys. Rev. Lett. 109, 050505, 2012. [Online]. Available: https://arxiv.org/abs/1204.5242

[8] D. V. Babukhin, ”Harrow-Hassidim-Lloyd algorithm without ancilla postselection”, Dukhov Research Institute of Automatics (VNIIA), 127055 Moscow, Russia, 2022. [Online]. Available: https://arxiv.org/abs/2208.02200

[9] Quantum random access memory, Vittorio Giovannetti1, Seth Lloyd2, Lorenzo Maccone3, V. Giovannetti, S. Lloyd, L. Maccone, Phys. Rev. Lett. 100, 160501, 2008. [Online]. Available: https://arxiv.org/abs/0708.1879