UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

On Quantum Solvers for Linear Algebraic Systems

Diploma Thesis

Nikos Papagiannis

Supervisor: Manolis Vavalis

July 2023

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

On Quantum Solvers for Linear Algebraic Systems

Diploma Thesis

Nikos Papagiannis

Supervisor: Manolis Vavalis

July 2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Κβαντικοί Επιλυτές Γραμμικών Αλγεβρικών Συστημάτων

Διπλωματική Εργασία

Νίκος Παπαγιάννης

Επιβλέπων/πουσα: Μανόλης Βάβαλης

Ιούλιος 2023

Approved by the Examination Committee:

Supervisor **Manolis Vavalis**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Christos Antonopoulos**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Elias Houstis**
Professor Emeritus, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

I am grateful to my supervisor Professor Vavalis who has been eager to help me and contribute to this thesis whenever it has been necessary. The completion of this thesis would also not have been possible without the support of my family, so I would like to express my deepest gratitude to them.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Nikos Papagiannis

Diploma Thesis

On Quantum Solvers for Linear Algebraic Systems

Nikos Papagiannis

Abstract

Solving large-scale linear systems is an integral part of many scientific disciplines, such as machine learning or circuit analysis. Problems of this kind take as input a matrix A and a vector $\vec{b}$ and aim to find a vector $\vec{x}$ such that $A\vec{x} = \vec{b}$. Classical linear solvers have a polynomial time complexity and it seems impossible to further improve their performance. This fact has directed scientific research to the study of corresponding quantum algorithms, which promise even exponential acceleration compared to existing methods. In this thesis I present and analyze some of the most important results of this research. In particular, I present a method which relies on Grover's algorithm, the HHL and WZP algorithms based on the eigendecomposition of the matrix A, the row and column iteration methods, as well as two hybrid algorithms that require the cooperation of a classical and a quantum computer, namely an algorithm that uses random walks and the VQLS algorithm. The latter is also examined from an experimental standpoint using the Qiskit open-source framework and a suitable quantum circuit is proposed which can enhance its performance.

Keywords:

quantum computing, linear systems, Grover's algorithm, HHL, WZP, row and column iteration methods, random walks, VQLS, Qiskit

Διπλωματική Εργασία

Κβαντικοί Επιλυτές Γραμμικών Αλγεβρικών Συστημάτων

Νίκος Παπαγιάννης

Περίληψη

Η επίλυση γραμμικών συστημάτων μεγάλης κλίμακας είναι αναπόσπαστο κομμάτι πολλών επιστημονικών κλάδων, όπως η μηχανική μάθηση ή η ανάλυση κυκλωμάτων. Τα προβλήματα αυτού του είδους λαμβάνουν ως δεδομένα εισόδου έναν πίνακα A και ένα διάνυσμα $\vec{b}$ και στοχεύουν στην εύρεση ενός διανύσματος $\vec{x}$ τέτοιου ώστε $A\vec{x} = \vec{b}$. Οι κλασικοί επιλυτές γραμμικών συστημάτων έχουν πολυωνυμική χρονική πολυπλοκότητα και φαίνεται πως δεν μπορούν να βελτιώσουν περαιτέρω την απόδοση τους. Το γεγονός αυτό έχει στρέψει την επιστημονική έρευνα στη μελέτη αντίστοιχων κβαντικών αλγορίθμων, οι οποίοι υπόσχονται μέχρι και εκθετική επιτάχυνση σε σχέση με τις υπάρχουσες μεθόδους. Στα πλαίσια αυτής της διπλωματικής εργασίας αναλύω μερικά από τα σημαντικότερα αποτελέσματα αυτής της έρευνας. Συγκεκριμένα παρουσιάζω μια μέθοδο βασισμένη στον αλγόριθμο του Grover, τους αλγορίθμους HHL και WZP που βασίζονται στην ιδιοδιάσπαση του πίνακα A, τις επαναληπτικές μεθόδους γραμμής και στήλης, καθώς και δυο υβριδικούς αλγορίθμους οι οποίοι προϋποθέτουν τη συνεργασία ενός κλασικού και ενός κβαντικού υπολογιστή: έναν αλγόριθμο που χρησιμοποιεί τυχαίους περιπάτους και τον αλγόριθμο VQLS. Ο τελευταίος εξετάζεται επίσης από πειραματική σκοπιά χρησιμοποιώντας το λογισμικό ανοιχτού κώδικα Qiskit και προτείνεται ένα κατάλληλο κβαντικό κύκλωμα που μπορεί να βελτιώσει την απόδοσή του.

Λέξεις-κλειδιά:

κβαντική υπολογιστική, γραμμικά συστήματα, αλγόριθμος του Grover, HHL, WZP, επαναληπτικές μέθοδοι γραμμής και στήλης, τυχαίοι περίπατοι, VQLS, Qiskit

Table of contents

List of figures

Chapter 1

Introduction and Motivation

1.1 Presentation of the subject

Linear algebra is the field of mathematics which deals with sets of linear equations of the form:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases}$$

where x_i are the unknown variables that need to be calculated, whereas the elements a_{ij} and b_i are known in advance. In linear algebra terminology, an equivalent representation for such linear sets of equations is $A\vec{x} = \vec{b}$ where A is a matrix which contains all the a_{ij}, while $\vec{b}$ and $\vec{x}$ are two vectors which contain the elements b_i and x_j respectively:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_m \end{bmatrix}, \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix}.$$

A special case of linear sets of equations concerns an equal number of equations and unknown variables, thus $m = n$. The algorithms which are presented in this thesis mainly focus on this case and the goal of the problem that they are trying to solve, which is commonly referred to as Linear Systems Problem (LSP), is to find the solution vector $\vec{x}$ for which $\vec{x} = A^{-1}\vec{b}$, where A^{-1} is the inverse of the matrix A.

It is common knowledge that linear algebra and linear equations play a significant role in almost all fields of science and engineering. Some indicative such fields are the following:

- Cryptography - encoding and decoding of messages
- Analysis of networks and graphs
- Circuit analysis
- Machine learning and Artificial Intelligence - applications for data classification, clustering, regression and so on
- Ranking algorithms in search engines, like Google
- Analysis of economic models
- Statistics

This list could be extended with a lot more examples, as the applications which illustrate the importance of linear algebra are countless. In fact, linear sets of equations either constitute the core of such applications or they are a subroutine of larger algorithms. In either case they often include enormous amounts of data and as a result, the value of n increases arbitrarily. This has mainly to do with the fact that nowadays data are produced in large quantities and need to be collected and analyzed. The question which arises is how fast these linear systems can be solved. This question is crucial if we consider their size. There is a variety of classical methods for the LSP with different time complexities:

- Methods like Cholesky decomposition and Gaussian elimination, which are suitable for dense matrices, have a time complexity of $\mathrm{O}(n^3)$.
- The conjugate gradient descent which is used for sparse matrices, namely for matrices which have at most s non-zero elements per row with $s \ll n$, has a time complexity of $\mathrm{O}(n\sqrt{\kappa})$. Here κ is the condition number of the matrix A and it increases when A tends to become non-invertible. Its value varies from linear system to linear system - it may have a logarithmic, polynomial or even exponential dependence on n.

In general, all these classical linear solvers are polynomial to n. Further improving their performance in a classical computer seems impossible. For this reason, in the past few years, researchers have turned their attention to algorithms which can be implemented in quantum

computers and can solve the LSP fast and efficiently. Quantum computing is a fascinating scientific research and development area still at its early stage of development. Currently, there are only intermediate-scale quantum computers which are also noisy. At the moment this thesis is written, the largest quantum computer globally, which was developed by IBM, consists of 433 qubits (the counterpart of classical bits for quantum computers). Although the company will probably surpass the 1000 qubit mark soon [6], complete quantum systems with millions of qubits are definitely not expected to be developed in the short run. Despite that, the unprecedented potential that quantum computing offers has been widely accepted mainly through the study of several R&D use cases in disciplines such as financial technology, medicine, and data science. In particular, this study has shown that quantum computers can perform a lot of calculations much faster than classical computers by exploiting certain principles and properties of quantum mechanics. For example, they can factor large numbers much more quickly than classical computers [7], which is an important task in cryptography. Taking this into consideration, it is evident that quantum computing is a promising field that has the potential to revolutionize many areas of science and technology.

The objective of this thesis is first to survey the peer review literature in order to examine if quantum computing is suitable for numerical linear algebra and then to propose and discuss certain paths to further elucidate the issues raised. The thesis mainly discusses this subject from a theoretical standpoint, however, it also proceeds to the analysis of a few practical experiments using Qiskit, an open-source SDK for working with quantum computers. There are several research projects in that direction, even so, this research area has yet to attract considerable attention and there are definitely a lot of ideas to reflect on.

1.2 Thesis Organization

The rest of this thesis is organized as follows:

- In Chapter 2 I present the necessary theoretical background of quantum mechanics and quantum computing and mainly focus on the fundamental concepts which are related to the motivation of my study.

- In Chapter 3 I provide some insight into the existing results in the area of quantum numerical linear algebra and review these results.

- In Chapter 4 I discuss my experimental results on a hybrid quantum-classical linear solver and propose a quantum circuit which enhances the performance of the algorithm.
- In Chapter 5 I summarize the contents of this thesis, make some concluding remarks and draw some conclusions.

Chapter 2

Background

2.1 Introduction to Quantum Mechanics

Quantum mechanics is an important field in Physics which examines the nature from the perspective of elementary particles - atoms or subatomic particles. The preconceived classical physics theory only focuses on macroscopic phenomena and deals with subjects such as Mechanics and Thermodynamics, so quantum mechanics is complementary to it, as it tries to explain the microscopic aspects of nature which were previously unknown.

The main feature of quantum mechanics is the fact that quantities like energy and momentum are ”quantized”, namely they can only take discrete values. Another fundamental concept of quantum mechanics is wave–particle duality according to which quantum entities cannot be described merely as particles or waves because they have characteristics of both particles and waves.

The exact behavior of quantum systems is impossible to predict, as Heisenberg's uncertainty principle confirms. According to it, we cannot predict simultaneously the exact position and momentum of a quantum particle even under special experimental conditions. Mathematically this is expressed with the inequality:

$$\sigma_x \cdot \sigma_p \geq \frac{\hbar}{2}$$

where σ_x is the standard deviation of the position, σ_p is the standard deviation of the momentum and $\hbar$ is the reduced Planck constant ($\hbar = \frac{h}{2\pi}$).

For this reason, the study of quantum systems is based on probabilistic models. These models rely on wave functions which are mathematical expressions that assign probability

amplitudes to every point in space that a quantum entity can be found. They are usually denoted by the Greek letter ψ. The Born rule [8] states that the square of the probability amplitudes of a wave function are proportional to the probability density of a quantum system being in a specific quantum state. As a consequence, the set of all the available probability amplitudes can form a probability density function (PDF) which estimates how likely it is to find a quantum particle in a specific position.

The wave function – and by extension the probability amplitudes – of a quantum system change over time. The Schrödinger equation is a differential equation which approaches these changes:

$$i\hbar\frac{\partial}{\partial t}\left|\psi(t)\right\rangle = \hat{H}\left|\psi(t)\right\rangle \tag{2.1}$$

In this equation, $\hbar$ is the reduced Planck constant, $\hat{H}$ is a Hamiltonian operator, namely an operator that expresses the total energy of a quantum system, and $|\psi(t)\rangle$ is a wave function written with bra-ket notation (more about it in Section 2.2). If $\hat{H}$ is assumed constant, the solution of the equation is:

$$\left|\psi(t)\right\rangle = e^{\frac{-i\hat{H}t}{\hbar}}\left|\psi(0)\right\rangle \tag{2.2}$$

The Schrödinger equation and its solution is the most prominent result of quantum mechanics and has played a significant role in the development of the field. In fact it is considered by physicists the conceptual counterpart of Newton's second law in classical Physics.

2.2 Quantum states and bra-ket notation

Quantum states, which occur from the corresponding wave functions, can be described with state vectors in a Hilbert space [9] that express the polarization of the quantum particles. These vectors can possibly be written with typical vector notation, but quantum computing conventionally uses the bra-ket notation which was introduced by Dirac. As its name suggests, bra-ket deploys two types of symbols, bra and ket. Ket is a column vector $|x\rangle$ and on the other hand bra is a row vector $\langle x|$ which is the conjugate transpose of $|x\rangle$.

The typical Euclidean vector space requires a set of orthonormal vectors to form its basis. The same applies for the Hilbert space described here – for example a two dimensional complex Hilbert space needs two basis vectors, namely $|0\rangle$ and $|1\rangle$. The equivalent representation

of these vectors with the conventional notation is:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Respectively in the case of an arbitrary quantum state $a|0\rangle + b|1\rangle$ with $a, b \in \mathbb{C}$, the equivalent vector is:

$$a|0\rangle + b|1\rangle = a \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} + b \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}.$$

A Hilbert space is an inner product space, so the operation $\langle x| |y\rangle \equiv \langle x|y\rangle$ between two quantum states is defined:

$$\langle x|y\rangle = \begin{bmatrix} x_1^* & x_2^* & \cdots & x_n^* \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix} = x_1^* y_1 + x_2^* y_2 + \cdots x_n^* y_n$$

As the basis vectors are orthonormal, the following properties hold:

$$\langle 0|0\rangle = \langle 1|1\rangle = 1, \langle 0|1\rangle = \langle 1|0\rangle = 0$$

The outer product $|x\rangle \langle y|$ of two quantum states can also be expressed mathematically. In this case, the result of the operation is a matrix of the form:

$$|x\rangle \langle y| = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} \begin{bmatrix} y_1^* & y_2^* & \cdots & y_n^* \end{bmatrix} = \begin{bmatrix} x_1 y_1^* & x_1 y_2^* & \cdots & x_1 y_n^* \\ x_2 y_1^* & x_2 y_2^* & \cdots & x_2 y_n^* \\ \cdots & \cdots & \cdots & \cdots \\ x_n y_1^* & x_n y_2^* & \cdots & x_n y_n^* \end{bmatrix}$$

2.3 The qubit

For the rest of this thesis I will refer to the elementary quantum particles used in quantum computing tasks as quantum bits or qubits. A qubit is the unit of quantum information and is the counterpart of the bit in conventional computers. It corresponds to a changeable quantum state and in cases where this state is $|0\rangle$ or $|1\rangle$ it resembles the classical bit which stores binary information. However, in contradiction to a bit, a qubit can be in a special quantum state which is a ”superposition” of the basis states $|0\rangle$ and $|1\rangle$, that is a linear combination of these two states to form another valid state. In this case the qubit is in a state $|\psi\rangle =$

$a\,|0\rangle + b\,|1\rangle$ with $a, b \in \mathbb{C}$, where a, b are normalized according to the formula $|a|^2 + |b|^2 = 1$. This can also be expressed with the inner product $\langle\psi|\psi\rangle = 1$. The capability of the qubit to be in such quantum states enables parallelism which accelerates a lot of algorithms. This is the main reason why quantum computing has great prospects and is studied thoroughly by many researchers.

2.4 Measurement of the qubit

Although qubits can be in superposition, it is not feasible to extract their states and this is the main limitation of quantum computing. Measurements of a quantum system are possible, however, they can only output $|0\rangle$ or $|1\rangle$ and they lead to the collapse of the quantum state which cannot be recovered afterwards. More specifically, the measurement of a qubit $|\psi\rangle = a\,|0\rangle + b\,|1\rangle$ produces $|0\rangle$ with probability $\Pr(|0\rangle) = |a|^2$ and $|1\rangle$ with probability $\Pr(|1\rangle) = |b|^2$. Geometrically this means that the probability of a qubit being measured in the state $|0\rangle$ or $|1\rangle$ is equal to the projection of the state vector on the respective basis vector.

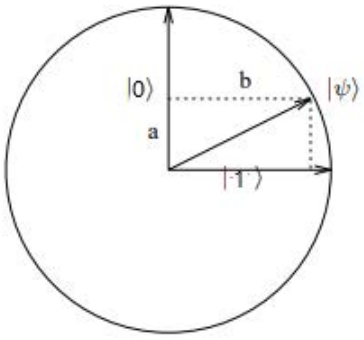

Figure 2.1: Geometrical 2D representation - measurement of a qubit

From the above it follows that despite the fact that a qubit can be in infinitely many states, only binary information can be obtained from it - similar to a classical bit. Furthermore, it should be noted that unknown quantum states cannot be cloned, which means that it is impossible to create copies of the state of a qubit and approximate it with multiple measurements.

2.5 Geometrical representation of quantum states

As stated above, the general state of a qubit is $a\,|0\rangle + b\,|1\rangle$ with $a, b \in \mathbf{C}$. This is translated into four independent real number variables ($\mathfrak{Re}(a)$, $\mathfrak{Im}(a)$, $\mathfrak{Re}(b)$, $\mathfrak{Im}(b)$), so the visual representation of quantum states does not seem possible. However, exploiting the fact that a phase shift of a quantum state does not change its physical properties, we find by transforming to polar coordinates:

$$|\psi\rangle = a\,|0\rangle + b\,|1\rangle \Rightarrow |\psi\rangle = r_a \cdot e^{i\phi_a} + r_b \cdot e^{i\phi_b}$$

which can alternatively be expressed as follows:

$$|\psi\rangle = r_a + r_b \cdot e^{i(\phi_b - \phi_a)} \Rightarrow |\psi\rangle = r_a + r_b \cdot e^{i\phi} \text{ where } \phi = \phi_b - \phi_a, \phi \in [0, 2\pi)\,.$$

This reduces the independent variables to three, so the quantum state can be represented in a three dimensional space. The variables can even be diminished to two, if we write the quantum state in the form:

$$|\psi\rangle = \cos\frac{\theta}{2} + \sin\frac{\theta}{2} \cdot e^{i\phi} \text{ where } \theta = 2\cos^{-1} r_a = 2\sin^{-1} r_b, \theta \in [0, \pi] \tag{2.3}$$

The final formula is used to describe the quantum state of a single qubit as a radius of a unit sphere called Bloch sphere, as the following figure illustrates:

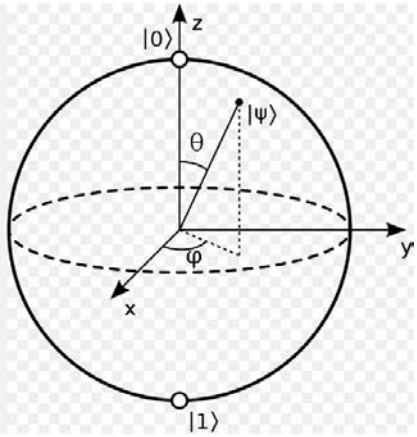

Figure 2.2: The Bloch sphere

2.6 Quantum systems of n qubits

A significant element that shows the power of quantum computing is the fact that a n-qubit system formulates a 2^n dimensional state space, so the number of its independent basis

states is exponential to the size of the system. On the other hand, a n-bit classical system generates a $2n$ dimensional vector space. This means that to create, for instance, a vector space of dimension 2^{20}, 2^{19} bits are required, whereas 20 qubits are sufficient.

The quantum state of a n-qubit system is equal to the tensor product of the quantum states of the individual qubits. The same applies for the basis of the state space of the quantum system, as it is calculated from the tensor product of the basis that corresponds to each qubit (which consists of $|0\rangle$ and $|1\rangle$). For example, the basis of a 2-qubit system is the set:

$$\{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\} \equiv \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$$

Similarly the quantum state of a 2-qubit system that stems from the tensor product of the quantum states of 2 qubits is:

$$(a_1 |0\rangle + b_1 |1\rangle) \otimes (a_2 |0\rangle + b_2 |1\rangle) = a_1a_2 |00\rangle + a_1b_2 |01\rangle + b_1a_2 |10\rangle + b_1b_2 |11\rangle$$

This can also be verified using typical vector notation and the properties of tensor products:

$$(a_1 |0\rangle + b_1 |1\rangle) \otimes (a_2 |0\rangle + b_2 |1\rangle) =$$

$$= \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \otimes \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} = \begin{bmatrix} a_1 \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} \\ b_1 \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} a_1a_2 \\ a_1b_2 \\ b_1a_2 \\ b_1b_2 \end{bmatrix} =$$

$$= a_1a_2 |00\rangle + a_1b_2 |01\rangle + b_1a_2 |10\rangle + b_1b_2 |11\rangle$$

Generally all quantum states that have the form $|\psi\rangle = x |00\rangle + y |01\rangle + z |10\rangle + w |11\rangle$ with $x, y, z, w \in \mathbf{C}$ are valid provided that the normalization $|x|^2 + |y|^2 + |z|^2 + |w|^2 = 1$ is true. However, it is not always possible to decompose a 2-qubit state into the states of its two qubits - This is equivalent to calculating the values of a_1, a_2, b_1, b_2 of the above formula. For example, decomposing the state $\frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle$ leads to the conclusion that $b_1b_2 = a_1a_2 \neq 0 \Rightarrow a_1 \neq 0, a_2 \neq 0, b_1 \neq 0, b_2 \neq 0$, while concurrently $a_1b_2 = a_2b_1 = 0$ is true which contradicts the previous statement.

The above observations are not only valid in the case of 2-qubit systems, but can easily be extended to an arbitrary number of qubits. For instance, the basis of a 4-qubit system is the set:

$$\{|0000\rangle, |0001\rangle, |0010\rangle, \cdots |1110\rangle, |1111\rangle\}$$

which for simplicity reasons is written in the form:

$$\{|0\rangle, |1\rangle, |2\rangle, \cdots |14\rangle, |15\rangle\} = \{|x\rangle \mid 0 \leq x \leq 2^4 - 1\}$$

2.7 Quantum gates and unitary operators

The 'base' of a classical computer are large circuits that include appropriate configurations of logic gates, like AND, OR , NOT. These gates take one or more input bits and produce an output bit. Similar to that, the functionality of a quantum computer relies on quantum circuits which consist of several quantum gates. These quantum gates, as well as the measurement of qubits, are the available resources that one has to solve a problem using quantum computing.

Quantum gates are not capable of changing the physical properties of the qubits. In fact, what they do is rotate the basis vectors of the state space and this way they modify the quantum state and affect the result of a possible measurement. A unitary operator corresponds to each quantum gate. This is a square matrix A which, when it is multiplied with the vector of the input quantum state of a quantum gate, produces the vector of the output quantum state. As this matrix is unitary, it needs to meet the requirement $AA^\dagger = A^\dagger A = I$. This property is very important, as it guarantees the preservation of the inner product of two quantum states when an operator A is applied to both of them:

$$\langle Ax|Ay\rangle = x^\dagger A^\dagger Ay = x^\dagger Iy = \langle x|y\rangle$$

This also affirms that a state vector $|x\rangle$ remains normalized after its transformation from an operator A:

$$\langle Ax|Ax\rangle = \langle x|x\rangle = 1$$

In addition, it should be noted all quantum gates are reversible, so their operators also need to be reversible, which can easily be confirmed:

$$A^\dagger A = AA^\dagger = I \Rightarrow A^{-1} = A^\dagger$$

The most simple quantum gates are the single qubit gates. They act on a qubit and lead to a change of its quantum state:

$$a\,|0\rangle + b\,|1\rangle \xrightarrow{A} a'\,|0\rangle + b'\,|1\rangle$$

or alternatively in matrix/vector notation:

$$\begin{bmatrix} a \\ b \end{bmatrix} \xrightarrow{A} \begin{bmatrix} a' \\ b' \end{bmatrix}$$

Therefore, the operator A that corresponds to a single qubit gate is a 2×2 matrix. Its first column defines how the 'coefficient' of $|0\rangle$ is modified - geometrically this represents the rotation of the basis vector $|0\rangle$. Similarly its second column defines how the 'coefficient' of $|1\rangle$ changes, which represents the rotation of the basis vector $|1\rangle$.

The most trivial gate is the identity gate I, which does not affect the quantum state of a qubit:

$$I = |0\rangle\langle 0| + |1\rangle\langle 1| = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \Longrightarrow |0\rangle \rightarrow |0\rangle, |1\rangle \rightarrow |1\rangle$$

The Pauli X, Y and Z gates cause a rotation of π radians around the x, y and z axes of the Bloch sphere respectively:

- The X gate is the counterpart of the NOT gate, as it transforms $|0\rangle$ to $|1\rangle$ and vice versa:

$$X = |0\rangle\langle 1| + |1\rangle\langle 0| = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \Longrightarrow |0\rangle \rightarrow |1\rangle, |1\rangle \rightarrow |0\rangle$$

- The Y gate maps $|0\rangle$ to $i|1\rangle$ and $|1\rangle$ to $-i|0\rangle$:

$$Y = -i|0\rangle\langle 1| + i|1\rangle\langle 0| = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \Longrightarrow |0\rangle \rightarrow i|1\rangle, |1\rangle \rightarrow -i|0\rangle$$

- The Z gate just flips the direction of $|1\rangle$:

$$Z = |0\rangle\langle 0| - |1\rangle\langle 1| = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \Longrightarrow |0\rangle \rightarrow |0\rangle, |1\rangle \rightarrow -|1\rangle$$

Another single qubit gate which plays a significant role in most quantum computing implementations is the Hadamard gate. When its input is the state $|0\rangle$ or $|1\rangle$, it creates a superposition where a measurement may produce $|0\rangle$ or $|1\rangle$ with equal probabilities. Basically this simulates a fair coin flip experiment. Geometrically the Hadamard gate first includes a $\frac{\pi}{2}$ radians rotation around the y axis and afterwards a π radians rotation around the x axis:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \Longrightarrow |0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \equiv |+\rangle, |1\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \equiv |-\rangle$$

The phase shift gates P are a family of quantum gates that do not affect the basis state $|0\rangle$, but shift the phase of the basis state $|1\rangle$ by an angle ϕ:

$$P(\phi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix} \Longrightarrow |0\rangle \rightarrow |0\rangle, |1\rangle \rightarrow e^{i\phi}|1\rangle$$

Some examples of these gates are:

- The S gate and its inverse:

$$S = P(\pi/2) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{bmatrix}, S^\dagger = P(-\pi/2) = \begin{bmatrix} 1 & 0 \\ 0 & e^{-i\pi/2} \end{bmatrix}$$

- The T gate and its inverse:

$$T = P(\pi/4) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}, T^\dagger = P(-\pi/4) = \begin{bmatrix} 1 & 0 \\ 0 & e^{-i\pi/4} \end{bmatrix}$$

- The Pauli Z gate which was presented above, as $Z = P(\pi)$.

Another category of single qubit quantum gates are the rotation gates which rotate the quantum state around the x, y and z axes of the Bloch sphere:

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

$$R_z(\theta) = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$$

To conclude the analysis of the single qubit gates, it is important to mention that the general formula of their operators is:

$$U(\theta, \lambda, \phi) = \begin{bmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\phi}\sin(\theta/2) & e^{i\lambda+i\phi}\cos(\theta/2) \end{bmatrix}$$

As a result, infinitely many gates can be created by tuning the parameters θ, λ, ϕ. All the above gate categories can also be described with this general operator. For instance, for the phase shift gates $P(\phi) = U(0, 0, \phi)$ is true.

Although this is not always the case, multiple qubit gate operators can often be expressed as the tensor product of single qubit gate operators. An example is the n-qubit Hadamard gate which stems from the tensor product of n single qubit Hadamard gates. It has a similar impact as the respective single qubit gate, as it creates an equal superposition of all the 2^n basis states of the n-qubit state space when its input qubits are all set to $|0\rangle$:

$$(H \otimes H \otimes \cdots \otimes H)\,|00\ldots0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \cdots \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) =$$

$$= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \tag{2.4}$$

An important two-qubit gate is the SWAP gate which swaps two qubits:

$$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Longrightarrow |00\rangle \rightarrow |00\rangle, |01\rangle \rightarrow |10\rangle, |10\rangle \rightarrow |01\rangle, |11\rangle \rightarrow |11\rangle$$

Another noteworthy two-qubit gate is the controlled NOT or CNOT gate. This gate performs an if-else operation: The first qubit plays the role of the control qubit and in case it is in the state $|1\rangle$, the second qubit flips its quantum state from $|0\rangle$ to $|1\rangle$ or vice versa, otherwise its state remains unchanged:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \Longrightarrow |00\rangle \rightarrow |00\rangle, |01\rangle \rightarrow |01\rangle, |10\rangle \rightarrow |11\rangle, |11\rangle \rightarrow |10\rangle$$

The Toffoli or CCNOT gate is the same as the CNOT gate, but it acts on three qubits, so there are two control qubits and both of them need to be in the state $|1\rangle$ for the third qubit to flip its state:

$$CCNOT = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The Toffoli gate is universal for classical computing, as it can be used to implement both NOT and AND gates:

- NOT gate: $CCNOT\,|1, 1, x\rangle = |1, 1, \neg x\rangle$
- AND gate: $CCNOT\,|x, y, 0\rangle = |x, y, x \wedge y\rangle$

The following figure demonstrates the circuit symbols of the aforementioned multiple qubit gates. As far as the CNOT and Toffoli gates are concerned, the control qubits are distinguished because they have a dot mark:

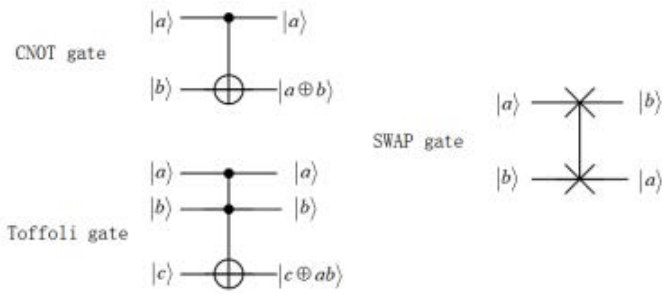

Figure 2.3: CNOT, Toffoli and SWAP gate circuit symbols

2.8 Relation of Hermitian and unitary matrices

In quantum computing there are two types of matrices that are widely used:

- Unitary matrices that were presented in the previous section, whose inverse is equal to their conjugate transpose: $AA^\dagger = A^\dagger A = I$
- Hermitian matrices, which are equal to their conjugate transpose: $H^\dagger = H$

An important property of both of these categories is that their matrices are diagonalizable, hence they can be expressed by the formula:

$$M = \sum_j \lambda_j |u_j\rangle \langle u_j|$$

where λ_j are the eigenvalues of the matrix M and u_j are the corresponding eigenvalues. This property reveals the form of the eigenvalues of Hermitian and unitary matrices. In particular, it can be easily be proved that:

- The eigenvalues of unitary matrices have a magnitude of 1, as $\lambda_j \lambda_j^* = 1 \; \forall j$. Therefore, they can alternatively be written in the form $e^{i\phi_j}$.
- The eigenvalues of Hermitian matrices are real numbers, as $\lambda_j = \lambda_j^* \; \forall j$.

The above observations lead to the conclusion that it is always possible to map a Hermitian matrix to a unitary matrix. This just requires retaining the eigenstates of the initial matrix and exponentiating its eigenvalues. Mathematically the relation between the two matrices is:

$$A = e^{iH}.$$

2.9 Universality of quantum gates

In classical computing there are two universal gates, NAND and NOR. They are called universal because they can construct any Boolean circuit without the need to use any other gates. The concept of universality exists in quantum computing as well. However, in this case there is no single quantum gate that can generate the rest of the gates, so universality refers to a set of quantum gates which can generate any quantum system independent from the number of qubits this system consists of.

The set of gates that achieves universality is not unique, but there are a few such sets like the following:

- The set consisting of the Toffoli and the Hadamard gates
- The set consisting of the rotation gates and the phase gates
- The set consisting of the Hadamard, CNOT, S and T gates. The first three gates are known as the Clifford set.

2.10 Implementation of functions and quantum parallelism

A function $f(x)$ of the form $f\colon x \rightarrow y = f(x)$ cannot be implemented in quantum computing similar to classical computing, because mapping x values to y values is not a unitary operation and often it is not even reversible. Therefore, to follow the basic principles of quantum computing, the implementation of functions $f(x)$ is based on "gate arrays" U_f which perform unitary transformations:

$$U_f : |x, y\rangle \rightarrow |x, y \oplus f(x)\rangle$$

where $\oplus$ is the bitwise xor operation.
Typically a gate array is a $(m+n)$ qubit operation - The first m qubits correspond to the size of the input x in bits and the last n qubits correspond to the size of the output $f(x)$ in bits. The latter are initialized to $|0\rangle$ so that the transformation that takes place is:

$$U_f : |x, 0\rangle \rightarrow |x, f(x)\rangle$$

This transformation can easily be reversed, as the inverse of a gate array is the gate array itself:

$$U_f U_f \, |x, y\rangle = |x, y \oplus f(x) \oplus f(x)\rangle = |x, y\rangle$$

The fundamental difference between gate arrays and functions implemented in classical computers is the fact that the former enable parallelism, as they can evaluate multiple input values x simultaneously. These values need to be expressed as an appropriate superposition in the first m qubits of the input of the gate array. U_f acts as a linear transformation, so a superposition of values x in the input will create a superposition of values $f(x)$ in the output. To illustrate that, an indicative example involves using qubits initialized to $|0\rangle$, creating an equal superposition of all basis states by applying the Hadamard gate and finally calculating the outputs of a function for all values between 0 and $2^m - 1$:

$$|0,0\rangle \xrightarrow{H} \frac{1}{\sqrt{2^m}} \sum_{x=0}^{2^m-1} |x,0\rangle \xrightarrow{U_f} \sum_{x=0}^{2^m-1} |x,f(x)\rangle$$

This example shows the strength of quantum parallelism, however, it should be noted that in practice only one value of $f(x)$ can be extracted from the resulting state, as a measurement leads to its collapse. It is not even possible to select an output value of interest, although there are algorithms which can amplify a specific value, so that it can be extracted with higher probability.

2.11 Inner products and the swap test

Although inner products of arbitrary quantum states $\langle x|y\rangle$ are used in many mathematical formulas of quantum computing, in practice it is impossible to accurately calculate them. However, it is important to approximate them, as these inner products express how similar quantum states are - two identical states produce an inner product equal to 1, while the inner product of orthogonal states is equal to 0. Apart from that, inner products could be used to extract an element of a state vector. For instance, assuming that we have a quantum state of the form:

$$|x\rangle = a_{00}\,|00\rangle + a_{01}\,|01\rangle + a_{10}\,|10\rangle + a_{11}\,|11\rangle$$

the element a_{00} could theoretically be computed with the operation:

$$\langle x|00\rangle = a_{00}\,\langle 00|00\rangle + a_{01}\,\langle 01|00\rangle + a_{10}\,\langle 10|00\rangle + a_{11}\,\langle 11|00\rangle = a_{00}$$

For the estimation of inner products a probabilistic implementation called swap test is used. The swap test is based on a quantum circuit which consists of three registers, two of them are the input quantum states whose inner product is approximated and the remaining register is the output qubit which is initialized to $|0\rangle$. The circuit includes three gates: a

Hadamard gate acting on the output qubit, a controlled SWAP gate acting on the input states - which swaps the states of these two states if the output qubit register is in the state $|1\rangle$ - and a second Hadamard gate. These gates are followed by a measurement of the output qubit, as the figure below shows:

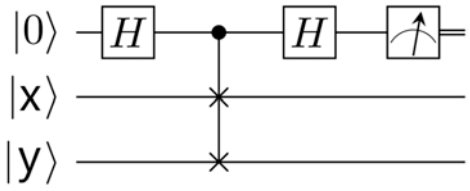

Figure 2.4: Swap Test

The transformation of the quantum state of the registers is expressed mathematically as follows:

$$|0,x,y\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}|0,x,y\rangle + \frac{1}{\sqrt{2}}|1,x,y\rangle \xrightarrow{C-SWAP} \frac{1}{\sqrt{2}}|0,x,y\rangle + \frac{1}{\sqrt{2}}|1,y,x\rangle \xrightarrow{H}$$

$$\xrightarrow{H} \frac{1}{2}|0,x,y\rangle + \frac{1}{2}|1,x,y\rangle + \frac{1}{2}|0,y,x\rangle - \frac{1}{2}|1,y,x\rangle = \frac{1}{2}|0\rangle\left(|x,y\rangle + |y,x\rangle\right) + \frac{1}{2}|1\rangle\left(|x,y\rangle - |y,x\rangle\right)$$

After the measurement occurs, the probabilities of the output qubit being in the state $|0\rangle$ or $|1\rangle$ can easily be calculated:

$$\Pr(|0\rangle) = \frac{1}{2}(\langle x,y| + \langle y,x|)\frac{1}{2}(|x,y\rangle + |y,x\rangle) = 2\cdot\frac{1}{4} + 2\cdot\frac{1}{4}|\langle x|y\rangle|^2 = \frac{1}{2} + \frac{1}{2}|\langle x|y\rangle|^2$$

$$\Pr(|1\rangle) = 1 - \Pr(|0\rangle) = \frac{1}{2} - \frac{1}{2}|\langle x|y\rangle|^2$$

The output of the circuit can be considered as a Bernoulli random variable which produces $|1\rangle$ with probability $\frac{1}{2} - \frac{1}{2}|\langle x|y\rangle|^2$. The results of multiple iterations of this experiment can be gathered in order to estimate the value of $|\langle x|y\rangle|$ based on their mean. It has been proved that to restrict the additive error of the estimation to a value ϵ, $\mathrm{O}(\frac{1}{\epsilon^2})$ such iterations are required.

2.12 The Hadamard test

Another quantum circuit which specializes in the calculation of inner products is the Hadamard test. It computes a value of the form $\mathfrak{Re}(\langle\psi|U|\psi\rangle)$, where U is a unitary operator and $|\psi\rangle$ is an arbitrary quantum state. It consists of Hadamard gates, a controlled U-operator and a measurement of the output qubit as the figure below illustrates:

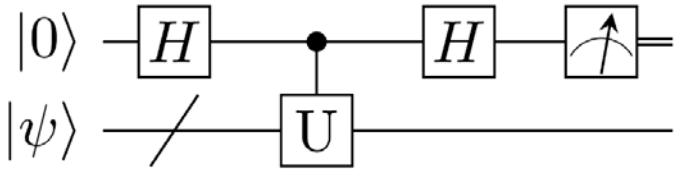

Figure 2.5: Hadamard Test

The input quantum state of the circuit evolves as follows:

$$|0\rangle \otimes |\psi\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |\psi\rangle \xrightarrow{C-U} \frac{1}{\sqrt{2}} |0\rangle \otimes |\psi\rangle + \frac{1}{\sqrt{2}} |0\rangle \otimes U |\psi\rangle \xrightarrow{H}$$

$$\xrightarrow{H} \frac{1}{2} |0\rangle |\psi\rangle + \frac{1}{2} |1\rangle |\psi\rangle + \frac{1}{2} |0\rangle U |\psi\rangle - \frac{1}{2} |1\rangle U |\psi\rangle = \frac{1}{2} |0\rangle \otimes (I+U) |\psi\rangle + \frac{1}{2} |1\rangle \otimes (I-U) |\psi\rangle$$

After the measurement occurs, the probabilities of taking $|0\rangle$ or $|1\rangle$ from the output qubit can be estimated:

$$\Pr(|0\rangle) = \frac{1}{4} \langle\psi| (I+U)^\dagger (I+U) |\psi\rangle = \frac{1}{4} \langle\psi| (I + U^\dagger + U + U^\dagger U) |\psi\rangle =$$

$$= \frac{1}{4}(2 + \langle\psi|U|\psi\rangle^* + \langle\psi|U|\psi\rangle) = \frac{1}{2}(1 + \mathfrak{Re}(\langle\psi|U|\psi\rangle))$$

$$\Pr(|1\rangle) = 1 - \Pr(|0\rangle) = \frac{1}{2}(1 - \mathfrak{Re}(\langle\psi|U|\psi\rangle))$$

Following the same concept as the one used for the swap test, the output of the Hadamard test circuit is a Bernoulli random variable which produces $|1\rangle$ with probability $\frac{1}{2}(1-\mathfrak{Re}(\langle\psi|U|\psi\rangle))$. Taking this into consideration, collecting multiple such outputs is sufficient to approximate $\mathfrak{Re}(\langle\psi|U|\psi\rangle)$.

2.13 Pure and mixed states, density operators

Quantum states are divided into two categories:

- Pure states, which are all the states that have been used in the previous sections and can be sufficiently represented with a vector $|x\rangle$ in braket notation. They can be defined precisely with no uncertainty even when they are complex superpositions of the basis states.

- Mixed states, which include some uncertainty about the quantum state. For this reason, these quantum states are a probabilistic combination of multiple pure states.

The density operators ρ are matrices which are designed to express both pure and mixed states:

- For pure states they are defined as the outer product of the quantum state:

$$\rho = |x\rangle \langle x| \tag{2.5}$$

- For mixed states that consist of a set of pure states $|x_j\rangle$ with probabilities of occurrence p_j ($\sum_j p_j = 1$), they are defined as the weighted sum of the outer products of these pure states:

$$\rho = \sum_j p_j |x_j\rangle \langle x_j| \tag{2.6}$$

Density operators are used to mathematically express how mixed quantum states change when unitary operators U are applied, as the formula below shows:

$$\rho = \sum_j p_j |x_j\rangle \langle x_j| \xrightarrow{U} \rho' = \sum_j p_j |Ux_j\rangle \langle Ux_j| = \sum_j p_j U |x_j\rangle \langle x_j| U^\dagger = U\rho U^\dagger \tag{2.7}$$

They have a lot of important properties which are worth mentioning:

1. Their trace, namely the sum of their diagonal elements, is 1:

$$Tr(\rho) = \sum_j \rho_{jj} = 1$$

.

2. Their diagonal elements ρ_{jj} express the probability of the occurrence of the corresponding pure state $|x_j\rangle$.

3. They are positive definite matrices. If we select an arbitrary state $|x\rangle$, we find:

$$\langle x|\rho|x\rangle = \sum_j p_j \langle x|x_j\rangle \langle x_j|x\rangle = \sum_j p_j |\langle x_j|x\rangle|^2 \geq 0$$

4. They are Hermitian matrices: $\rho = \rho^\dagger$

5. The trace of ρ^2 reveals the purity of a quantum state. In particular:

For pure states: $Tr(\rho^2) = 1$

For mixed states: $Tr(\rho^2) < 1$

2.14 Quantum Fourier Transform

In general, the Fourier transform is a mathematical procedure which converts a function from the time domain to the frequency domain. A special case of the Fourier transform is the discrete Fourier transform (DFT) which handles functions that consist of finite equally spaced samples and produces an output of the same size in the frequency domain. The counterpart of the DFT in quantum computing is the Quantum Fourier Transform (QFT). In comparison to the classical DFT, a special characteristic of the QFT is the fact that its input sequence always includes a total number of samples - quantum states - which is a power of 2. The formula of the QFT is the following:

$$|x\rangle \rightarrow \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{\frac{2\pi i x y}{2^n}} |y\rangle \tag{2.8}$$

In a n-qubit quantum system, the QFT transforms a basis state $|x\rangle$ to a superposition of all possible 2^n basis states where a measurement can produce any of them with equal probability.

An important property of the resulting state in (2.8) is that it is unentangled, thus it can be expressed as the tensor product of the states of single qubits. In particular, by writing x and y in binary form:

$$x = 2^{n-1}x_1 + 2^{n-2}x_2 + \cdots 2^1 x_{n-1} + 2^0 x_n$$

$$y = 2^{n-1}y_1 + 2^{n-2}y_2 + \cdots 2^1 y_{n-1} + 2^0 y_n$$

and by exploiting the equality below:

$$e^{\frac{2\pi i x y}{2^n}} |y_1 y_2 \cdots y_n\rangle = e^{2\pi i (0.x_n) y_1} |y_1\rangle \, e^{2\pi i (0.x_{n-1} x_n) y_2} |y_2\rangle \cdots e^{2\pi i (0.x_1 x_2 \cdots x_n)} |y_n\rangle \tag{2.9}$$

it is possible to factorize the state of (2.8) as follows:

$$\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{\frac{2\pi i x y}{2^n}} |y\rangle =$$

$$= \frac{1}{\sqrt{2^n}} (|0\rangle + e^{2\pi i (0.x_n) y_1} |1\rangle)(|0\rangle + e^{2\pi i (0.x_{n-1} x_n) y_2} |1\rangle) \cdots (|0\rangle + e^{2\pi i (0.x_1 x_2 \ldots x_n) y_n} |1\rangle) \tag{2.10}$$

With the help of (2.10) the QFT can be implemented in a quantum system which consists of Hadamard gates and phase shift gates, as the figure below illustrates:

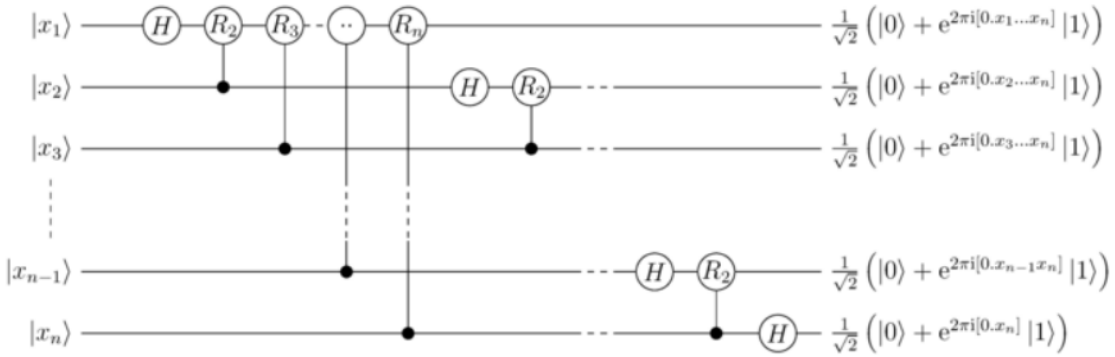

Figure 2.6: Implementation of the Quantum Fourier Transform

The R_k gates which are used in the above circuit are phase shift gates of the form:

$$R_k = P\left(\frac{2\pi}{2^k}\right) = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{bmatrix}$$

The inverse of the QFT can easily be implemented: the inverse of the Hadamard gates are the Hadamard gates themselves, whereas the inverse of the phase shift gates $P(\phi)$ are the phase shift gates $P(-\phi)$. This implementation is of vital importance, as it can take an input state which is a superposition of the form of (2.8) and can produce the corresponding value of the basis state $|x\rangle = |x_1 x_2 \cdots x_n\rangle$. A significant application of the QFT is presented in section 2.15.

2.15 Phase estimation

The term ”quantum phase estimation” is used to describe a problem which focuses on an unknown unitary operator U that has an eigenvector $|\psi\rangle$ with a corresponding eigenvalue $e^{2\pi i\phi}$ ($0 \leq \phi < 1$) and acts on n-qubit quantum systems. Although the form of U is undetermined, the problem assumes the availability of gates that perform controlled U^{2^k} operations ($k \in \mathbb{N}$). The purpose of this problem is to calculate the value of ϕ - and by extension the eigenvalue of U. Its solution is based on the following circuit:

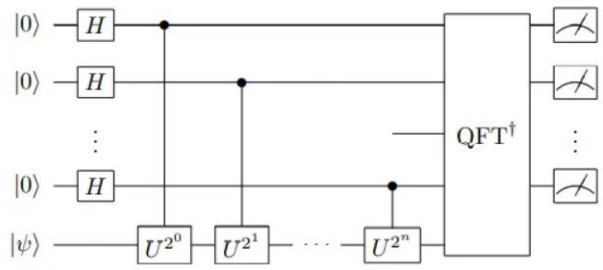

Figure 2.7: Phase Estimation Algorithm

According to known properties of eigenvectors and eigenvalues, the equality $U^{2^k}|\psi\rangle = (e^{2\pi i\phi})^k|\psi\rangle$ is true. Therefore, when a Hadamard and a controlled U^{2^k} operator are applied to one of the qubits of the circuit which are set to $|0\rangle$, the resulting state is:

$$|0\rangle|\psi\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle|\psi\rangle + |1\rangle|\psi\rangle) \xrightarrow{U^{2^k}}$$

$$\xrightarrow{U^{2^k}} \frac{1}{\sqrt{2}}(|0\rangle|\psi\rangle + |1\rangle U^{2^k}|\psi\rangle) = \frac{1}{\sqrt{2}}(|0\rangle|\psi\rangle + |1\rangle e^{2\pi i 2^k\phi}|\psi\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 2^k\phi}|1\rangle)\otimes|\psi\rangle \tag{2.11}$$

We can observe that the qubit register which contains $|\psi\rangle$ remains unchanged. Using (2.11) for all the qubits of the circuit, the subsequent state of the quantum system before the inverse QFT is applied is the following:

$$(|0\rangle + e^{2\pi i 2^{n-1}\phi}|1\rangle)(|0\rangle + e^{2\pi i 2^{n-2}\phi}|1\rangle)\cdots(|0\rangle + e^{2\pi i\phi}|1\rangle) = \sum_{i=0}^{2^n-1} e^{2\pi i\phi y}|y\rangle \tag{2.12}$$

As the quantum state of (2.12) resembles the one of (2.8), if ϕ can be written with $\leq n$ binary decimals ($\phi = 0.a_1a_2\cdots a_n$), the inverse QFT can produce the precise value of ϕ. On the other hand, in cases where this is not possible, it has been proved that the phase estimation algorithm outputs a best n-bits approximation of the value of ϕ with probability $\geq \frac{4}{\pi^2}$.

2.16 Grover's algorithm

One of the most well-known algorithms which attracted a lot of interest in the field of quantum computing is Grover's algorithm. This is an unstructured search algorithm, namely given a black box function f, its purpose is to find the unique x among a set of m possible values, so that $f(x)$ takes a specific value. Moreover, no information is known in advance about the structure of the set of candidate values.

The classical counterpart of Grover's algorithm has a time complexity of $O(m)$, as in the worst case all the m values need to be evaluated in order to find the solution. Grover's algorithm leads to a polynomial acceleration of this approach, as it solves the same problem with time complexity $O(\sqrt{m})$.

Mathematically in quantum terms the problem that Grover's algorithm solves includes an operator $\hat{O}$ - which is called Oracle - that acts on n qubits and transforms the 2^n basis states according to the rule:

$$\begin{cases} \hat{O}\left|x\right\rangle = -\left|x\right\rangle, \left|x\right\rangle = \left|x^*\right\rangle \\ \hat{O}\left|x\right\rangle = \left|x\right\rangle, \left|x\right\rangle \neq \left|x^*\right\rangle \end{cases}$$

The target of the algorithm is to find the unique basis vector $\left|x^*\right\rangle$.

Grover's algorithm includes a specific sequence of steps:

1. Initially a superposition of all 2^n basis states is prepared using the Hadamard gate:

$$\left|00\cdots0\right\rangle \xrightarrow{H} \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} \left|x\right\rangle$$

2. After that, the Oracle operator is applied to the resulting superposition:

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} \left|x\right\rangle \xrightarrow{\hat{O}} \frac{1}{\sqrt{2^n}} \left(\left(\sum_{x \neq x^*} \left|x\right\rangle \right) - \left|x^*\right\rangle \right)$$

3. The next step is called inversion about the average which is a transformation of the form:

$$\sum_{i=0}^{2^n-1} a_i \left|i\right\rangle \rightarrow \sum_{i=0}^{2^n-1} (2M - a_i) \left|i\right\rangle$$

where M is the mean value of all the a_i. This transformation is achieved with a unitary operator called Diffusion operator which has the form:

$$D = \begin{bmatrix} \frac{2}{2^n} - 1 & \frac{2}{2^n} & \cdots & \frac{2}{2^n} \\ \frac{2}{2^n} & \frac{2}{2^n} - 1 & \cdots & \frac{2}{2^n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{2}{2^n} & \frac{2}{2^n} & \cdots & \frac{2}{2^n} - 1 \end{bmatrix}$$

4. The algorithm continues with repeating the application of the Oracle and Diffusion operators $\frac{\pi}{4}\sqrt{2^n}$ times in total.

5. The process concludes with the measurement of the n qubits which will output the value of x^*.

Intuitively the purpose of the use of the Oracle and the Diffusion operators is not obvious. In fact their combination iteratively increases the amplitude of $|x^*\rangle$ and reduces all the other amplitudes, as the following figures illustrate:

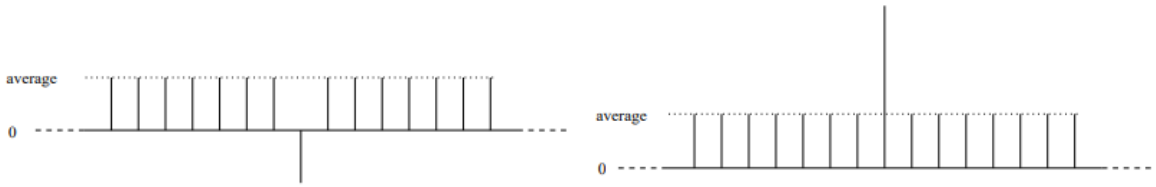

Figure 2.8: The effect of the application of the Oracle operator (left) and the Diffusion operator (right)

When the algorithm concludes, all the amplitudes tend to 0 except for the amplitude of $|x^*\rangle$ which tends to 1, so the measurement produces the expected basis vector with high probability.

Alternatively Grover's algorithm can also be explained with its geometrical representation which is based on the 2D plot that is shown below:

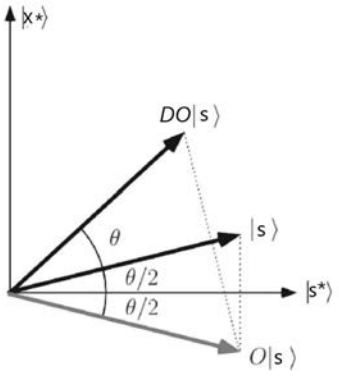

Figure 2.9: Geometrical representation of Grover's algorithm

The axis of $|s^*\rangle$ corresponds to all basis vectors except for $|x^*\rangle$, $|s\rangle$ is the superposition of all basis vectors - which is the result of step 1 of the algorithm - and $\theta/2$ is the angle between $|s\rangle$ and $|s^*\rangle$. The result of step 2 is $\hat{O}\,|s\rangle$ which is actually a reflection of $|s\rangle$ about the axis of $|s^*\rangle$. After that the result of step 3 is $D\hat{O}\,|s\rangle$ which basically reflects the previous result about $|s\rangle$. In total the gain of the use of the Oracle and Diffusion operators is the rotation of the initial vector $|s\rangle$ by an angle of $\theta/2$ towards the axis of $|x^*\rangle$. After k iterations of steps 2 and 3, the respective total rotation is $k\theta/2$. It can be proved that for $k = \frac{\pi}{4}\sqrt{2^n}$ the vector finally identifies with the axis of $|x^*\rangle$, so a measurement takes place and the algorithm finishes.

Chapter 3

Literature Review

3.1 Introduction

In this chapter I analyze and provide some insight into the research that has been done on quantum linear system solvers over the last few years. First of all, in 3.2 I describe the quantum mechanical version of the linear system problem. In 3.3 I present a preliminary method which utilizes Grover's algorithm as its main tool. Afterwards I discuss the main eigenvalue quantum methods for linear systems. In particular, in 3.4 I focus on the most well-known quantum solver, the HHL algorithm, and in 3.5 I analyze the WZP algorithm which is similar to the HHL algorithm and can be efficiently applied both to dense and sparse matrices. In 3.6 I discuss the iterative solution algorithms and in particular I describe the row and column iteration methods. In 3.7 I focus on the variational quantum linear solver which is a hybrid method that requires both a classical and a quantum computer in order to compensate for the flawed nature of the current quantum computers. Finally, in 3.8 I present another hybrid algorithm which is based on random walks. To complete this chapter, in 3.9 I summarize and compare the algorithms which were analyzed in the previous sections.

3.2 Quantum linear system problem

In the introduction of this thesis I defined the Linear Systems Problem (LSP) which has the form $A\vec{x} = \vec{b}$, where $A \in \mathbf{R^{n \times n}}$ and $\vec{b} \in \mathbf{R^n}$ are known and the objective is to find the unknown vector $\vec{x} \in \mathbf{R^n}$ such that $\vec{x} = A^{-1}\vec{b}$. The quantum mechanical version of this problem, which is commonly referred to as Quantum Linear Systems Problem (QLSP), has

the form:

$$A\left|x\right\rangle = \left|b\right\rangle$$

where A is a quantum operator, while $|x\rangle$ and $|b\rangle$ are quantum states represented by quantum vectors. The operator A and the vector $|b\rangle$ are given and the purpose of the QLSP is to construct a quantum state $|x\rangle$ such that:

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

Contrary to the classical LSP, $|x\rangle$ is not the exact solution of the linear system, because such quantum vectors are normalized by default. In fact, $|x\rangle$ is proportional to the actual solution vector:

$$\left|x\right\rangle = \frac{\sum_{i=1}^{n} x_i \left|i\right\rangle}{\sqrt{\sum_{i=1}^{n} \left|x_i\right|^2}}.$$

For a quantum linear solver to be successful, given a precision parameter $\epsilon > 0$, the produced quantum state ρ_x needs to satisfy the following inequality for the trace distance $D_{\rho_x,x}$:

$$D_{\rho_x,x} = \frac{1}{2} Tr|\rho_x - \left|x\right\rangle \left\langle x\right| | \leq \epsilon$$

3.3 Grover's algorithm solution method

3.3.1 Description and main results

One of the most simple and accessible ways to solve a linear system with a quantum algorithm was proposed by Maji in [1]. In this paper she introduces an approach that deploys Grover's algorithm to solve $n \times n$ sets of equations of specific form using total resources of $\lceil \log_2 n \rceil$ qubits. More specifically, this algorithm is applicable to systems $A\vec{x} = \vec{b}$ where the matrix A is unitary:

$$AA^{\dagger} = I$$

The vectors $\vec{x}, \vec{b}$ either have a length of 1 or they are normalized, so that they can be represented as quantum state vectors. To find the solution vector $\vec{x}$ that satisfies the set of equations, Maji tries to find a unitary operator U for which $UA = I$. This is actually the inverse of A and satisfies the relation:

$$UA = I \Rightarrow UA\vec{x} = \vec{x} \Rightarrow U\vec{b} = \vec{x}$$

To achieve this, she represents the columns of matrix A as quantum states $|\psi_i\rangle$:

$$A = \Big[|\psi_1\rangle\, |\psi_2\rangle \cdots |\psi_n\rangle\Big]$$

which means that:

$$UA = \Big[U\,|\psi_1\rangle\, U\,|\psi_2\rangle \cdots U\,|\psi_n\rangle\Big] = I.$$

As a consequence, the matrix-vector multiplication $U\,|\psi_i\rangle$ needs to be equal to the i-th column of the identity matrix.

Maji suggests utilizing Grover's algorithm to calculate U. She uses the unknown solution vector as the target vector $|x^*\rangle$ and afterwards she aggregates the Oracle and Diffusion operators which are used during the iterations of the algorithm. To illustrate the idea of this algorithm, she presents the solution of the following 4×4 set of equations:

$$\begin{aligned} x_1 + x_2 + x_3 + x_4 &= 2 \\ x_1 - x_2 - x_3 + x_4 &= 0 \\ x_1 - x_2 + x_3 - x_4 &= 0 \\ x_1 + x_2 - x_3 - x_4 &= 0 \end{aligned}$$

In order to get a unitary matrix A and a normalized vector $\vec{b}$, all equations have to be divided by 2. The resulting A and $\vec{b}$ are shown below:

$$A = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & -1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 & -1/2 \\ 1/2 & 1/2 & -1/2 & -1/2 \end{bmatrix}, \vec{b} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The application of Grover's algorithm shows that $U = A$ and the solution vector is calculated to be:

$$\vec{x} = U\vec{b} = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$$

The circuit which is used for the multiplication of U with $\vec{b}$ appears in the following figure:

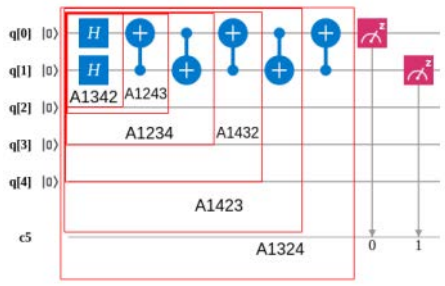

Figure 3.1: Circuit implementation of $U\vec{y}$ ([1])

The input to the circuit corresponds to the vector $\vec{b} = \begin{bmatrix}1000\end{bmatrix}^T = |00\rangle$, whereas the quantum gates form the operator U. The notation A_{ijkl} expresses the permutation of the equations - for example, A_{1243} means that the third and forth equations of the linear system are permuted at this point. The circuit contains explicitly Hadamard and CNOT gates. The latter are shown with the symbol that contains a "+" in the figure and their role is just to permute the equations. The two Hadamard gates at the beginning of the circuit are in fact sufficient to form A_{1342}:

$$A_{1342} = H \otimes H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \otimes \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} = \begin{bmatrix}1/2 & 1/2 & 1/2 & 1/2\\ 1/2 & -1/2 & 1/2 & -1/2\\ 1/2 & 1/2 & -1/2 & -1/2\\ 1/2 & -1/2 & -1/2 & 1/2\end{bmatrix}$$

For the experiments which are presented in the paper, the IBM quantum simulator was used. The accuracy of the results was evaluated with fidelity [10], which measures the similarity between the calculated solution vector and the theoretical solution vector and is equal to the absolute value of the inner product of the two vectors, if these vectors represent pure quantum states. The paper mentions a fidelity of 0.9878 for the above set of linear equations. In addition to that, it cites the following plots that concern the density matrix of the solution state vector:

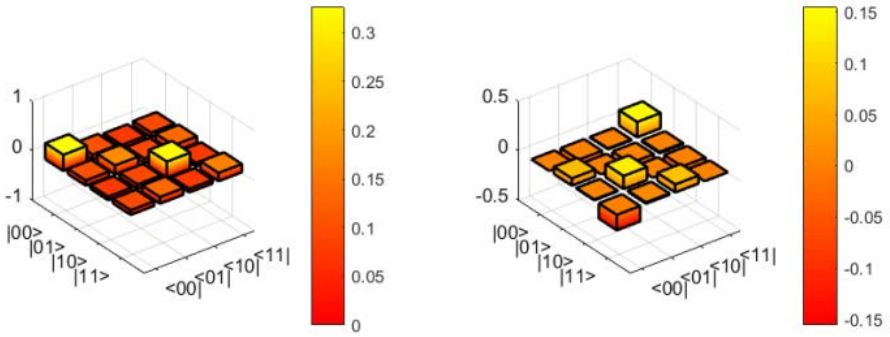

Figure 3.2: The density matrix of the solution state vector (left), additive error of the elements of the density matrix(right) ([1])

3.3.2 Review of the algorithm

Although the approach of Maji includes interesting ideas, it seems to have some important limitations which cannot be overlooked. First of all, there is probably no need to compute the unitary matrix U, as its form is known in advance:

$$UA = I \Rightarrow UAA^{\dagger} = A^{\dagger} \Rightarrow U = A^{\dagger}$$

The above example also contains only real numbers, so U is just the transpose of A:

$$U = A^{-1} = A^{T}$$

In addition, the use of Grover's algorithm does not seem to comply with the description which was given in 2.16 where an equal superposition of all basis states is rotated to find an unknown basis vector. The algorithm can be extended to cases where the target vector is not just a basis vector, but a superposition of multiple basis states, however, even in this case it does not allow us to compute complex quantum state vectors of arbitrary form.

The resources that the approach of this paper needs should also be taken into consideration. More specifically, it should be noted that apart from the qubits that are required, a reconfigurable quantum processing unit is needed which can adjust the quantum circuit according to the form of the linear equations. Finding the structure of the quantum circuit is also a substantial challenge. The example of the paper includes only two types of quantum gates, however, other linear systems may require more complex combinations of gates. A similar challenge occurs during the preparation of vector $\vec{b} \equiv |b\rangle$ in quantum form. In the example

of the paper $|b\rangle = |00\rangle$, so this step is omitted. Nevertheless, other linear systems may need the selection of some gates which will transform $|00\rangle$ to the quantum state that corresponds to $|b\rangle$.

To conclude, the work of Maji is very interesting and serves as an introduction to the field of quantum solvers for linear systems. Its dependence on Grover's algorithm implies an approximate time complexity of $\mathrm{O}(\sqrt{n})$ which translates to a quadratic speed-up in comparison to the fastest classical algorithms. However, it seems to have some restrictions and it is limited to few sets of equations of specific form.

3.4 The HHL solution method

3.4.1 The conception of the algorithm

The most prominent quantum solution method for linear systems of equations was proposed by Harrow, Hassidim and Lloyd [11]. Their approach, which is commonly referred to as HHL in bibliography, involves $n \times n$ linear systems of the form $A\vec{x} = \vec{b}$ where the matrix A is Hermitian and the vectors $\vec{x}, \vec{b}$ have a length of 1 or they are normalized, so that they can be represented as quantum states. The algorithm can be extended for non-Hermitian matrices, by transforming a linear set of equations as follows:

$$A' = \begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix}, \vec{x'} = \begin{bmatrix} 0 \\ \vec{x} \end{bmatrix}, \vec{b'} = \begin{bmatrix} \vec{b} \\ 0 \end{bmatrix}$$

After this transformation, the new matrix A' is Hermitian, as $A'^\dagger = A'$, and $A'\vec{x'} = \vec{b'}$ is also satisfied.

Hermitian matrices have some special properties and this is the reason why they are used for this algorithm. More specifically, a Hermitian matrix A can be decomposed as follows:

$$A = \sum_{i=1}^{n} \lambda_i |u_i\rangle \langle u_i|$$

where λ_i are its eigenvalues and $|u_i\rangle$ are its corresponding eigenvectors. To find the inverse of A we just need to substitute the eigenvalues λ_i with λ_i^{-1}:

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

Therefore, if we express the vector $|b\rangle \propto \vec{b}$ so that it is a linear combination of the eigenvectors of A:

$$|b\rangle = \sum_{i=1}^{n} b_i |u_i\rangle$$

we can find the solution of the linear system using the formula:

$$|x\rangle = A^{-1} |b\rangle = \sum_{i=1}^{n} \lambda_i^{-1} b_i |u_i\rangle .$$

The HHL algorithm relies on this idea and it is explained in more detail in the next subsection.

3.4.2 Description of the algorithm

The purpose of the HHL algorithm is to produce an output solution quantum state of the form $|x\rangle = \sum_{i=0}^{n-1} x_i |i\rangle$ so that the elements x_i are proportional to the ones of the solution vector. Harrow et al suggest that not all elements of this vector are extracted, because this would require at least n iterations of their implementation (measurements of the output qubits lead to the collapse of the quantum state, so the whole procedure needs to be repeated). Therefore, the writers propose the estimation of an operator $\vec{x}^{\dagger} M \vec{x}$ instead which requires only one iteration of the algorithm and can reveal a variety of features and properties about $\vec{x}$.

The qubit requirements of the algorithm of Harrow et al are the following:

- m qubits of working space initialized in $|0\rangle^{\otimes m}$
- $\lceil \log_2 n \rceil$ qubits which host the vector $|b\rangle$
- an ancilla (supplementary) qubit initialized in $|0\rangle$

The paper does not include an analytical explanation about how to prepare the vector $|b\rangle$ as a quantum state, but proposes the work of Grover et al [12] as a tool which can be used for this reason. Using the eigenvectors $|u_i\rangle$ as the basis of the state space, $|b\rangle$ can be represented as follows:

$$|b\rangle = \sum_{i=1}^{n} \beta_i |u_i\rangle \tag{3.1}$$

where $\beta_i = \langle u_i | b \rangle$. The HHL solution method can be divided into three steps:

1. Initially the algorithm contains a phase estimation step where the working space qubits play the role of the control qubits and the unitary operator which is used is the conditional Hamiltonian evolution [13]:

$$U = \sum_{k=0}^{T-1} |k\rangle \langle k| \otimes e^{iAkt_0/T}.$$

An analytical explanation of this operator is beyond the scope of this thesis. In a few words its purpose is to construct the exponentiated e^{iAt} for a superposition of different values of t which is then applied to $|b\rangle$, as an operator of this form is suitable for mapping the eigenvalues of A on the working space. The parameter t_0 takes a value $\mathrm{O}(\kappa/\epsilon)$ - where κ is the condition number of the matrix A and ϵ is the additive error of the solution vector - and in some experimental applications ([2], [14]) it is set equal to 2π. On the other hand, T is equal to 2^m (m is the number of qubits in the working space). The resulting state after the first step of the algorithm is:

$$\sum_{i=1}^{n} \beta_i |u_i\rangle |\lambda_i\rangle . \tag{3.2}$$

where λ_i are the eigenvalues of A represented with a precision of m bits.

2. The second step of the algorithm includes a controlled rotation process $R(\lambda_i^{-1})$ which is applied on the ancilla qubit. As A is Hermitian, λ_i^{-1} are the eigenvalues of the inverse of A. These eigenvalues need to be incorporated in the state (3.2). To achieve this, the values of λ_i first need to be extracted mechanically in order to calculate the angle θ_i of the rotation, for which $\sin(\theta_i) = \frac{C}{\lambda_i}$ with $C \in \mathbf{R}$. The state that occurs from this process is:

$$\sum_{i=1}^{n} \beta_i |u_i\rangle |\lambda_i\rangle \left(\sqrt{1 - \frac{C^2}{\lambda_i^2}} |0\rangle + \frac{C}{\lambda_i} |1\rangle \right), C \in \mathbf{R}. \tag{3.3}$$

3. The final step is the reverse of the first step, so the qubits of the working space are reset to $|0\rangle$ and the resulting state is:

$$\sum_{i=1}^{n} \beta_i |u_i\rangle \left(\sqrt{1 - \frac{C^2}{\lambda_i^2}} |0\rangle + \frac{C}{\lambda_i} |1\rangle \right). \tag{3.4}$$

The algorithm terminates with a postselection on the ancilla qubit being in the state $|1\rangle$. This means that we condition on the outcome of the measurement of this qubit being $|1\rangle$, so the quantum state that occurs from this process is:

$$\sum_{i=1}^{n} C \frac{\beta_i}{\lambda_i} |u_i\rangle$$

which is proportional to the theoretical solution. A schematic diagram which summarized the steps of the HHL algorithm is shown below:

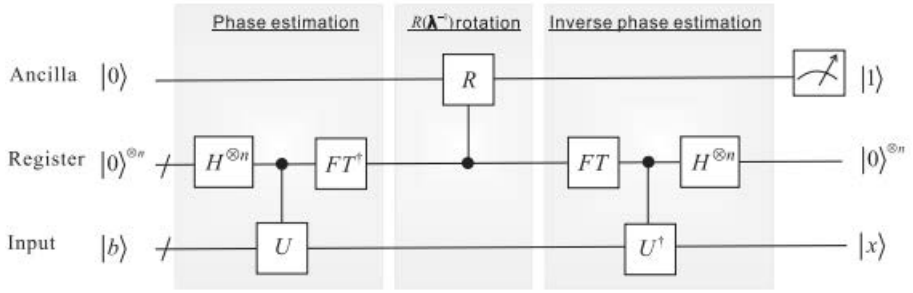

Figure 3.3: Schematic representation of the HHL algorithm ([2])

3.4.3 Review of the algorithm

The performance of classical linear solvers varies, but most of them solve a linear system with an approximate time complexity of $\mathrm{O}(n^3)$, which means that it scales polynomially with the size of the system. On the other hand, the corresponding complexity of the proposed quantum HHL algorithm is $\mathrm{O}(s^2\kappa^2 \log n/\epsilon)$ if the matrix A is s-sparse, namely if it has at most s non-zero elements per row. This means that its runtime scales logarithmically with the size of the system and an exponential speed-up is achieved in comparison to the classical algorithm, provided that A is not very dense. The density of the matrix is not the only factor that may reduce the performance of the algorithm though. We need to take into consideration that κ sometimes scales polynomially or even exponentially with n. In this case it is doubtful that HHL can be faster than its classical counterparts. For this reason, it is important to control the scaling of κ which could possibly be achieved in two ways according to the paper:

- By processing only the well conditioned part of A:

$$\sum_{j,\lambda_j<1/\kappa} \lambda_j^{-1}\beta_j \left|u_j\right\rangle \left|\text{well}\right\rangle + \sum_{j,\lambda_j\geq 1/\kappa} \beta_j \left|u_j\right\rangle \left|\text{ill}\right\rangle$$

- By applying preconditioners in order to improve the condition of the matrix before proceeding with the algorithm.

Moreover, it is important to note the dependence of the complexity of HHL on the precision parameter ϵ. This is mainly related to the use of the phase estimation routine as part of the algorithm and could reduce its performance significantly if high accuracy of the solution is required. Finally we should take into account that there are some possible sources of failure. First of all, phase estimation can produce wrong results under certain circumstances as it is explained in 2.15. Apart from that, postselection is not a process which is guaranteed to

succeed. According to the paper, if the constant C is $\mathrm{O}(1/\kappa)$ then $\mathrm{O}(\kappa)$ repetitions of the routine including the postselection step seem to be enough to produce the desired result.

To conclude, the HHL algorithm is a breakthrough in the field of quantum linear solvers although it has some limitations. This work is definitely a significant contribution to the research of quantum computing and this is illustrated by the fact that the solvers which were developed later where mainly inspired by it.

3.5 WZP method for dense matrices

Another notable quantum solution method for linear sets of equations is that proposed by Wossnig, Zhao and Prakash [15] which is commonly referred to as WZP in bibliography. This algorithm has some similarities with the HHL algorithm and its main advantage is that it can work well both for dense and sparse matrices.

The algorithm focuses on $n \times n$ linear systems of the form $A\vec{x} = \vec{b}$ where A is Hermitian. However, this is not actually a restriction, as a general form matrix can be transformed to a Hermitian matrix using the formula that was described in the case of the HHL algorithm.

3.5.1 Quantum singular value estimation

The WZP algorithm relies on a procedure called quantum singular value estimation (QSVE) which was proposed by Kerenidis and Prakash [16] in their implementation of a quantum recommendation system. QSVE is a factorization method which helps calculate the eigenvalues of the matrix A. More specifically, it takes as an input a state of the form $\sum_i a_i |v_i\rangle$ where v_i are the singular vectors of A and transforms it to a state of the form $\sum_i a_i |v_i\rangle |\bar{\sigma}_i\rangle$ where $\bar{\sigma}_i$ are approximations of the singular values of A. This process is based on two matrices P and Q which are defined as follows:

- The i-th column of the matrix $P \in \mathbf{R}^{n^2 \times n}$ is equal to $e_i \otimes \frac{A_i}{\|A_i\|}$ with $1 \leq i \leq n$, where A_i is the i-th row of the matrix A. This means that P has the following form:

$$P = \begin{bmatrix} \frac{A_1}{\|A_1\|} & 0 & \cdots & 0 \\ 0 & \frac{A_2}{\|A_2\|} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \frac{A_m}{\|A_m\|} \end{bmatrix}$$

- The j-th column of the matrix $Q \in \mathbf{R}^{n^2 \times n}$ is equal to $\frac{A_F}{\|A\|_F} \otimes e_j$ with $1 \leq j \leq n$, where $A_F \in \mathbf{R^n}$ is a vector which contains the Frobenius norms of the rows of the matrix A and $\|A\|_F$ is the Frobenius norm of the matrix A. This is equivalent to:

$$Q = \begin{bmatrix} \frac{A_F}{\|A\|_F} & 0 & \cdots & 0 \\ 0 & \frac{A_F}{\|A\|_F} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \frac{A_F}{\|A\|_F} \end{bmatrix}$$

From the the above matrices, it is clear that P and Q are orthogonal, as $P^\dagger P = I_n$ and $Q^\dagger Q = I_n$. Moreover, the following equalities hold:

- $$|Pe_i\rangle = |i, A_i\rangle = \frac{1}{\|A_i\|} \sum_{j=1}^{n} A_{ij} |i, j\rangle \, , 1 \leq i \leq n \tag{3.5}$$

- $$|Qe_j\rangle = |A_F, j\rangle = \frac{1}{\|A\|_F} \sum_{i=1}^{n} \|A_i\| \, |i, j\rangle \, , 1 \leq j \leq n \tag{3.6}$$

and for the implementation of the these quantum states a data structure is used which performs the mappings:

$$U_M : |i, 0\rangle \rightarrow |i, A_i\rangle = \frac{1}{\|A_i\|} \sum_{j=1}^{n} A_{ij} \, |i, j\rangle$$

$$U_N : |0, j\rangle \rightarrow |A_F, j\rangle = \frac{1}{\|A\|_F} \sum_{i=1}^{n} \|A_i\| \, |i, j\rangle$$

An analytical presentation of this data structure is beyond the scope of this thesis, but an interested reader can find more details in [16]. Combining (3.5) and (3.6) to calculate the (i, j)-th element of the matrix $P^\dagger Q$ we find:

$$(P^\dagger Q)_{i,j} = \langle i, A_i | A_F, j \rangle = \frac{A_{ij}}{\|A_i\|} \frac{A_i}{\|A\|_F} = \frac{A_{ij}}{\|A\|_F}$$

Therefore, the matrices P and Q can be used to factorize a normalized version of the matrix A:

$$P^\dagger Q = \frac{A}{\|A\|_F} \tag{3.7}$$

Based on the matrices P and Q, two other matrices U and V are defined:

$$U = 2PP^\dagger - I_{n^2}, V = 2QQ^\dagger - I_{n^2}.$$

These are the reflections in the column space of P and Q respectively. This is due to the fact that for any vector $\vec{x}$ that does not belong to the column space of P:

$$U\vec{x} = (2PP^\dagger - I)\vec{x} = -\vec{x}$$

and in addition to that, U does not change a vector which belongs to the column space of P, because:

$$UP = (2PP^\dagger - I)P = 2P - P = P.$$

A similar proof shows that V is a reflection in the column space of Q. The matrix which occurs from the multiplication of U with V is denoted by W:

$$W = UV$$

and it a significant part of the QSVE algorithm. In particular, it can be proved that given a singular vector v_i of A with singular value σ_i, Qv_i is an eigenvector of W with eigenvalue $e^{i\theta_i}$ where $\cos\theta_i/2 = \sigma_i/\|A\|_F$.

This result stems from the research on bipartite quantum walks [17] and its proof is worth mentioning. A preliminary element of this proof is the fact that an orthogonal projection which maps a vector Qv of the column space of Q to the column space of P is based on the orthogonal projector $PP^\dagger$, as:

$$(PP^\dagger)Qv = \frac{1}{\|A\|_F}PAv.$$

Similarly mapping a vector Pw to the column space of Q is based on the orthogonal projector $QQ^\dagger$:

$$(QQ^\dagger)Pw = \frac{1}{\|A\|_F}QA^\dagger w.$$

Given a singular value σ_i of A with right singular unit vector v_i and left singular unit vector u_i, we have:

$$Av_i = \sigma_i u_i \Rightarrow PP^\dagger Qv_i = \frac{\sigma_i}{\|A\|_F}Pu_i \tag{3.8}$$

$$A^\dagger u_i = \sigma_i v_i \Rightarrow QQ^\dagger Pu_i = \frac{\sigma_i}{\|A\|_F}Qv_i \tag{3.9}$$

It can be observed that (3.8) transforms Qv_i to Pu_i, thus it transforms a vector from the column space of Q to the column space of P. On the other hand, (3.9) does the opposite process. Furthermore, it is easy to see that:

$$|Qv_i| = \sqrt{v_i^\dagger Q^\dagger Qv_i} = \sqrt{v_i^\dagger v_i} = |v_i|.$$

and similarly we can find that $|Pu_i| = |u_i|$. Taking these observations into consideration, it is evident that $|\ \frac{\sigma_i}{\|A\|_F}\ | \leq 1$, so we can define an angle $\theta_i/2$ such that:

$$\cos(\theta_i/2) = \frac{\sigma_i}{\|A\|_F}.$$

Geometrically $\theta_i/2$ is equal to the angle between Pu_i and Qv_i, as:

$$u_i^\dagger P^\dagger Q v_i = \frac{1}{\|A\|_F} u_i^\dagger A v_i = \frac{1}{\|A\|_F} u_i^\dagger \sigma_i u_i = \frac{\sigma_i}{\|A\|_F} = \cos(\theta_i/2).$$

We can alternatively express (3.8) and (3.9) as follows:

$$PP^\dagger Q v_i = \cos(\theta_i/2) P u_i$$

$$QQ^\dagger P u_i = \cos(\theta_i/2) Q v_i.$$

The subspace with basis $\{Pu_i, Qv_i\}$ can be either 1-dimensional or 2-dimensional, but here we focus on the non-trivial 2-dimensional case. This subspace is invariant under the action of the matrix W. In fact, W is a reflection in the column space of Q followed by a reflection in the column space of P which in the case of the aforementioned 2-dimensional space translates to a reflection on axis Qv_i followed by a reflection on axis Pu_i. The total effect of these two reflections is a rotation by an angle that is double the angle between the axes, which is equal to $\theta_i/2$ as mentioned above. Therefore e^{θ_i} is an eigenvalue of W and the following equality holds:

$$W(Qv_i) = e^{\theta_i}(Qv_i)$$

Having explained the main idea behind QSVE, the rest of this subsection analyzes the different steps of this routine for a matrix $A \in \mathbf{R}^{n \times n}$:

1. First of all we prepare the quantum state of an arbitrary vector which can be expressed using a basis of the vector space that consists of the singular vectors v_i of A:

$$|x\rangle = \sum_i a_i |v_i\rangle .$$

2. We transform the previous quantum state using a Q operator:

$$|Qx\rangle = \sum_i a_i |Qv_i\rangle$$

and append $\lceil log_2 n \rceil$ qubits initialized in $|0\rangle$ to the resulting state. These qubits will be useful for the next step.

3. We perform Phase Estimation for W. This requires the preparation of operators of the form W^{2^k}. The eigenvectors which are needed are included in the state $|Qx\rangle$. The qubits which were previously initialized to $|0\rangle$ will output an approximation $\bar{\theta}_i$, where $\bar{\theta}_i$ is an estimation of the eigenvalues of W. The resulting state of this step is the following:

$$\sum_i a_i \left|Qv_i, \bar{\theta}_i\right\rangle$$

4. Knowing $\bar{\theta}_i$, it is easy to transform the state in order to approximate the singular values σ_i of A. This is achieved using the formula:

$$\bar{\sigma_i} = \cos(\bar{\theta}_i/2)\|A\|_F.$$

5. To complete the routine, we need to reverse the effect of the operator Q which was used during step 2 and reach the final state:

$$\sum_i a_i \left|v_i\right\rangle \left|\bar{\sigma_i}\right\rangle$$

3.5.2 Description of the algorithm

The above routine is a significant part of the proposed WZP method. As the algorithm works with Hermitian matrices, the knowledge of the approximations $\bar{\sigma_i}$ of the singular values of A translates to the knowledge of the absolute value of the eigenvalues of A, thus $\bar{\sigma_i} = |\bar{\lambda_i}|$. This facilitates the solution of a linear system, however, the signs of the eigenvalues of A remain to be found. This is achieved during the following complete routine of the WZP method which assumes that A has been scaled so that $\lambda_i \in [-1, -1/\kappa] \cup [1/\kappa, 1]$ (κ is the condition number of A):

1. Initially we prepare the state of an arbitrary vector and express it as a linear combination of the singular vectors v_i of A:

$$|b\rangle = \sum_i \beta_i \left|v_i\right\rangle$$

2. We use the QSVE routine for the matrices A and $A + \mu I$ with $\mu = 1/\kappa$. The matrix $A + \mu I$ has the same eigenvectors as A and its eigenvalues are equal to $\mu + \lambda_i$. The resulting state of this step is the following:

$$\sum_i \beta_i \left|v_i\right\rangle_A \left||\bar{\lambda_i}|\right\rangle_B \left||\bar{\lambda_i} + \mu|\right\rangle_C$$

The notation A, B, C divides the generated quantum state into individual registers. The values of the registers B and C reveal the sign of the eigenvalues λ_i. More specifically, if $\lambda_i > 0$ then $|\lambda_i + \mu| > |\lambda_i|$, so the value of register C is larger than the value of register B. On the other hand, if $\lambda_i < 0$ we have:

$$\lambda_i < -1/k \Rightarrow \lambda_i < -\mu \Rightarrow \lambda_i + \mu < 0$$

because of the aforementioned scaling of A. Thus, this means that:

$$\lambda_i < \lambda_i + \mu \Rightarrow -|\lambda_i| < -|\lambda_i + \mu| \Rightarrow |\lambda_i| > |\lambda_i + \mu|$$

and, as a consequence, in this case the value of register B is larger than the value of register C.

3. At this point we need an additional ancilla register D which is set to $|1\rangle$ if register B contains a larger value than register C, namely when $\lambda_i < 0$. We apply a conditional phase gate so that the state that occurs is the following:

$$\sum_i (-1)^{f_i} \beta_i \left|v_i\right\rangle_A \left||\bar{\lambda}_i|\right\rangle_B \left||\bar{\lambda}_i + \mu|\right\rangle_C \left|f_i\right\rangle_D$$

4. Finally we add another ancilla register and apply a controlled rotation by $\gamma = \mathrm{O}(1/\kappa)$ conditioned on the register B. After that, the registers B, C, D are not necessary for the result, so we uncompute them, namely we reset them to $|0\rangle$, and reach the state:

$$\sum_i (-1)^{f_i} \beta_i \left|v_i\right\rangle \left(\frac{\gamma}{|\bar{\lambda}_i|} |0\rangle + \sqrt{1 - \left(\frac{\gamma}{\bar{\lambda}_i}\right)^2} |1\rangle \right).$$

To obtain a state $|x\rangle$ proportional to the solution of the linear system, we postselect on the register which was added in this step being in state $|0\rangle$. The choice of the value of γ is such that the measurement produces $|0\rangle$ with high probability, so in any case not too many iteration should be necessary.

3.5.3 Review of the algorithm

Similar to the previous quantum linear solvers, the WZP method proves quantum supremacy and achieves an exponential speed-up in the size of a linear system, while it works quite well for dense matrices. More specifically, the complexity of the algorithm is $\mathrm{O}(\kappa^2 \log n \|A\|_F / \epsilon)$ where κ is the condition number of the matrix A and and n is the size of the linear system.

According to the paper, in many cases the Frobenius norm of A is bounded so that $\|A\|_F = \mathrm{O}(\sqrt{n})$ and as a result, the complexity of the algorithm is approximately $\mathrm{O}(\kappa^2 \log n\sqrt{n}/\epsilon)$. This is comparable to the other linear solvers which are presented in this thesis.

The WZP method has some resemblance to the HHL algorithm, mainly because it is based on the eigenvalues of A in order to calculate the solution vector. In addition, both algorithms use phase estimation as part of their routines. However, this can significantly increase their cost, as phase estimation is an expensive procedure when high accuracy is required. On the other hand, WZP has an important advantage over HHL. In particular, HHL requires the preparation of suitable Hamiltonians which are difficult to implement in order to appropriately represent A, whereas WZP relies on a treelike data structure which seems easier to construct. Therefore, it would be justified to prefer WZP over HHL under certain conditions.

3.6 Row and column iteration methods

3.6.1 Theoretical presentation of the algorithms

The use of iteration methods as quantum solvers for linear sets of equations has been studied extensively during recent years, as the potential of quantum parallelism can lead to a speed-up in comparison to the corresponding classical algorithms. In general classical iterative methods are widely used nowadays for large scale linear systems, because they are faster than the direct methods, especially in cases where not a very precise solution is required. The complexity of these classical algorithms is approximately $\mathrm{O}(Tn)$ where T is the total number of iterations and n is the size of a system $A\vec{x} = \vec{b}$. On the other hand, quantum iterative algorithms achieve an exponential acceleration in n, however, their main limitation is that T significantly reduces their performance - some of them even have an exponential dependence on it. This is mainly due to the requirement on multiple copies of the approximation x_k of the solution, which is calculated during the k-th iteration, in order to proceed to the $(k+1)$-th iteration. The no-cloning theorem states that creating copies of quantum states is impossible, so the algorithms need to restart repeatedly from x_0 for an exponential number of times until the desired number of copies of x_k is reached.

Current research indicates that it is probably not feasible to implement a quantum algorithm that can outperform classical iterative solution methods both in n and T. An exponential

speed-up in n is definitely possible, however, a polynomial complexity in T is the best we can expect at the moment.

An interesting work towards that direction was presented by Shao et al [3]. Their proposed algorithm is based on the row and column iteration methods which are used in a lot of large scale data science applications nowadays. The main advantage of these methods is that for every iteration not the whole matrix A is needed, but only one of its rows or one of its columns. More specifically the two methods are defined as follows:

- The row (or Kaczmarz) iteration method selects in each iteration a row with index $1 \leq i_k \leq n$ with probability proportional to the norm of the rows of the matrix. If we denote the i-th row of A as a_i and the i-th element of the vector $\vec{b}$ as b_i, the $(k+1)$-th approximation of the solution of the system is calculated according to the following formula:

$$x_{k+1} = x_k - \left(\frac{a_{i_k}^T x_k}{\|a_{i_k}\|} - \frac{b_{i_k}}{\|a_{i_k}\|} \right) \frac{a_{i_k}}{\|a_{i_k}\|} \tag{3.10}$$

  Geometrically a step of the row iteration method is equivalent to the orthogonal projection of the vector x_k on the plane $a_{i_k}^T x = b_{i_k}$. For example, for a small linear system with $n = 2$, this process can be represented in a 2D plane, as the following figure illustrates:

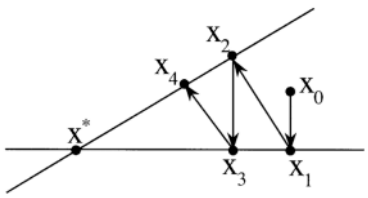

Figure 3.4: Row iteration method for $n = 2$ ([3])

- The column (or coordinate descent) iteration method selects in each iteration a column with index $1 \leq j_k \leq n$ with probability proportional to the norm of the columns of the matrix. If we denote the j-th row of A as c_j and the j-th column of the identity matrix as e_j, the $(k+1)$-th approximation of the solution of the system is calculated according to the following formula:

$$x_{k+1} = x_k + \frac{c_{j_k}^T (b - Ax_k)}{\|c_{j_k}\|^2} e_{j_k} \tag{3.11}$$

If we denote the residual of the k-th iteration as $r_k = b - Ax_k$, the expression of (3.11) can be written as follows:

$$x_{k+1} = x_k + \frac{c_{j_k}^T r_k}{\|c_{j_k}\|^2} e_{j_k}, \tag{3.12}$$

where the residual also follows an iterative scheme:

$$r_{k+1} = b - Ax_{k+1} = b - Ax_k - A\frac{c_{j_k}^T r_k}{\|c_{j_k}\|^2} e_{j_k} = r_k - \frac{c_{j_k} c_{j_k}^T}{\|c_{j_k}\|^2} r_k \tag{3.13}$$

3.6.2 Quantum row iteration method

The quantum algorithm of Shao et al assumes the existence of a QRAM [18] which is responsible for preparing a row a_t in the form of a quantum state. This means that the QRAM plays the role of a unitary operator V_t which performs the transformation:

$$V_t |0\rangle = |a_t\rangle .$$

As $|a_t\rangle$ is a unit vector, all the rows of A are supposed to be scaled to $\|a_t\| = 1$ so that they can be represented as quantum states. As far as the solution unit vector approximations $|x_k\rangle$ are concerned, for the purpose of the algorithm they are included in quantum states $|X_k\rangle$, such that:

$$|X_k\rangle = \sqrt{p}\,|0\rangle\,|x_k\rangle + \sqrt{1-p}\,|1\rangle\,|\cdots\rangle .$$

The basic idea of the algorithm first of all involves preparing a quantum state of the form:

$$\beta\,|0\rangle\,|X_k\rangle + \gamma\,|1\rangle\,|0\rangle\,|a_t\rangle \text{ , with } \beta^2 + \gamma^2 = 1.$$

Applying a SWAP gate between the first two qubits of such a state we get:

$$|0\rangle\,(\beta\sqrt{p}\,|0\rangle\,|x_k\rangle + \gamma\,|1\rangle\,|a_t\rangle) + \beta\sqrt{1-p}\,|1\rangle\,|0\rangle\,|\cdots\rangle \tag{3.14}$$

For the next step the algorithm deploys the following unitary operator:

$$U_t = \begin{bmatrix} I_n - |a_t\rangle\langle a_t| & |a_t\rangle\langle a_t| \\ |a_t\rangle\langle a_t| & I_n - |a_t\rangle\langle a_t| \end{bmatrix}$$

An equivalent expression for this operator is the following:

$$U_t = I_2 \otimes (I_n - |a_t\rangle\langle a_t|) + X \otimes (|a_t\rangle\langle a_t|) \tag{3.15}$$

By substituting the below equality:

$$V_t |0\rangle = |a_t\rangle \Rightarrow |a_t\rangle\langle a_t| = V_t |0\rangle\langle 0| V_t^\dagger$$

the expression (3.15) can alternatively be written as follows:

$$U_t = (I_2 \otimes V_t)\left(I_2 \otimes (I_n - |0\rangle\langle 0|) + X \otimes |0\rangle\langle 0|\right)\left(I_2 \otimes V_t^{\dagger}\right) \tag{3.16}$$

Using (3.15), we find that the application of the operator U_t on the first part of the quantum state of (3.14) produces the quantum state which is calculated below:

$$\begin{aligned} & U_t\left(\beta\sqrt{p}\,|0\rangle\,|x_k\rangle + \gamma\,|1\rangle\,|a_t\rangle\right) = \\ & = |0\rangle \otimes \left(\beta\sqrt{p}\,(I_n - |a_t\rangle\langle a_t|)\,|x_k\rangle + \gamma\,|a_t\rangle\langle a_t|\,|a_t\rangle\right) + \\ & + |1\rangle \otimes \left(\beta\sqrt{p}\,|a_t\rangle\langle a_t|\,|x_k\rangle + \gamma\,(I_n - \langle a_t|\,|a_t\rangle)\,|a_t\rangle\right) = \\ & = |0\rangle \otimes \left(\beta\sqrt{p}\,(I_n - |a_t\rangle\langle a_t|)\,|x_k\rangle + \gamma\,|a_t\rangle\right) + |1\rangle \otimes \left(\beta\sqrt{p}\,\langle a_t|x_k\rangle\,|a_t\rangle\right) \end{aligned} \tag{3.17}$$

From the definition of the row iteration method in (3.10) it can be observed that in terms of quantum states:

$$|x_{k+1}\rangle \propto \|x_k\|\,(I_n - |a_t\rangle\langle a_t|)\,|x_k\rangle + b_t\,|a_t\rangle$$

and this expression resembles the first part of (3.17) if the values of β, γ are chosen appropriately. For example, the values $\beta = \|x_k\|$ and $\gamma = b_t\sqrt{p}$ are two possible options.

The above analysis is a quite intuitive explanation of the concept of the quantum row iteration algorithm. The subsequent part of this subsection includes an analytical presentation of the sequence of steps of this algorithm. In this regard:

1. As this is an iterative algorithm, an initial approximation $|x_0\rangle$ of the unit vector of the solution is selected. The iteration number k is set to 0 and a variable v_k is set to 1. The state of the quantum system is expressed as follows:

$$|X_k\rangle = \frac{\|x_k\|}{v_k}\,|0\rangle^{\otimes k}\,|x_k\rangle + |0^{\perp}\rangle^{\otimes k}\,|\cdots\rangle \tag{3.18}$$

2. The row with index $t_k \in \{1, \cdots, n\}$, which is used for each iteration, is chosen in order to prepare a state of the form:

$$|Y_k\rangle = \beta_{t_k}\,|0\rangle\,|X_k\rangle + \gamma_{t_k}\,|1\rangle\,|0\rangle^{\otimes k}\,|a_{t_k}\rangle$$

where $\beta_{t_k}^2 = \frac{v_k^2}{v_k^2 + b_{t_k}^2}$ and $\gamma_{t_k}^2 = 1 - \beta_{t_k}^2$. To implement this state, the transformation $|00\rangle \rightarrow (\beta_{t_k}\,|0\rangle + \gamma_{t_k}\,|1\rangle)\,|0\rangle$ is achieved with a suitable rotation operator, whereas $|X_k\rangle$ and $|a_{t_k}\rangle$ are incorporated using appropriate controlled operations.

3. A SWAP operator exchanges the first with the $(k+1)$-th qubit of $|Y_k\rangle$ and an operator of the form $I_2^{\otimes k} \otimes U_{t_k}$ is applied:

$$|X_{k+1}\rangle = (I_2^{\otimes k} \otimes U_{t_k})\text{swap}_{1,k+1} |Y_k\rangle \Rightarrow$$

$$|X_{k+1}\rangle = (I_2^{\otimes k} \otimes U_{t_k})\text{swap}_{1,k+1} \left(\beta_{t_k} |0\rangle |X_k\rangle + \gamma_{t_k} |1\rangle |0\rangle^{\otimes k} |a_{t_k}\rangle\right) \Rightarrow$$

$$|X_{k+1}\rangle = (I_2^{\otimes k} \otimes U_{t_k})\text{swap}_{1,k+1} \left(\frac{\beta_{t_k}\|x_k\|}{v_k} |0\rangle^{\otimes k} |0\rangle |x_k\rangle + \gamma_{t_k} |1\rangle |0\rangle^{\otimes k} |a_{t_k}\rangle + |0^{\perp}\rangle^{\otimes(k+1)} |\cdots\rangle\right) \Rightarrow$$

$$|X_{k+1}\rangle = |0\rangle^{\otimes(k+1)} \otimes \left(\frac{\beta_{t_k}\|x_k\|}{v_k} (I_n - |a_{t_k}\rangle \langle a_{t_k}|) |x_k\rangle + \gamma_{t_k} (|a_{t_k}\rangle \langle a_{t_k}|) |a_{t_k}\rangle\right) + |0^{\perp}\rangle^{\otimes(k+1)} |\cdots\rangle \Rightarrow$$

$$|X_{k+1}\rangle = |0\rangle^{\otimes(k+1)} \otimes \left(\frac{\beta_{t_k}\|x_k\|}{v_k} (I_n - |a_{t_k}\rangle \langle a_{t_k}|) |x_k\rangle + \gamma_{t_k} |a_{t_k}\rangle\right) + |0^{\perp}\rangle^{\otimes(k+1)} |\cdots\rangle \tag{3.19}$$

From the previous step we have $\beta_{t_k}^2 = \frac{v_k^2}{v_k^2+b_{t_k}^2}$ and $\gamma_{t_k}^2 = 1 - \beta_{t_k}^2 = \frac{b_{t_k}^2}{v_k^2+b_{t_k}^2}$, so $b_{t_k}^2\beta_{t_k}^2 = \gamma_{t_k}^2 v_k^2 \Rightarrow \gamma_{t_k} = \frac{\beta_{t_k} b_{t_k}}{v_k}$. As a result, (3.19) can alternatively be written as follows:

$$|X_{k+1}\rangle = \frac{\beta_{t_k}}{v_k} |0\rangle^{\otimes(k+1)} \otimes (\|x_k\| (I_n - |a_{t_k}\rangle \langle a_{t_k}|) |x_k\rangle + b_{t_k} |a_{t_k}\rangle) + |0^{\perp}\rangle^{\otimes(k+1)} |\cdots\rangle \Rightarrow$$

$$|X_{k+1}\rangle = \frac{\|x_{k+1}\|}{v_{k+1}} |0\rangle^{\otimes(k+1)} |x_{k+1}\rangle + |0^{\perp}\rangle^{\otimes(k+1)} |\cdots\rangle \text{ with } v_{k+1} = \frac{v_k}{\beta_{t_k}}$$

It can be observed that the produced state has the form of (3.18), thus an iteration of the algorithm is completed at this point. To proceed to the next iteration of the algorithm, we just need to set $k = k + 1$ and go to step 2.

3.6.3 Quantum column iteration method

Similarly to the row iteration method, for the column iteration method we assume that there is a unitary operator S_j which is appropriate for the preparation of the column vectors $|c_j\rangle$:

$$S_j^{\dagger} |j\rangle = |c_j\rangle$$

These columns are scaled to $\|c_j\| = 1$ so that they can be represented as quantum states. The first step of the algorithm requires an approximation of the solution x_0 which is also used for the calculation of the initial residual $r_0 = b - Ax_0$. Afterwards the algorithm updates the respective quantum states $|x_k\rangle$ and $|r_k\rangle$ as follows during each iteration:

$$|x_{k+1}\rangle \propto \|x_k\| |x_k\rangle + \|r_k\| |t_k\rangle \langle c_{t_k}|r_k\rangle = \|x_k\| |x_k\rangle + \|r_k\| |t_k\rangle \langle t_k|S_{t_k}|r_k\rangle$$

$$|r_{k+1}\rangle \propto \|r_k\|(I_n - |c_{t_k}\rangle \langle c_{t_k}|)\,|r_k\rangle \tag{3.20}$$

Throughout the algorithm $|x_k\rangle$ and $|r_k\rangle$ are included in quantum states $|X_k\rangle$ and $|R_k\rangle$ respectively which have the form:

$$|R_k\rangle = \|r_k\|\,|0\rangle^{\otimes k}\,|r_k\rangle + |0^{\perp}\rangle^{\otimes k}\,|\cdots\rangle$$

$$|X_k\rangle = \frac{\|x_k\|}{k+1}\,|0\rangle^{\otimes 2k}\,|x_k\rangle + |0^{\perp}\rangle^{\otimes 2k}\,|\cdots\rangle \tag{3.21}$$

The update of $|r_k\rangle$ is achieved with a combination of a SWAP operator which exchanges the first with the $(k+1)$-th qubit and a $I_2^{\otimes k} \otimes U_{t_k}$ operator. These operators are actually applied to the state $|0\rangle\,|R_k\rangle$ as illustrated below:

$$|R_{k+1}\rangle = \left(I_2^{\otimes k} \otimes U_{t_k}\right) \text{swap}_{1,k+1}\,|0\rangle\,|R_k\rangle \Rightarrow$$

$$|R_{k+1}\rangle = |0\rangle^{\otimes(k+1)}\,\|r_k\|\,(I_n - |c_{t_k}\rangle \langle c_{t_k}|)\,|r_k\rangle + |0^{\perp}\rangle^{\otimes(k+1)}\,|\cdots\rangle \Rightarrow$$

$$|R_{k+1}\rangle = \|r_{k+1}\|\,|0\rangle^{\otimes(k+1)}\,|r_k\rangle + |0^{\perp}\rangle^{\otimes(k+1)}\,|\cdots\rangle$$

This resembles the update of the solution vector $|x_k\rangle$ in the row iteration algorithm if we set $v_k = 1$. On the other hand, the update of $|x_k\rangle$ in the column iteration algorithm follows a different procedure. To illustrate this, we assume that the initial approximations of the solution and the residual are represented as follows:

$$|\tilde{R}_k\rangle = \|r_k\|\,|0\rangle\,S_{t_k}\,|r_k\rangle + |0^{\perp}\rangle\,|\cdots\rangle$$

$$|\tilde{X}_k\rangle = \frac{\|x_k\|}{v}\,|0\rangle\,|x_k\rangle + |0^{\perp}\rangle\,|\cdots\rangle$$

Using two ancilla qubits we combine these quantum states in a state of the form:

$$|\phi_1\rangle = \beta\,|00\rangle\,|\tilde{X}_k\rangle + \gamma\,|10\rangle\,|\tilde{R}_k\rangle \text{ with } \beta^2 + \gamma^2 = 1$$

At this point we apply a W_{t_k} operator, where:

$$W_t = \begin{bmatrix} I_n & 0 & 0 \\ 0 & I_n - |t\rangle\langle t| & |t\rangle\langle t| \\ 0 & |t\rangle\langle t| & I_n - |t\rangle\langle t| \end{bmatrix}$$

as well as a SWAP operator which exchanges the second with the third qubit. The resulting state is:

$$|\phi_2\rangle = \text{swap}_{2,3} W_{t_k}\,|\phi_1\rangle \Rightarrow$$

$$|\phi_2\rangle = \text{swap}_{2,3} W_{t_k} \left(\beta |00\rangle |\tilde{X_k}\rangle + \gamma |10\rangle |\tilde{R_k}\rangle \right)$$

$$|\phi_2\rangle = \text{swap}_{2,3} \left(\beta |00\rangle |\tilde{X_k}\rangle + \gamma |01\rangle |t_k\rangle \langle t_k| |\tilde{X_k}\rangle + |10\rangle |\cdots\rangle \right) \Rightarrow$$

$$|\phi_2\rangle = |00\rangle \left(\frac{\beta \|x_k\|}{v} |0\rangle |x_k\rangle + \gamma \|r_k\| |1\rangle |t_k\rangle \langle t_k| S_{t_k} |r_k\rangle \right) + |0^{\perp}\rangle^{\otimes 2} |\cdots\rangle$$

The first part of $|\phi_2\rangle$ is similar to (3.20),but in order to compute $|x_{k+1}\rangle$ we need to rotate its third qubit with an appropriate rotation operator of the form:

$$G_k = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \text{ with } c^2 + s^2 = 1$$

This leads to the following state:

$$|\phi_3\rangle = (I_4 \otimes G_k \otimes I_n) |\phi_2\rangle \Rightarrow$$

$$|\phi_3\rangle = |000\rangle \left(\frac{c\beta \|x_k\|}{v} |x_k\rangle + s\gamma \|r_k\| |t_k\rangle \langle t_k| S_{t_k} |r_k\rangle \right) + |0^{\perp}\rangle^{\otimes 3} |\cdots\rangle \tag{3.22}$$

Combining (3.20) and (3.21), we expect the quantum state of $|X_{k+1}\rangle$ to be:

$$|X_{k+1}\rangle = \frac{1}{k+2} \left(\|x_k\| |x_k\rangle + \|r_k\| |t_k\rangle \langle t_k| S_{t_k} |r_k\rangle \right) + |0^{\perp}\rangle^{\otimes (2k+2)} |\cdots\rangle .$$

Comparing this state with (3.22) and setting $v = k + 1$, we find that $\frac{c\beta}{k+1} = s\gamma = \frac{1}{k+2}$. Shao et al propose $c = \beta = \sqrt{\frac{k+1}{k+2}}$ and $s = \gamma = \sqrt{\frac{1}{k+2}}$ for the implementation of their algorithm.

At this point the reader should have sufficient understanding of how the quantum column iteration algorithm works. The rest of this subsection summarizes and analyzes its main steps. In particular:

1. The algorithm begins with an initial approximation $|x_0\rangle$ of the unit vector of the solution. Based on that, an initial residual $r_0 = b - Ax_0$ is calculated. The iteration number k is set to 0 and the quantum states $|X_k\rangle$ and $|R_k\rangle$ are prepared.

2. The column with index $t_k \in \{1, \cdots, n\}$, which is used for each iteration, is chosen in order to prepare a state of the form:

$$|\psi_0\rangle = \sqrt{\frac{k+1}{k+2}} |00\rangle |X_k\rangle + \sqrt{\frac{1}{k+2}} |10\rangle (I^{\otimes 2k} \otimes S_{t_k}) |0\rangle^{\otimes k} |R_k\rangle$$

3. Two SWAP operators exchange the second with the $(k+2)$-th qubit and the first with the $(k+1)$-th qubit of the state $|\psi_0\rangle$. Afterwards, in order to update $|X_k\rangle$, an operator of

the form $\left(I_2^{\otimes(2k+1)} \otimes G_k \otimes I_n\right)\left(I_2^{\otimes 2k} \otimes W_{t_k}\right)$ is applied. To provide more insight into this process, by substituting $|X_k\rangle$ and $|R_k\rangle$ we can alternatively write $|\psi_0\rangle$ as follows:

$$|\psi_0\rangle = \sqrt{\frac{k+1}{k+2}}\,|00\rangle\left(\frac{\|x_k\|}{k+1}\,|0\rangle^{\otimes 2k}\,|x_k\rangle + |0^{\perp}\rangle^{\otimes 2k}\,|\cdots\rangle\right) +$$

$$+\sqrt{\frac{1}{k+2}}\,|10\rangle\left(\|r_k\|\,|0\rangle^{\otimes 2k}\,S_{t_k}\,|r_k\rangle + |0^{\perp}\rangle^{\otimes 2k}\,|\cdots\rangle\right)$$

The use of the SWAP gates leads to the state $|\psi_1\rangle$:

$$|\psi_1\rangle = |0\rangle^{\otimes 2k}\otimes\left(\sqrt{\frac{k+1}{k+2}}\frac{\|x_k\|}{k+1}\,|00\rangle\,|x_k\rangle + \sqrt{\frac{1}{k+2}}\|r_k\|\,|10\rangle\,S_{t_k}\,|r_k\rangle\right)+|0^{\perp}\rangle^{\otimes 2k}\,|\cdots\rangle$$

The operator $I_2^{\otimes 2k} \otimes W_{t_k}$ transforms the previous state to a new state $|\psi_2\rangle$:

$$|\psi_2\rangle = |0\rangle^{\otimes(2k+1)}\otimes\left(\sqrt{\frac{k+1}{k+2}}\frac{\|x_k\|}{k+1}\,|0\rangle\,|x_k\rangle + \sqrt{\frac{1}{k+2}}\|r_k\|\,|1\rangle\,|t_k\rangle\,\langle t_k|S_{t_k}|r_k\rangle\right)+|0^{\perp}\rangle^{\otimes(2k+1)}\,|\cdots\rangle$$

Finally the operator $I_2^{\otimes(2k+1)} \otimes G_k \otimes I_n$ leads to a state $|\psi_3\rangle$ with the form:

$$|\psi_3\rangle = |0\rangle^{\otimes(2k+2)}\otimes\left(\sqrt{\frac{k+1}{k+2}}\frac{\|x_k\|}{k+1}\sqrt{\frac{k+1}{k+2}}\,|x_k\rangle + \sqrt{\frac{1}{k+2}}\sqrt{\frac{1}{k+2}}\|r_k\|\,|t_k\rangle\,\langle t_k|S_{t_k}|r_k\rangle\right)$$

$$+\,|0^{\perp}\rangle^{\otimes(2k+2)}\,|\cdots\rangle \Rightarrow$$

$$|\psi_3\rangle = |0\rangle^{\otimes(2k+2)}\otimes\left(\frac{\|x_k\|}{k+2}\,|x_k\rangle + \frac{1}{k+2}\|r_k\|\,|t_k\rangle\,\langle t_k|S_{t_k}|r_k\rangle\right)$$

$$+\,|0^{\perp}\rangle^{\otimes(2k+2)}\,|\cdots\rangle$$

and it can be observed that $|\psi_3\rangle = |X_{k+1}\rangle$.

4. The state of $|R_k\rangle$ is updated as it is explained above, namely using the same operators which are used for the update of $|X_k\rangle$ in the row iteration algorithm. At this point, an iteration of the algorithm is completed, so we set $k = k + 1$ and go to step 2.

3.6.4 Review of the algorithms

The row and column iteration algorithms are an interesting idea for a quantum linear solver. Their main advantage compared to the other iterative methods which are known from classical computing is the fact that in order to solve a linear system of the form $A\vec{x} = \vec{b}$ neither the whole matrix A nor the vector $\vec{b}$ need to be represented in quantum states. On the contrary, every iteration requires only one row or one column of A and one element of $\vec{b}$.

Moreover, the algorithms only need a QRAM for the preparation of the quantum states and unlike the HHL algorithm there are no requirements for Hamiltonian simulations or phase estimation.

The performance of the quantum row and column iteration algorithms is approximately $\mathrm{O}\left(\kappa_s^2(\log n)\log\frac{1}{\epsilon}\right) \approx \mathrm{O}(T\log n)$, where κ_s is the scaled condition number:

$$\kappa_s = \|A\|_F\|A^{-1}\|$$

and $\|A\|_F$ is the Frobenius norm of A:

$$\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}|a_{ij}|^2}.$$

This is comparable to the other quantum solvers like HHL and also shows the supremacy of the algorithms in comparison to their classical counterparts whose performance is approximately $\mathrm{O}\left(\kappa_s^2 n\log\frac{1}{\epsilon}\right)$.

On the other hand, the algorithms also have one limitation, namely their dependence on ancilla qubits which are used during step 2 of both row and column algorithms. It can be observed that this leads to a linear increase of the qubit requirements with the number of iterations, which might be a problem if a lot of iterations are necessary in order to achieve high accuracy.

3.7 Variational quantum linear solver

3.7.1 Outline of the algorithm

All the previous quantum linear solvers involve quantum supremacy and promise to solve linear systems faster than their classical counterparts. However, in practice they have not been implemented in actual quantum computers for large-scale systems of equations. This is due to the fact that currently only noisy intermediate-scale quantum (NISQ) computers have been created which contain a limited number of qubits and are susceptible to quantum decoherence, which means that they tend to deviate from the standard theory of quantum mechanics. This is a significant limitation, especially when error correction methods cannot easily be applied to such quantum systems.

Large-scale quantum computers are not expected to emerge soon, so we are forced to lower our expectations concerning the immediate application of the aforementioned quantum

algorithms for linear systems and resort to more realistic options. In this regard, Bravo-Prieto proposed the variational quantum linear solver (VQLS) [4], a hybrid quantum - classical algorithm which current quantum computers can handle. The concept of this algorithm is to incorporate classical computations in order to reduce the complexity of the quantum circuits which are used.

In this subsection I focus on a short description of the structure of the VQLS algorithm which is summarized in the following figure:

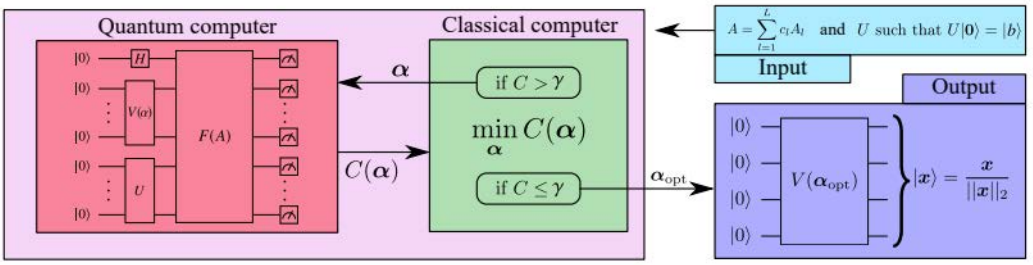

Figure 3.5: Diagram of the VQLS algorithm ([4])

Considering a linear system of the form $A\vec{x} = \vec{b}$, the input to the algorithm is a combination of gates U such that $U|0\rangle = |b\rangle$, where $|b\rangle$ is proportional to $\vec{b}$. In addition, the input includes the matrix A in the form of a linear combination of unitary matrices A_l:

$$A = \sum_{l=1}^{L} c_l A_l, c_l \in \mathbf{C}.$$

The target is to find $|x\rangle$ that is proportional to the solution $\vec{x}$ by using an ansatz - an educated guess - for the combination of gates $V(a)$ which are used for an approximation of the solution $|x(a)\rangle$ such that:

$$|x(a)\rangle = V(a)|0\rangle .$$

The parameters a are inputs from the classical computer to the quantum computer and the latter calculates $|x(a)\rangle$, as well as the value of a cost function $C(a)$. The calculated $C(a)$ is sent to the classical computer which runs an optimization algorithm that changes the values of a in order to minimize the cost function. The new a is sent to the quantum computer and the whole process is repeated until an optimal value $a = a_{opt}$ is found which leads to a sufficient approximation of the solution:

$$|x\rangle \approx |x(a_{opt})\rangle = V(a_{opt})|0\rangle$$

and also $C(a_{opt}) \leq \gamma$ is satisfied where γ is a termination parameter that acts as a threshold.

3.7.2 Implementation details

In this subsection more details about the implementation of the VQLS algorithm are discussed. First of all, the form of the cost function needs to be defined. In particular, it is designed to measure the component of $|\Phi\rangle = A|x\rangle$ which is orthogonal to $|b\rangle$. The authors of the paper propose four different cost functions:

1. The global cost functions include two similar cost functions:

   - $$\hat{C_G} = Tr(|\Phi\rangle\langle\Phi|(I - |b\rangle\langle b|)) = \langle x|H_G|x\rangle$$

     where the matrix H_G is defined as follows:

     $$H_G = A^\dagger(I - |b\rangle\langle b|)A$$

   - The second cost function is a normalized version of the previous one:

     $$C_G = \hat{C_G}/\langle\Phi|\Phi\rangle$$

2. The local cost functions also include two cost functions:

   - $$\hat{C_L} = \langle x|H_G|x\rangle$$

     where the matrix H_L is defined as follows:

     $$H_L = A^\dagger U\left(1 - \frac{1}{n}\sum_{j=1}^{n}|0_j\rangle\langle 0_j| \otimes I_{\bar{j}}\right)U^\dagger A$$

     and $|0_j\rangle$ is the zero state on the j-th qubit, whereas $I_{\bar{j}}$ is the identity matrix which excludes the j-th qubit.

   - The second cost function is again a normalized version of the previous one:

     $$C_L = \hat{C_L}/\langle\Phi|\Phi\rangle$$

All the above cost functions are suitable for the solution of a $n \times n$ system, but according to the writers, the local cost functions lead to better convergence for large values of n. It can be proved that:

$$C_G \geq \frac{\epsilon^2}{\kappa^2}, C_L \geq \frac{1}{n}\frac{\epsilon^2}{\kappa^2}$$

where κ is the condition number of the matrix A and ϵ is the precision parameter. The right part of these inequalities can be used to define the termination parameter γ.

As far as the gate sequence $V(a)$ is concerned - which is used to compute $|x(a)\rangle$ -, mathematically it can be expressed as follows:

$$V(a) = G_{k_1}(a_1)G_{k_2}(a_2)\cdots G_{k_l}(a_l)$$

where k_1, k_2 etc. correspond to the gate types and their position in the quantum circuit. In cases where $k_1 = k_2 = \cdots = k_l$, the quantum circuit has a fixed structure and the optimization only adjusts the values of the parameters a. This is a simple strategy, however, according to the paper it does not work well for large-scale systems, so it should be avoided. As an alternative, the paper proposes dividing the circuit into layers and performing layer-by-layer optimization.

Finally another noteworthy element of the VQLS method is the training algorithm that the classical computer uses in order to minimize $C(a)$. There are a few different approaches to achieve that. For example, the writers propose a method which in each iteration randomly selects a direction w in the parameter space and solves the optimization problem $min_{s\in\mathbf{R}}C(a + sw)$ in order to find the most suitable value for s.

3.7.3 Review of the algorithm

The performance of the VQLS algorithm was evaluated by the writers by solving up to 1024×1024 linear systems using Rigetti's Quantum Cloud Services. Thus, the complexity of the algorithm which I present here is based on experimental data and not theoretical formulas. According to the results of the paper, the time that a linear system needs to be solved has a close to linear dependence on the condition number κ, a logarithmic dependence on the inverse of the precision parameter $\frac{1}{\epsilon}$ and a logarithmic dependence on the size n of the system. Therefore, the overall complexity of the algorithm is approximately $\mathrm{O}(\kappa \log(\frac{1}{\epsilon}) \log n)$.

Although the calculated complexity is not so reliable, as it is based on a limited number of specific experiments, it is sufficient to show that VQLS leads to an exponential speed-up in n compared to the classical solvers of linear systems. This result is important and impressive if we consider that it can be achieved using implementations in the currently available flawed quantum computers.

3.8 Random walks solver

3.8.1 Classical random walks

The use of random walks is a quite different approach for the solution of linear systems than the ones which were presented before. The systems which this method handles have the form $\vec{x} = H\vec{x} + \vec{b}$ where H is a $n \times n$ matrix. This is equivalent to the classical form $A\vec{x} = \vec{b}$ if we set $A = I - H$. From the well-known Neumann series, if $\|H\| < 1$, then:

$$I + H + H^2 + H^3 + \cdots = \sum_{i=0}^{\infty} H^i = (I - H)^{-1} = A^{-1}$$

and as a result, the solution vector can be calculated as follows:

$$\vec{x} = A^{-1}\vec{b} = (I - H)^{-1}\vec{b} = \sum_{i=0}^{\infty} H^i\vec{b}. \tag{3.23}$$

The implementation of the random walks relies on Markov chains which help with the estimation of the solution vector. These Markov chains are expressed with a $n \times n$ matrix whose element P_{ij} is equal to the probability of the transition from the i-th row to the j-th row. Mathematically this means that given a random walk $\gamma = (i_0, i_1, \cdots, i_k)$ which terminates after k steps, then:

$$P(i_{n+1} = j | i_n = i, n < k) = P_{ij}$$

There are a few ways to construct the matrix P and to define how the corresponding random walk terminates:

- Neumann and Ulam [19] suggested that the matrix P meets the following requirements:

$$P_{ij} \geq 0, \sum_j P_{ij} \leq 1, H_{ij} \neq 0 \rightarrow P_{ij} \neq 0.$$

  In addition, a vector T is defined which contains the termination probabilities:

$$T_i = 1 - \sum_j P_{ij}.$$

  After a random walk $\gamma = (i_0, i_1, \cdots, i_k)$ finishes, an approximation of the i_0-th element of the solution vector can be calculated using the formula:

$$\tilde{x_{i_0}} = \frac{H_{i_0 i_1} \cdots H_{i_{k-1} i_k} b_{i_k}}{P_{i_0 i_1} \cdots P_{i_{k-1} i_k} T_{i_k}}$$

- Dimov et al [20] proposed that $P_{ij} = \frac{|H_{ij}|}{\sum_j |H_{ij}|}$. In this case, given a random walk $\gamma = (i_0, i_1, \cdots)$, there are no termination probabilities, but some weights W_i are used instead, such that:

$$W_0 = 1, W_1 = W_0 \frac{H_{i_0 i_1}}{P_{i_0 i_1}} \cdots, W_n = W_{n-1} \frac{H_{i_{n-1} i_n}}{P_{i_{n-1} i_n}}.$$

The random walk finishes after its k-th step if $|W_k| < \epsilon$, where ϵ is a predefined tolerance. After that an approximation of the i_0-th element of the solution vector is calculated as follows:

$$\tilde{x_{i_0}} = \sum_{n=0}^{k} W_n b_{i_n}$$

3.8.2 Outline of the algorithm

A hybrid classical-quantum linear solver which utilizes random walks was proposed by Chen et al [5]. This algorithm implements the random walks in a quantum computer and performs the computations of the approximations of the solution vector in a classical computer. This subsection does not include an analytical explanation of this method, but a general description of its structure.

Given a $n \times n$ linear system, $A\vec{x} = \vec{b}$, the proposed algorithm diverges a little from the random walk methodology described in the previous subsection, as an additional parameter γ is inserted, such that:

$$A = I - \gamma P, 0 < \gamma < 1$$

where P is a Markov chain transition matrix that satisfies $P_{ij} \geq 0, \sum_j P_{ij} = 1$. Consequently, (3.23) is transformed as follows:

$$\vec{x} = \sum_{s=0}^{\infty} \gamma^s P^s \vec{b} \tag{3.24}$$

To approximate the I_0-th element of x, a truncated version of (3.24) is used that has the form:

$$x_{I_0}^{(c)} = \sum_{s=0}^{c} \gamma^s \sum_{I_1, I_2, \cdots, I_s = 1}^{n} P_{I_0 I_1} \cdots P_{I_{s-1} I_s} b_{I_s} \tag{3.25}$$

The proposed algorithm is based on this formula, which is evaluated by a quantum random walk on a graph consisting of n nodes. As far as the value of c is concerned, it inserts an error of $\mathrm{O}(\gamma^c)$ in the calculation of $\vec{x}$. Thus, given a tolerance of ϵ, c should be selected to be approximately $\log\left(\frac{1}{\epsilon}\right) \log\left(\frac{1}{\gamma}\right)$.

Contrary to the previous quantum solvers which were presented in this thesis, the form of this algorithm does not require the vectors $\vec{x}$ and $\vec{b}$ to be represented as quantum states, thus these data can remain as classical data. Only the matrix A - or equivalently the matrix P - needs to be appropriately inserted in quantum registers. The writers of the paper suggest the Hamming cube structure [21] to encode the n-node graph that corresponds to the transition matrix P. In a few words, this means that they use $m = \log_2 n$ qubits for this graph by associating each node $J \in \{0, 1, \cdots, n-1\}$ with a binary string $|J\rangle = |j_{m-1}, \cdots, j_1, j_0\rangle$, where j_l is a binary digit. For example, 2 qubits are sufficient for $n = 4$, as the following figure illustrates:

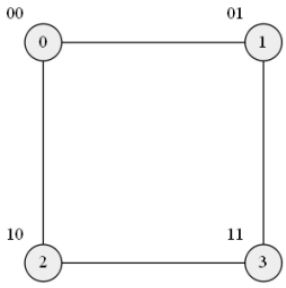

Figure 3.6: 4-node graph with the Hamming cube structure ([5])

Apart from the m qubits that are needed for the construction of the graph, the proposed implementation requires an additional qubit which acts as a coin register. Given a node represented as a m-bit binary string $(j_{m-1}, \cdots, j_1, j_0)$, the initial quantum state of the algorithm is:

$$|\psi_{0,J}\rangle = |0\rangle_\diamond \otimes |j_{m-1}, \cdots, j_1, j_0\rangle_q = |0\rangle_\diamond \otimes |J\rangle_q .$$

The transitions from a node of the graph to another during the different steps of a random walk are achieved by flipping the bits j_i with appropriate probabilities. This functionality is based on the coin register; starting with j_0 and continuing with the rest of the graph qubits in sequence, the coin register is rotated by an angle θ_k and then a CNOT gate is applied to the graph qubit which is processed. Mathematically this is expressed with the operator:

$$\mathcal{U} = \prod_{k=0}^{m-1} (|0\rangle_\diamond \langle 0|_\diamond \otimes I_q + |1\rangle_\diamond \langle 1|_\diamond \otimes X_k)(U(u_k) \otimes I_q)$$

where the first parenthesis includes the expression for the CNOT gate, the second parenthesis includes the expression for the coin register rotation and the operator U is the general single

qubit gate:

$$U(u) = U(\theta, \phi, \lambda) = \begin{bmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\phi}\sin(\theta/2) & e^{i\lambda+i\phi}\cos(\theta/2) \end{bmatrix}$$

After flipping the coin register once for all the graph qubits, the resulting quantum state is the following:

$$\mathcal{U}\left|\psi_{0,J}\right\rangle = \sum_{i_{m-1},\cdots,i_0=0}^{1} \prod_{l=0}^{m-1} U(u_l)_{i_l,i_{l-1}} \left|i_{m-1}\right\rangle_{\diamond} \otimes \left|i_{m-1} \oplus j_{m-1}, \cdots, i_1 \oplus j_1, i_0 \oplus j_0\right\rangle_q$$

where $i_{-1} = 0$. Based on this state, the transition probabilities can be calculated as follows:

$$P_{JJ'} = \prod_{l=0}^{m-1} |U(u_l)_{i_l,i_{l-1}}|^2 = \prod_{l=0}^{m-1} |\cos^2(\frac{\theta_l}{2})|^{1-(i_l \oplus i_{l-1})} \prod_{l=0}^{m-1} |\sin^2(\frac{\theta_l}{2})|^{i_l \oplus i_{l-1}}$$

where $\left|I\right\rangle_q = \left|i_{m-1}, \cdots, i_1, i_0\right\rangle_q$ can be derived from $\left|J'\right\rangle_q = \left|I\right\rangle_q \oplus \left|J\right\rangle_q$. The calculation of these transition probabilities can be extended for multiple applications of the operator $\mathcal{U}$, but this leads to complex formulas whose thorough analysis is beyond the scope of this thesis. For a concise description of these formulas the interested reader can read [5].

The corresponding transition matrix for the graph shown in 3.6 is the following:

$$P = \begin{bmatrix} \cos^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right) & \sin^2\left(\frac{\theta_0}{2}\right)\sin^2\left(\frac{\theta_1}{2}\right) & \cos^2\left(\frac{\theta_0}{2}\right)\sin^2\left(\frac{\theta_1}{2}\right) & \sin^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right) \\ & \cos^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right) & \sin^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right) & \cos^2\left(\frac{\theta_0}{2}\right)\sin^2\left(\frac{\theta_1}{2}\right) \\ & & \cos^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right) & \sin^2\left(\frac{\theta_0}{2}\right)\sin^2\left(\frac{\theta_1}{2}\right) \\ & & & \cos^2\left(\frac{\theta_0}{2}\right)\cos^2\left(\frac{\theta_1}{2}\right) \end{bmatrix}$$

where the missing elements are omitted due to the symmetry of the matrix. It can be proved that P cannot be written as a tensor product of two independent 2×2 matrices, which means that the two qubits of the graph are entangled, hence the probability that the one flips depends on the probability that the other one flips. This observation can be extended for graphs that consist of more qubits. This is a key difference between this algorithm and its counterparts that run in a classical computer.

3.8.3 Review of the algorithm

The proposed random walks solver is a creative approach to solve a linear system using a quantum computer, as it exploits the probabilistic nature of the quantum states in order to implement a Markov chain structure. There are several reasons to opt for this type of algorithm. First of all, it does not have extravagant space requirements, as $1 + \log n$ qubits

are sufficient. In addition, according to the writers the time complexity for the computation of an element of the solution vector is $\mathrm{O}(\log n)$. It seems that they have probably not taken into account the additional cost of the classical operations, however, the algorithm still seems to outperform the classical methods, especially when we are interested only in part of the solution. Finally, contrary to the other quantum solvers, it is important to note that the output of this solver is classical data, so it more accessible and we can easily extract and read it.

3.9 Further research resources, summary and comparison of the algorithms

Apart from the already mentioned algorithms, there is a plethora of other resources related to linear systems and quantum computing and the interested reader can refer to them in order to acquire a more global view of the subject [22–43]. As far as the quantum solvers which are presented in this thesis are concerned, as the following table illustrates, all of them imply supremacy compared to the best performing classical linear solvers.

Algorithm	Complexity
Grover 's solver	$\mathrm{O}(\sqrt{n})$
HHL algorithm	$\mathrm{O}(s^2\kappa^2 \log n/\epsilon)$
WZP algorithm	$\mathrm{O}(\kappa^2 \log n \|\|A\|\|_F/\epsilon)$
Row and column iteration methods	$\mathrm{O}\left(\kappa_s^2 \log n \log \frac{1}{\epsilon}\right)$
VQLS algorithm	$\mathrm{O}(\kappa \log n \log \frac{1}{\epsilon})$
Random walks algorithm	$\geq \mathrm{O}(\log n)$ per x_i

Table 3.1: The complexity of the quantum solvers

More specifically, all of them except for the Grover's solver theoretically achieve an exponential speed-up in the size of the linear system. However, a lot depends on the condition number κ of the matrix A which sometimes scales polynomially with the size of the linear system. As a result, if A is ill-conditioned, the quantum algorithms may even be slower than the classical methods. This problem can partially be solved using several approaches like preconditioners, however it is often impossible to restrict κ in a scale logarithmic to the number of equations. Moreover, the dependence of the complexity of the algorithms on the preci-

sion parameter ϵ should also be taken into consideration. Expecting very accurate solutions is definitely not recommended, as this can lead to an arbitrary increase in the runtime of the algorithms.

The aforementioned observations can lead to some skepticism about the effectiveness of the quantum linear solvers, but their performance is anyway undoubtedly impressive. It is also encouraging that some of them, like the VQSL algorithm, have been tested, so this confirms that these solvers are not limited to theoretical studies which can not be practically implemented in quantum computers.

Chapter 4

Experimental Results

4.1 Introduction

In this chapter I present and analyze a practical implementation of the hybrid quantum - classical VQLS algorithm which was discussed in 3.7. The requisite mathematical formulas which are the fundamental components of this implementation are the following:

$$\begin{cases} A = \sum_{l=1}^{L} c_l A_l, c_l \in \mathbf{C} \\ |x(a)\rangle = V(a)\,|0\rangle \\ |\Phi\rangle = A\,|x(a)\rangle \\ U\,|0\rangle = |b\rangle \end{cases}$$

As it was explained in 3.7, given a quantum linear system of the form $A\,|x\rangle = |b\rangle$, the matrix A has to be written as a linear combination of unitary operators A_l, as the first formula indicates. The second formula introduces the ansatz $V(a)$ which transforms a set of qubits initialized to $|0\rangle$ to an approximation of the solution $|x(a)\rangle$. The goal of the VQLS algorithm is to construct the best possible $V(a)$ by finding the optimal a. The third formula calculates a quantum state $|\Phi\rangle$ which stems from the multiplication of A with the approximate solution $|x(a)\rangle$. Finally the last formula includes the operator U which is responsible for the preparation of $|b\rangle$ starting from a set of qubits initialized to $|0\rangle$. As far as the cost function is concerned, the one used for this implementation is the following:

$$C = \frac{\langle\Phi|\,(\mathbb{I} - |b\rangle\,\langle b|)\,|\Phi\rangle}{\langle\Phi|\Phi\rangle} = \frac{\langle\Phi|\Phi\rangle - \langle\Phi|b\rangle\,\langle b|\Phi\rangle}{\langle\Phi|\Phi\rangle} = 1 - \frac{|\,\langle b|\Phi\rangle\,|^2}{\langle\Phi|\Phi\rangle}$$

To calculate this in practice using a quantum computer, we have to decompose the terms $\langle\Phi|\Phi\rangle$ and $|\langle b|\Phi\rangle|^2$ as follows and apply the Hadamard test:

- $\langle\Phi|\Phi\rangle = \langle x(a)|A^\dagger A|x(a)\rangle = \langle 0|V(a)A^\dagger AV(a)|0\rangle = \sum_m \sum_n c_m^* c_n \langle 0|V(a)^\dagger A_m^\dagger A_n V(a)|0\rangle$
- $|\langle b|\Phi\rangle|^2 = |\langle b|Ax(a)\rangle|^2 = |\langle 0|U^\dagger AV(a)|0\rangle|^2 = \langle 0|U^\dagger AV(a)|0\rangle\langle 0|V(a)^\dagger A^\dagger U|0\rangle =$

$$= \sum_m \sum_n c_m^* c_n \langle 0|U^\dagger A_n V(a)|0\rangle \langle 0|V(a)^\dagger A_m^\dagger U|0\rangle$$

Given that all the involved operators exclusively include real numbers, the equality

$$\langle 0|U^\dagger A_i V(a)|0\rangle = \langle 0|V(a)^\dagger A_i^\dagger U|0\rangle$$

can be exploited, which simplifies the required calculations.

For the presented experiments, the coding for the quantum computation routines relies on the Qiskit open-source framework which was developed by IBM and allows access to both simulators and cloud-based quantum processors. Qiskit supports Python 3.7 or later versions and for the purposes of this thesis the included Aer simulators were used. The main inspiration of this part of my research has been the Qiskit tutorial focusing on applied quantum algorithms [44]. Similar to this tutorial, I try to solve simple 8×8 linear systems utilizing the VQLS algorithm. The main contributions of my work compared to the aforementioned tutorial are the following:

1. By coding and constructing quantum operator blocks, I reduce the qubit requirements for the inner product calculations compared to the corresponding circuits presented in the tutorial.

2. I utilize different optimizers included in the scipy Python library and compare them in order to discover which one produces the optimal a and by extension the best ansatz $V(a)$.

3. I propose an alternative ansatz structure to the one mentioned in the tutorial [44] and the paper [4] and achieve better convergence results for the examined linear systems.

4.2 Discussion of the results

First of all, the tutorial features Hadamard test circuits that include an ancilla qubit in order to efficiently incorporate the controlled operations. Coding the $V(a)$, U and A_i operators

by constructing block operators as illustrated in the figure below, this additional qubit can be omitted. In cases of more complex linear systems more than one ancilla qubits may be omitted using this technique, which is definitely a considerable improvement if we take into account the current small-scale quantum systems.

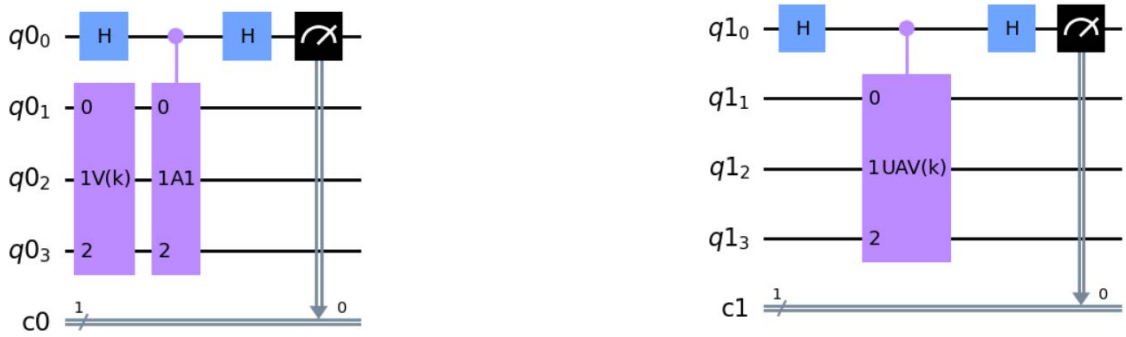

Figure 4.1: Inner product circuits for $\langle\Phi|\Phi\rangle$ (left) and $|\langle b|\Phi\rangle|^2$ (right)

The paper explains that the ansatz can take many different forms and presents an indicative fixed ansatz structure which includes alternating R_y gates with controlled-Z gates, as it is shown below:

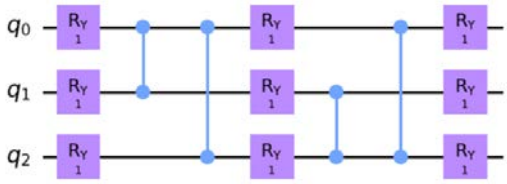

Figure 4.2: Inital ansatz structure (R_y, CZ gates)

The question which arises is which optimizer is the most appropriate for the calculation of the optimal values of a, at least provided that the ansatz has the above form. To answer this, I try to solve a linear system such that $A = 0.55\mathbb{I} + 0.45\mathbb{Z}_3$, where $\mathbb{I}$ is the identity operator and $\mathbb{Z}_3$ is an operator which includes a single qubit Z gate that acts on the third qubit, and $U = H^{\otimes 3}$, thus $|b\rangle = H^{\otimes 3}|000\rangle$. The optimizers which I experiment with stem from the scipy Python library and an interested reader can discover details about their theoretical background in [45–52]. The convergence of these optimizers is shown in the following plot:

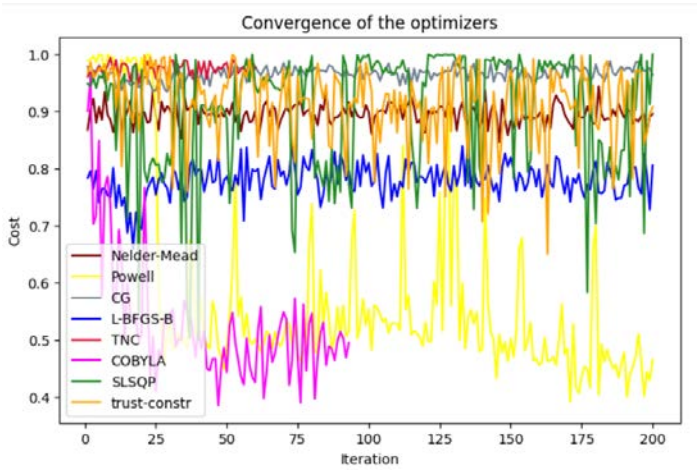

Figure 4.3: The convergence of the optimizers

It is evident that the best performing algorithm is the COBYLA optimizer. In a few words, this optimizer performs constrained optimization using linear approximation when the derivative of the target function is unknown. The Powell optimizer produces comparable results, but it requires a lot more iterations to achieve them and its behavior is very unstable. Furthermore, the accuracy of its calculated solution vector is not sufficient, as the residual plot below indicates:

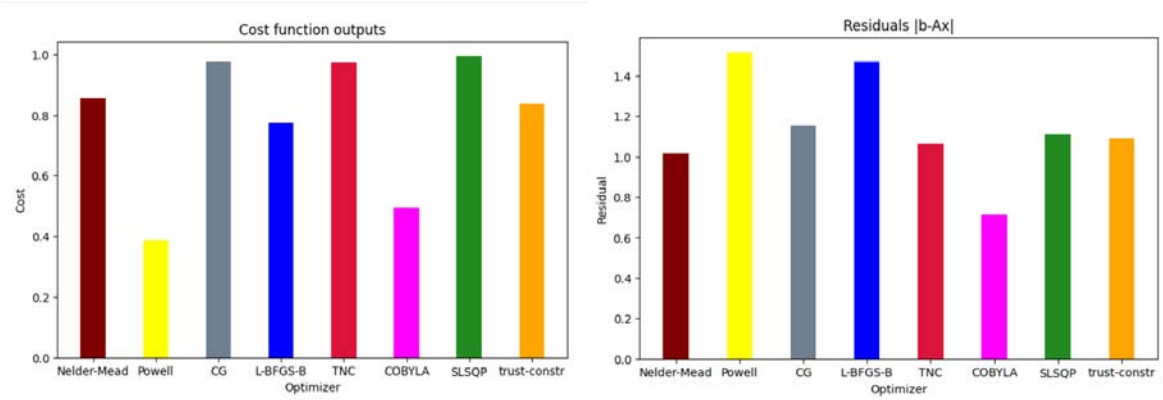

Figure 4.4: The cost function outputs of the optimizers (left) and the corresponding residuals |b-Ax| (right)

On the other hand, the use of the COBYLA optimizer leads to a quantum state that is a much better approximation of the solution. Its density matrix can be seen in 4.5 - white squares indicate positive values and black squares correspond to negative values, whereas the size of the squares is proportional to the magnitude of the values.

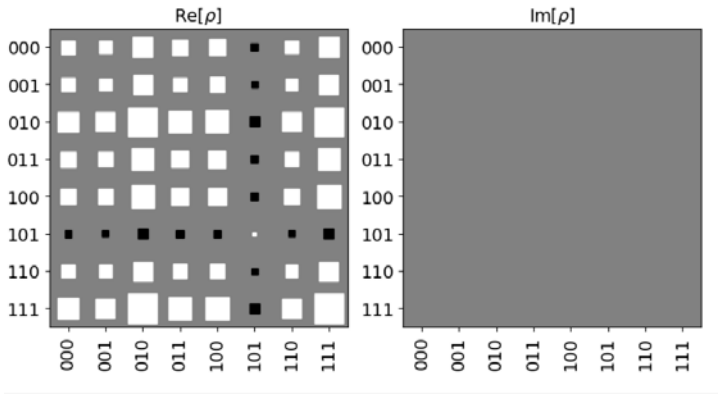

Figure 4.5: Density matrix of the COBYLA quantum solution vector

Further improving the performance of the VQLS algortithm largely depends on the structure of the ansatz. For this reason, I propose an ansatz consisting of four gates for each qubit - a $U(\theta, 0, 0)$, a $U(\theta, 0, \pi)$, a $U(\theta, \pi, 0)$ and a $U(\theta, \pi, \pi)$ gate, where U is the general single qubit unitary gate:

$$U(\theta, \lambda, \phi) = \begin{bmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\phi}\sin(\theta/2) & e^{i\lambda+i\phi}\cos(\theta/2) \end{bmatrix}$$

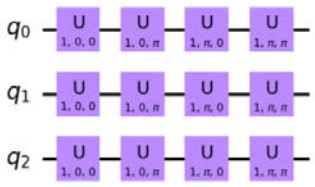

Figure 4.6: Proposed ansatz structure (U gates)

The values θ are the parameters which are tuned by the optimizer. This structure is suitable for linear systems that explicitly contain real numbers, as the selected gates cover all the single qubit gate cases whose operators do not include complex numbers. Trying to solve the aforementioned linear system using this ansatz, the resulting density matrix is illustrated in the following figure:

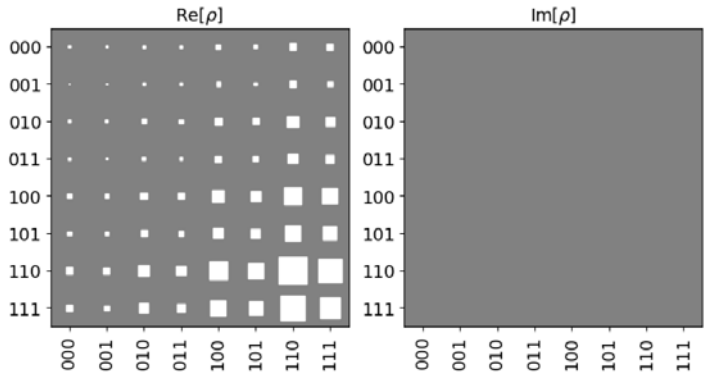

Figure 4.7: Density matrix of the quantum solution vector produced by the proposed ansatz

The results indicate a clear improvement in the performance of the algorithm, since the cost function converges much more efficiently:

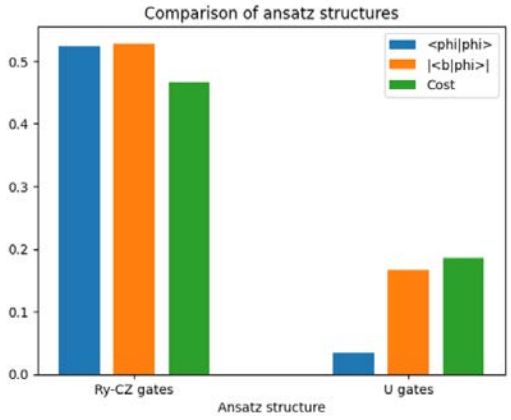

Figure 4.8: Comparison of the ansatz structures ($A = 0.55\mathbb{I} + 0.45\mathbb{Z}_3$)

The same applies for most of the linear systems which were tested. For instance, if we modify the above linear system so that $A = 0.3\mathbb{Z}_1 + 0.4\mathbb{Z}_2$, we find comparable results:

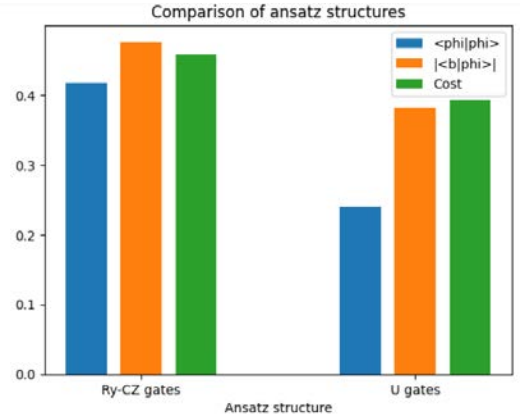

Figure 4.9: Comparison of the ansatz structures ($A = 0.3\mathbb{Z}_1 + 0.4\mathbb{Z}_2$)

Consequently, it seems that the proposed ansatz is worth considering for the solution of linear systems using the VQLS algorithm, as it serves as a reliable alternative to other quantum circuit structures which have been examined in practice and in theory.

Chapter 5

Synopsis and Prospects

The question arising from this research is whether the algorithms studied will possibly replace the well-known classical linear solvers, such as the Gaussian elimination. Much depends on whether the scientific community will be able to build universal quantum computers consisting of millions of qubits, and whether error correction techniques will be able to ensure the stability of these systems under all conditions.

The performance of these quantum linear solvers is undoubtedly promising, as they achieve even exponential speed-up in the size of the linear system compared to their classical counterparts. The dependence of most of them on the condition number of the involved matrices raises some concerns though. Therefore, possible future research efforts should focus on limiting or even eliminating this dependence. In addition, the inability to extract the whole solution vector from the output quantum states is something that cannot be overlooked.

The study of the presented algorithms is also still at a theoretical level and minimal practical implementations have been published which mainly concern small linear systems. In this regard, my thesis attempts to contribute to the enhancement of the performance of the VQLS algorithm from an experimental perspective by solving 8×8 linear systems using the Aer Qiskit simulators. The question is if these methods will successfully be extended in practice for large-scale systems. This is by no means certain, however, the current indications allow us to be optimistic.

Bibliography

[1] Rituparna Maji, Bikash Behera, and Prasanta Panigrahi. Solving Linear Systems of Equations by Using the Concept of Grover's Search Algorithm: An IBM Quantum Experience, 3 2017.

[2] X-D Cai, Christian Weedbrook, Z-E Su, M-C Chen, Mile Gu, M-J Zhu, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Experimental quantum computing to solve systems of linear equations. *Physical review letters*, 110(23):230501, 2013.

[3] Changpeng Shao and Hua Xiang. Row and column iteration methods to solve linear systems on a quantum computer. *Physical Review A*, 101(2):022322, 2020.

[4] Carlos Bravo-Prieto, Ryan LaRose, Marco Cerezo, Yigit Subasi, Lukasz Cincio, and Patrick J Coles. Variational quantum linear solver. *arXiv preprint arXiv:1909.05820*, 2019.

[5] Chih-Chieh Chen, Shiue-Yuan Shiau, Ming-Feng Wu, and Yuh-Renn Wu. Hybrid classical-quantum linear solver using noisy intermediate-scale quantum machines. *Scientific reports*, 9(1):16251, 2019.

[6] Charles Q Choi. Ibm's quantum leap: The company will take quantum tech past the 1,000-qubit mark in 2023. *IEEE Spectrum*, 60(1):46–47, 2023.

[7] Ray LaPierre and Ray LaPierre. Shor algorithm. *Introduction to Quantum Computing*, pages 177–192, 2021.

[8] Nicolaas P Landsman. Born rule and its interpretation. In *Compendium of quantum physics*, pages 64–70. Springer, 2009.

[9] Nicholas Young. *An introduction to Hilbert space*. Cambridge university press, 1988.

[10] Richard Jozsa. Fidelity for mixed quantum states. *Journal of modern optics*, 41(12):2315–2323, 1994.

[11] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for solving linear systems of equations . *Phys. Rev. Lett.*, 15(103), 2009.

[12] Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions. *arXiv preprint quant-ph/0208112*, 2002.

[13] Gerhard C Hegerfeldt and Dirk G Sondermann. Conditional hamiltonian and reset operator in the quantum jump approach. *Quantum and Semiclassical Optics: Journal of the European Optical Society Part B*, 8(1):121, 1996.

[14] Jian Pan, Yudong Cao, Xiwei Yao, Zhaokai Li, Chenyong Ju, Hongwei Chen, Xinhua Peng, Sabre Kais, and Jiangfeng Du. Experimental realization of quantum algorithm for solving linear systems of equations. *Physical Review A*, 89(2):022313, 2014.

[15] Leonard Wossnig, Zhikuan Zhao, and Anupam Prakash. Quantum linear system algorithm for dense matrices. *Physical review letters*, 120(5):050502, 2018.

[16] Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. *arXiv preprint arXiv:1603.08675*, 2016.

[17] Mario Szegedy. Quantum speed-up of markov chain based algorithms. In *45th Annual IEEE symposium on foundations of computer science*, pages 32–41. IEEE, 2004.

[18] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical review letters*, 100(16):160501, 2008.

[19] George E Forsythe and Richard A Leibler. Matrix inversion by a monte carlo method. *Mathematics of Computation*, 4(31):127–129, 1950.

[20] IT Dimov, TT Dimov, and TV Gurov. A new iterative monte carlo approach for inverse matrix problem. *Journal of Computational and Applied Mathematics*, 92(1):15–35, 1998.

[21] Richard W Hamming. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160, 1950.

[22] Do Diep, Do Giang, and Phan Phu. Application of Quantum Gauss-Jordan Elimination Code to Quantum Secret Sharing Code. *International Journal of Theoretical Physics*, 57, 2018.

[23] Koji Nagata, Tadao Nakamura, Han Geurdes, J Batle, Ahmed Farouk, Do Diep, and Santanu Patro. Efficient Quantum Algorithms of Finding the Roots of a Polynomial Function. *International Journal of Theoretical Physics*, 57, 2018.

[24] Stefan Heinrich. From Monte Carlo to quantum computation. *Math. Comput. Simul.*, 62:219–230, 2003.

[25] Thomas Wong. *Introduction to Classical and Quantum Computing*. Rooted Grove, Omaha, Nebraska, 2022.

[26] Mihir K. Bhaskar, Stuart Hadfield, Anargyros Papageorgiou, and Iasonas Petras. Quantum algorithms and circuits for scientific computing. *Quantum Information and Computation*, 16(3&4):197–236, 3 2016.

[27] Stefan Heinrich. Numerical Analysis on a Quantum Computer. In Ivan Lirkov, Svetozar Margenov, and Jerzy Waśniewski, editors, *Large-Scale Scientific Computing*, pages 28–39, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[28] Chi Zhang. *Quantum Algorithms and Complexity for Numerical Problems*. PhD thesis, Columbia University, 2011.

[29] Ngoc Do, Do Diep, Do Giang, and Nguyen Minh. Quantum Gauss-Jordan Elimination and Simulation of Accounting Principles on Quantum Computers. *ViAsM16.01*, 56, 2017.

[30] Hefeng Wang and Hua Xiang. A quantum eigensolver for symmetric tridiagonal matrices. *Quantum Information Processing*, 18(3), 2019.

[31] Eleanor Rieffel and Wolfgang Polak. An introduction to quantum computing for non-physicists. *ACM Computing Surveys (CSUR)*, 32(3):300–335, 9 2000.

[32] Giacomo Nannicini. An introduction to quantum computing, without the physics. *SIAM Review*, 62(4), 2020.

[33] Siddhartha Srivastava and Veera Sundararaghavan. Box algorithm for the solution of differential equations on a quantum annealer. *Physical Review A*, 99(5):52355, 5 2019.

[34] Lin Lin. Lecture Notes on Quantum Algorithms for Scientific Computation. 1 2022.

[35] Matthias Möller and Cornelis Vuik. On the Impact of Quantum Computing Technology on Future Developments in High-Performance Scientific Computing. *Ethics and Inf. Technol.*, 19(4):253–269, 12 2017.

[36] Guillermo González, Rahul Trivedi, and J. Ignacio Cirac. Quantum algorithms for powering stable Hermitian matrices. *Physical Review A*, 103(6), 2021.

[37] Stuart Hadfield. Quantum Algorithms for Scientific Computing and Approximate Optimization. 5 2018.

[38] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning, 9 2017.

[39] Juan José García Ripoll. Quantum-inspired algorithms for multivariate analysis: From interpolation to partial differential equations. *Quantum*, 5, 2021.

[40] Piotr Gawron, Dariusz Kurzyk, and Łukasz Pawela. QuantumInformation.jl—A Julia package for numerical computation in quantum information theory. *PLoS ONE*, 13(12), 2018.

[41] Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H Booth, et al. The variational quantum eigensolver: a review of methods and best practices. *Physics Reports*, 986:1–128, 2022.

[42] Lin Lin and Yu Tong. Optimal polynomial based quantum eigenstate filtering with application to solving quantum linear systems. *Quantum*, 4:361, 2020.

[43] Andrew M Childs, Robin Kothari, and Rolando D Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, 2017.

[44] The variational quantum linear solver. `https://learn.qiskit.org/course/ch-applications/the-variational-quantum-linear-solver`. Accessed: 2023-06-14.

[45] Saša Singer and John Nelder. Nelder-mead algorithm. *Scholarpedia*, 4(7):2928, 2009.

[46] Michael JD Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2):155–162, 1964.

[47] John L Nazareth. Conjugate gradient method. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(3):348–353, 2009.

[48] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[49] Stephen G Nash. A survey of truncated-newton methods. *Journal of computational and applied mathematics*, 124(1-2):45–59, 2000.

[50] Michael JD Powell. *A direct search optimization method that models the objective and constraint functions by linear interpolation*. Springer, 1994.

[51] Paul T Boggs and Jon W Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

[52] Andrew R Conn, Nicholas IM Gould, and Philippe L Toint. *Trust region methods*. SIAM, 2000.