Quantum Information & Computation
ISSN 1533-7146 Volume 25 (2025) pp. 640–667
©Cerebration Science Publishing

Article

On Quantum Solvers for Linear Algebraic Systems

Nikos Papagiannis [1,†] **and Manolis Vavalis** [2,*]

[1] Computer Science Department, Purdue University, West Lafayette, IN 47907, USA
[2] Department of Electrical and Computing Engineering, University of Thessaly, Pedion Areos, Volos 38334, Greece
[†] Work performed while this author was with the Department of Electrical and Computing Engineering at the University of Thessaly.
[*] Corresponding author: mav@uth.gr

Received date: 7 July 2025; Accepted date: 10 September 2025; Published online: 6 November 2025

Abstract: Solving large-scale linear systems is an integral part of many scientific disciplines. 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 paper, we present and analyze some of the most important related results. In particular, we present the HHL and WZP algorithms based on the eigen-decomposition of a matrix, 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; HHL; WZP; row and column iteration methods; random walks; VQLS; Qiskit

1. Introduction

There exist no quantum computing computers that formally comply with the principles of quantum mechanics. Nevertheless, many vendors have already delivered intermediate-scale quantum computers that follow different development approaches which are rather "noisy". Currently, the largest quantum computer consists of 433 qubits, and it has been recently announced the first such computer with more than 1000 qubits while other companies will probably surpass this mark very soon.

Regardless of the fact that quantum systems with millions of qubits are not expected to be developed in the short run, the unprecedented potential that such systems offer 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, these studies have shown that quantum computers can perform calculations much faster than classical computers by exploiting certain principles and properties of quantum mechanics. Such supremacy is for example shown in [1] for computations in cryptography.

Quantum commercialization reports originating from various countries around the globe seem to be rather convincing and present impressive spending intentions and revenue predictions [2].

The benefit from the computing throughput provided by quantum computers has the potential to change many industries. Several quantum applications have recently been proposed and applied to use cases with great success. Demonstrations have been constructed using various types of qubits. However, many important problems might very well not have the inherent properties required for benefiting from quantum computing. Is this the case for scientific computing in general and numerical linear algebra in particular?

For more than half a century numerical linear algebra has been motivating early explorers, as John von Neumann [3] and Alan Turing [4], computer architects and developers, and generally the whole Computer industry [5]. It still does so, as new computer architectures such as FPGAs, GPUs, and TPUs (Tensor Processor Units) emerge.

DOI: 10.2478/qic-2025-0035

It is worth recalling that numerical linear algebra and in particular, the LINPACK Benchmark has been, since 1993, the basis to assemble and maintain a list of the 500 most powerful computer systems around the globe [6].

Linear algebra mainly deals with systems of linear algebraic equations of the form $A\vec{x} = \vec{b}$ where $A \in \mathbb{R}^{m\times n}$, $\vec{x} \in \mathbb{R}^n$ and $\vec{b} \in \mathbb{R}^m$, where x_i are the unknown variables that need to be calculated, and a_{ij}'s and b_i's are given. To avoid complexity in the presentation and without any loss of generality for the rest of this paper we assume that $m = n$.

Linear algebra plays a significant role in almost all fields of science and engineering and linear systems of algebraic equations either constitute the core of many applications or are cornerstones of larger algorithms. In either case, they often include enormous amounts of data as n increases arbitrarily.

There is a variety of classical methods for the solution of Linear Systems Problem (LSP) which may be categorized as follows:

Direct Methods like the Gaussian elimination, which is suitable for dense matrices and has a time complexity of $\mathrm{O}(n^3)$.

Iterative Methods like the conjugate gradient descent which is used for sparse matrices and has a time complexity of $\mathrm{O}(n\sqrt{\kappa})$, where κ relies on the condition number of A having a logarithmic, polynomial or even exponential dependence on n.

In general, all these classical linear solvers are of polynomial complexity. Further improving their performance in a classical computer seems almost impossible. This alone justifies the turn of the researchers' attention to algorithms that can be implemented in quantum computers and can solve the LSP fast and efficiently.

The objective of our study is two-fold.

1. Survey the peer-reviewed literature focusing on articles that examine whether quantum computing is suitable for numerical scientific computing in general and numerical linear algebra in particular.
2. Present our efforts that lead to improvements of already proposed methods.

This review allows us to claim that the intersection between quantum computing and high-performance scientific computing has not yet received the attention it deserves. To the best of our knowledge, our paper is one of the few attempts to address this particular issue. It offers an updated exposition of the subject through a much-missing literature review.

Our motivation is further supported by the necessity of linear solvers, which exhibit efficiency that can only be achieved by quantum algorithms. Such necessity recently led to the exploitation of quantum solvers for several specific use-cases. Next, we briefly mention a few.

Linear systems describing geologic fracture networks are too vast to be fully solved, even with the most advanced classical techniques, and simplifying them through methods like up-scaling yields only rough approximations that might miss key network properties [7]. For example, omitting smaller fractures could disrupt percolation, eliminating full connectivity within a fracture region.

The computational core in many areas of computational geophysics, in general, and hydrological modeling, in particular, is a linear algebraic system whose size and characteristics prohibit the use of direct solvers. The plethora of existing commercial computational systems rely on iterative methods that apparently require preconditioning. Such an inverse Laplacian preconditioner is introduced in [8] that reduces the system's condition number scaling from $O(N)$ to $O(\sqrt{N})$ and supports quantum implementation. This is a key step toward a quantum linear system solver that leads a new real-world application in hydrology.

For more related case studies that appeared very recently in the peer, we refer to Section VIII, entitled "Applications of quantum linear systems solvers", in [9] and the references within as well as the article entitled at https://quantumzeitgeist.com/quantum-algorithms-for-solving-linear-systems-why-they-matter/ which accurately represent the overall activities presented in the peer-review articles as well as in grey literature reports.

Our review does not focus on quantum mechanics and assumes readers with introductory quantum computing knowledge (e.g., basic gates and states) and some linear algebra background, supplemented by the paper's preliminaries. We mainly consider this subject from a theoretical standpoint; however, we also proceed to the analysis of a few practical experiments using Qiskit, an open-source SDK for working with quantum computers.

The rest of this paper is organized as follows. In Section 2 we present the necessary theoretical background of quantum mechanics and quantum computing. We mainly focus on the fundamental concepts which are related to the motivation of our study. Section 3 provides the state of the art in the area of quantum numerical linear algebra. In Section 4 we discuss our experimental results on a hybrid quantum-classical linear solver and propose a quantum

circuit that enhances the performance of the algorithm. The synopsis and our preliminary concluding remarks can be found in Section 5.

2. Background

This paper assumes that the reader is aware of the basics of quantum computing. These include, among others, the knowledge of the qubit as the unit of quantum information, quantum states and bra-ket notation, quantum gates and unitary operators, Hermitian matrices, as well as the concepts of quantum systems of n qubits and quantum parallelism. We will take just the meaning of these preliminary terms for granted, however, absolute beginners should not be discouraged, as [10], and a plethora of other sources, provide a thorough and comprehensible introduction to the subject of quantum computing. For example, we assume familiarity with basic quantum state preparation (as in [11], Ch. 5); for a quick recap, see the related material at https://www.youtube.com/channel/UCs9jqTGwJgVpBe50KxLozkw.

Specifically, we first introduce approximation methods of inner products of quantum states, which are necessary for quantum linear algebra as anticipated. In this regard, Sections 2.1 and 2.2 provide insight into the swap test and the Hadamard test respectively. Then in Section 2.3 we explain the concept of density operators which are essential for the description of the quantum solution of an LSP. We conclude with the presentation of the Quantum Fourier Transform in Section 2.4 and the consequent quantum phase estimation in Section 2.5.

2.1. Inner Products and the Swap Test

In quantum computing inner products $\langle x|y\rangle$ express how similar arbitrary quantum states are - the inner product of identical states is 1, while the inner product of orthogonal states is 0. Despite the fact that inner products emerge in numerous quantum formulas and algorithms, in practice it is only possible to approximate them.

For their estimation, a probabilistic implementation called swap test is used. This is based on the 3-register quantum circuit of Figure 1 which includes two input registers for the quantum states and an output register.

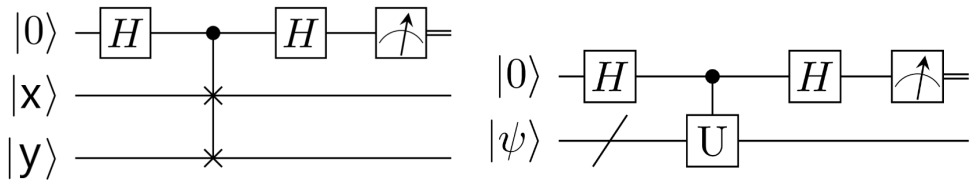


Figure 1. The Swap (**left**) and the Hadamard (**right**) tests.

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 as follows

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

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

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 for the additive error of this estimation to be less than ϵ, $O(\frac{1}{\epsilon^2})$ such iterations are required.

2.2. The Hadamard Test

Another quantum circuit for 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 an arbitrary unitary operator and $|\psi\rangle$ is an arbitrary quantum state. The Hadamard test is a 2-register quantum circuit, illustrated in Figure 1.
The input quantum state of the circuit evolves as follows:

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

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

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 $\frac{1}{2}(1 - \mathfrak{Re}(\langle\psi|U|\psi\rangle))$. Therefore, collecting multiple such outputs is sufficient to approximate $\mathfrak{Re}(\langle\psi|U|\psi\rangle)$.

2.3. Pure and Mixed States, Density Operators

Quantum states are divided into two categories:

Pure states: They can be sufficiently represented with a vector $|x\rangle$ in bra-ket notation and can be defined without uncertainty even when they are complex superpositions of the basis states.

Mixed states: They 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 that are designed to express both pure and mixed states:

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

$$\rho = |x\rangle \langle x| . \tag{1}$$

- 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:

$$\rho = \sum_j p_j |x_j\rangle \langle x_j| . \tag{2}$$

Density operators are Hermitian positive semi-definite matrices with unit trace, so $\mathrm{Tr}(\rho) = \sum_j \rho_{jj} = 1$. The diagonal elements ρ_{jj} in a given orthonormal basis $\{|j\rangle\}_{j=0}^{2^n-1}$ represent the probabilities of obtaining the outcome $|j\rangle$ when measuring in that basis.

2.4. Quantum Fourier Transform

The counterpart of the discrete Fourier transform (DFT) in quantum computing is the Quantum Fourier Transform (QFT), whose input sequence includes a total number of samples (quantum states) equal to a power of 2. The QFT formula is given by:

$$|x\rangle \longrightarrow \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{\frac{2\pi i x y}{2^n}} |y\rangle . \tag{3}$$

An important property of the resulting state in (3) is that it can be expressed as a tensor product of single-qubit states. To see this, we write the integers x and y in binary form:

$$x = \sum_{j=0}^{n-1} 2^j x_j, \qquad y = \sum_{j=0}^{n-1} 2^j y_j,$$

where each $x_j, y_j \in \{0, 1\}$ and x_0, y_0 denote the least significant bits. We also use binary fraction notation:

$$0.x_1 x_2 \cdots x_n = \frac{x_1}{2} + \frac{x_2}{2^2} + \cdots + \frac{x_n}{2^n},$$

where x_1 denotes the first digit after the binary point (contributing $1/2$), x_2 the second digit (contributing $1/2^2$), and so on. With this notation, the exponential factor can be decomposed as

$$\begin{aligned} e^{\frac{2\pi i x y}{2^n}} |y_{n-1} \cdots y_0\rangle \;=\; & \left(e^{2\pi i (0.x_{n-1})}\right)^{y_{n-1}} |y_{n-1}\rangle \otimes \\ & \left(e^{2\pi i (0.x_{n-2} x_{n-1})}\right)^{y_{n-2}} |y_{n-2}\rangle \otimes \cdots \otimes \\ & \left(e^{2\pi i (0.x_0 x_1 \cdots x_{n-1})}\right)^{y_0} |y_0\rangle . \end{aligned} \tag{4}$$

This shows that the state in (3) factorizes as

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

$$\frac{1}{\sqrt{2^n}} \left(|0\rangle + e^{2\pi i (0.x_{n-1})} |1\rangle \right) \left(|0\rangle + e^{2\pi i (0.x_{n-2} x_{n-1})} |1\rangle \right) \cdots \left(|0\rangle + e^{2\pi i (0.x_0 x_1 \cdots x_{n-1})} |1\rangle \right). \tag{5}$$

Figure 2 illustrates how (5) can be implemented in a quantum system that consists of Hadamard gates and controlled phase shift gates:

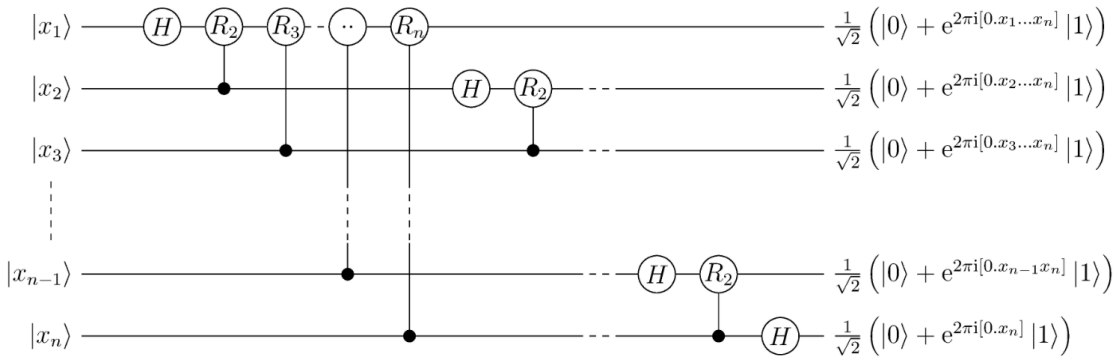


Figure 2. Implementation of the Quantum Fourier Transform. (Reproduced from https://commons.wikimedia.org/wiki/File:Q_fourier_nqubits.pngviaWikimediaCommons).

The R_k gates used in the 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 by reversing the circuit. This inverse is of vital importance, since it can take an input state of the form (3) and recover the corresponding computational basis state $|x\rangle = |x_{n-1} x_{n-2} \cdots x_0\rangle$. Its significance will become evident in Section 2.5.

2.5. Phase Estimation

The quantum phase estimation problem [12] refers to 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. The problem aims to calculate the value of ϕ – and by extension the eigenvalue of U. It assumes the availability of gates that perform controlled U^{2^k} operations ($k \in \mathbb{N}$) although the form of U is undetermined. Its solution is based on the circuit depicted in Figure 3.

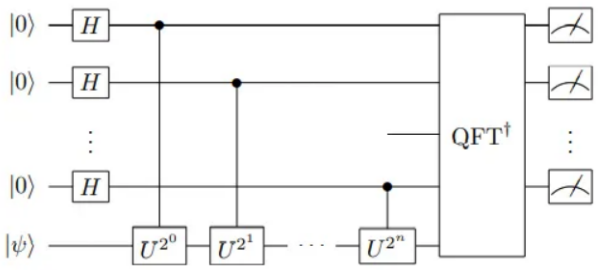


Figure 3. Phase Estimation Algorithm.

Based on properties of eigenvectors and eigenvalues we get that $U^{2^k}|\psi\rangle = (e^{2\pi i\phi})^{2^k}|\psi\rangle$. 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:

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

$$\xrightarrow{U^{2^k}} \frac{1}{\sqrt{2}}(|0\rangle|\psi\rangle + |1\rangle U^{2^k}|\psi\rangle) = \frac{1}{\sqrt{2}}(|0\rangle|\psi\rangle + |1\rangle e^{2\pi i 2^k\phi}|\psi\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 2^k\phi}|1\rangle)\otimes|\psi\rangle\,. \quad (6)$$

Using (6) for all the qubits of the circuit, the subsequent state of the quantum system before the inverse QFT is applied is

$$(|0\rangle + e^{2\pi i 2^{n-1}\phi}|1\rangle)(|0\rangle + e^{2\pi i 2^{n-2}\phi}|1\rangle)\cdots(|0\rangle + e^{2\pi i\phi}|1\rangle) = \sum_{y=0}^{2^n-1} e^{2\pi i\phi y}|y\rangle\,. \quad (7)$$

As the quantum state of (7) resembles the one of (3), 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 $\geq \frac{4}{\pi^2}$.

3. Literature Review

Next, we review selected recent papers that present quantum linear system solvers. In particular, we describe in Section 3.1 the quantum mechanical version of the linear system problem. Then we discuss the main eigenvalue quantum methods for linear systems. Specifically, in Section 3.2 we focus on the most well-known quantum solver, the HHL algorithm, and in Section 3.3 we analyze the WZP algorithm which is similar to the HHL algorithm and can be efficiently applied both to dense and sparse matrices. In Section 3.4 we discuss the iterative solution algorithms and in particular we describe the row and column iterative methods. In Section 3.5 we 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 Section 3.6 we present another hybrid algorithm that is based on random walks. We summarize and briefly compare the above algorithms in Section 3.7.

3.1. Quantum Linear System Problem

A Linear Systems Problem (LSP) takes as input a matrix $A \in \mathbf{R^{m\times n}}$ and a vector $\vec{b} \in \mathbf{R^m}$ and aims to find a vector $\vec{x} \in \mathbf{R^n}$ such that $A\vec{x} = \vec{b}$. The quantum mechanical version of LSP, commonly referred as Quantum Linear Systems Problem (QLSP), has the form

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

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

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

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

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

For a quantum solution to be successful, given a threshold $\epsilon > 0$, the produced quantum density operator ρ_x needs to satisfy the following inequality for the trace distance $D_{\rho_x,x}$

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

3.2. The HHL Solution Method

The most prominent quantum solution method for linear systems of equations was proposed by Harrow, Hassidim, and Lloyd [13]. Their approach, which is commonly referred to as HHL, involves $n \times n$ linear systems of the form $A\vec{x} = \vec{b}$ where the matrix A is Hermitian and the vectors $\vec{x}, \vec{b}$ are normalized so that they can be represented as quantum states. The algorithm can be extended for non-Hermitian matrices, by transforming the system into the form $A'\vec{x'} = \vec{b'}$ where

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

The new matrix A' is Hermitian, since $A'^\dagger = A'$. The preference for Hermitian matrices is due to their capability to be decomposed as follows:

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

where λ_i are their eigenvalues and $|u_i\rangle$ their corresponding eigenvectors. This simplifies the calculation of their inverse matrix

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

Expressing the vector $|b\rangle \propto \vec{b}$ as a linear combination of the eigenvectors of A

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

it is possible to find the solution of the linear system using the formula

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

3.2.1. Description of the Algorithm

The output of the HHL algorithm is a solution quantum state of the form $|x\rangle = \sum_{i=0}^{n-1} x_i |i\rangle$ whose elements are proportional to the ones of the actual solution vector. Not every element of this vector is extracted when the knowledge of the whole solution is not essential because this would need at least n iterations of the implementation. An alternative proposed is to estimate $\vec{x}^\dagger M \vec{x}$ where M is an operator. This requires only one iteration and can reveal a variety of features and properties about $\vec{x}$. The qubit requirements of the HHL algorithm 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 preparation of the vector $|b\rangle$ is based on the work of Grover et al. [14] and is taken for granted for the purposes of this algorithm. Using the eigenvectors $|u_i\rangle$ as the basis of the state space, $|b\rangle$ can be represented as follows:

$$|b\rangle = \sum_{i=1}^{n} \beta_i |u_i\rangle \tag{8}$$

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

1. **Phase estimation:** The working space qubits constitute the control qubits while the conditional Hamiltonian evolution unitary operator [15] is used:

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

   A full analytical derivation of this operator can be found in [15]. 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.
   So, phase estimation yields the eigenvalue-encoded state

   $$\sum_{i=1}^{n} \beta_i |u_i\rangle |\lambda_i\rangle, \tag{9}$$

   as seen in Section 2.5 with a precision of m bits.
2. **Controlled rotation:** This process is applied on the ancilla qubit. As A is Hermitian, λ_i^{-1} are the eigenvalues of the inverse of A which need to be incorporated in the state (9). 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

   $$\sum_{i=1}^{n} \beta_i |u_i\rangle |\lambda_i\rangle \left(\sqrt{1 - \frac{C^2}{\lambda_i^2}} |0\rangle + \frac{C}{\lambda_i} |1\rangle \right), C \in \mathbf{R}. \tag{10}$$

3. **Inverse phase estimation:** This is the reverse of the first step, so the qubits of the working space are reset to $|0\rangle$ and the resulting state is:

   $$\sum_{i=1}^{n} \beta_i |u_i\rangle \left(\sqrt{1 - \frac{C^2}{\lambda_i^2}} |0\rangle + \frac{C}{\lambda_i} |1\rangle \right). \tag{11}$$

   The algorithm terminates with a post-selection 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 that summarizes the steps of the HHL algorithm is given in Figure 4:

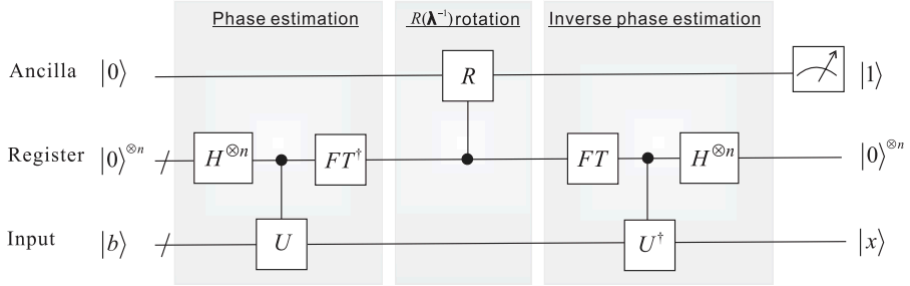


Figure 4. Schematic representation of the HHL algorithm ([16]).

3.2.2. Summarizing the HHL Algorithm

Contrary to the optimal classical linear solvers which solve a linear system with a time complexity that scales polynomially with the size of the system, the corresponding complexity of the proposed quantum HHL algorithm is $\mathrm{O}(s^2\kappa^2 \log n/\epsilon)$ in the case A is s-sparse. This translates into an exponential speed-up. The density of the matrix is a significant, but not the sole factor that determines the performance of the algorithm. The condition number κ also needs to be taken into consideration, as it sometimes scales polynomially or even exponentially with n. In this case, it is doubtful that HHL can be faster than its classical counterparts and the scaling of κ must be controlled. This could possibly be achieved in two ways according to [13]:

- By processing only the well-conditioned part of A:

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

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

The dependence of the complexity of HHL on the precision parameter ϵ is also noteworthy. This is mainly related to the use of the phase estimation routine which can significantly reduce the performance of the algorithm if high accuracy of the solution is required. Finally, possible sources of failure need to be considered. Phase estimation is one of them because it can produce wrong results under certain circumstances (Section 2.5). In addition, post-selection is not guaranteed to succeed. According to [13], if the constant C is $\mathrm{O}(1/\kappa)$, $\mathrm{O}(\kappa)$ repetitions of the routine including the post-selection step seem to be enough to produce the desired result.

The HHL algorithm is a breakthrough in the field of quantum linear solvers although it has certain limitations. This work is the cornerstone of the field of quantum linear solvers and this is illustrated by the fact that it inspired several solvers developed recently.

3.3. WZP Method for Dense Matrices

Another notable quantum solver for linear sets of equations is that proposed by Wossnig, Zhao, and Prakash [17] which is commonly referred to as WZP. This algorithm bears some resemblance to the HHL algorithm, but it is appropriate for both dense and sparse matrices. Similar to HHL, the matrix A needs to be Hermitian, but non-Hermitian matrices may also be treated as described for HHL above.

3.3.1. Quantum Singular Value Estimation

The WZP algorithm relies on quantum singular value estimation (QSVE) which has been proposed in [18] for the implementation of a quantum recommendation system. QSVE is a factorization method that estimates the eigenvalues of the matrix A. 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. Recall that any matrix $A \in \mathbf{R}^{n\times n}$ can be decomposed using the singular value decomposition (SVD) as

$$A = U\Sigma V^\dagger,$$

where U and V are unitary matrices whose columns are the left and right singular vectors of A, and Σ is a diagonal matrix whose nonnegative entries σ_i are called the singular values of A. In particular, for each singular value σ_i we have

$$Av_i = \sigma_i u_i, \qquad A^\dagger u_i = \sigma_i v_i,$$

where v_i is a right singular vector and u_i is the corresponding left singular vector. We will use this notation throughout the description of QSVE. 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.
- 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.

P and Q are orthogonal, as $P^\dagger P = I_n$ and $Q^\dagger Q = I_n$ and the following equalities hold

$$|Pe_i\rangle = |i, A_i\rangle = \frac{1}{\|A_i\|}\sum_{j=1}^{n} A_{ij}\,|i,j\rangle\,, 1 \le i \le n, \tag{12}$$

$$|Qe_j\rangle = |A_F, j\rangle = \frac{1}{\|A\|_F}\sum_{i=1}^{n}\|A_i\|\,|i,j\rangle\,, 1 \le j \le n. \tag{13}$$

For the implementation of these quantum states, a data structure is used that performs the mappings:

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

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

The complete presentation of this data structure is available in the cited literature [18], while we include here the essential elements relevant to our discussion. The combination of (12) and (13) proves that the matrices P and Q can be used to factorize a normalized version of the matrix A:

$$(P^\dagger Q)_{i,j} = \langle i, A_i | A_F, j\rangle = \frac{A_{ij}}{\|A_i\|}\frac{\|A_i\|}{\|A\|_F} = \frac{A_{ij}}{\|A\|_F} \Rightarrow P^\dagger Q = \frac{A}{\|A\|_F}. \tag{14}$$

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

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

These are the reflections in the column space of P and Q respectively. The matrix

$$W = UV$$

is the basis of the QSVE algorithm. 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 [19]. A preliminary element of its proof is the fact that an orthogonal projection that maps a vector Qv of the column space of Q to the column space of P is based on the orthogonal projector $PP^\dagger$ while mapping a vector Pw to the column space of Q is based on the orthogonal projector $QQ^\dagger$:

$$(PP^\dagger)Qv = \frac{1}{\|A\|_F}PAv, \quad (QQ^\dagger)Pw = \frac{1}{\|A\|_F}QA^\dagger w.$$

Given a singular value σ_i of A with right singular unit vector v_i and left singular unit vector u_i, it follows that:

$$Av_i = \sigma_i u_i \Rightarrow PP^\dagger Qv_i = \frac{\sigma_i}{\|A\|_F}Pu_i, \tag{15}$$

$$A^\dagger u_i = \sigma_i v_i \Rightarrow QQ^\dagger Pu_i = \frac{\sigma_i}{\|A\|_F}Qv_i. \tag{16}$$

It can be observed that (15) 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, (16) does the inverse. Furthermore, it can be proved that:

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

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

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

Thus, an alternative expression for (15) and (16) is the following:

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

The subspace with basis $\{Pu_i, Qv_i\}$ can be 2-dimensional on the non-trivial case. This subspace is invariant under the action of the matrix W, as 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 $\theta_i/2$ between the axes. Therefore e^{θ_i} is an eigenvalue of W and the following equality holds:

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

The previous analysis is essential for the comprehension of the steps of QSVE. These include:

1. The preparation of 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 $A \in \mathbf{R}^{n\times n}$:

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

2. The transformation of the previous quantum state using a Q operator:

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

   Then $\lceil log_2 n \rceil$ qubits initialized in $|0\rangle$ are appended to the resulting state. These qubits will be required for the next step.

3. Phase estimation extracts $\bar{\theta_i}$ as an estimation of the eigenvalues of W, building on the $|Qx\rangle$ state. The resulting state of this step is

$$\sum_i a_i |Qv_i, \bar{\theta_i}\rangle .$$

4. The transformation of the previous state in order to approximate the singular values σ_i of A, taking into account $\bar{\theta_i}$. This is achieved using the formula

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

5. The reversion of the effect of the operator Q which was used in step 2 so that the final state is reached

$$\sum_i a_i |v_i\rangle |\bar{\sigma_i}\rangle .$$

3.3.2. Description of the WZP Algorithm

The above process 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 through the following complete algorithm of the WZP method which assumes that A has been scaled so that $\lambda_i \in [-1, -1/\kappa] \cup [1/\kappa, 1]$, where κ is the condition number of A

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

$$|b\rangle = \sum_i \beta_i |v_i\rangle .$$

2. 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 $\mu+\lambda_i$ giving us that

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

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. 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 $\lambda_i < 0$, then

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

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, the value of register B now is larger than the value of register C.

3. Introduce 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

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

4. Add another ancilla register and apply a controlled rotation by $\gamma = \mathrm{O}(1/\kappa)$ conditioned on the register B. Afterward, the registers B, C, D are no longer required, so we reset them 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 post-select on the register that was introduced 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 iterations should be required.

3.3.3. Summarizing the WZP Method

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. The complexity of the algorithm is $\mathrm{O}(\kappa^2 \log n \|A\|_F/\epsilon)$. According to [17], in a few 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 our paper.

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 tree-like data structure which seems easier to construct. Therefore, it would be justified to prefer WZP over HHL under certain conditions.

3.4. Row and Column Iteration Methods

3.4.1. Theoretical Presentation of the Algorithms

The use of iterative methods as quantum solvers for linear sets of equations has been actively investigated in recent years [20–22], as the potential of quantum parallelism can lead to a speed-up in comparison to the corresponding classical algorithms. 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 or the matrix is sparse. 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 the system $A\vec{x} = \vec{b}$.

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 algorithms often 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 is proposed in [20] and is based on the row and column iteration methods which are used in a lot of large-scale data science applications. The main advantage of these methods is that for every iteration, not the whole matrix A is needed. The two methods are defined as follows:

- **Row (or Kaczmarz) iteration method:** It 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 as follows

$$x_{k+1} = x_k - \left(\frac{a_{i_k}^T x_k}{\|a_{i_k}\|} - \frac{b_{i_k}}{\|a_{i_k}\|} \right) \frac{a_{i_k}}{\|a_{i_k}\|}. \tag{17}$$

  Geometrically a step of the row iteration method is equivalent to the orthogonal projection of the vector x_k on the plane $a_{i_k}^T x = b_{i_k}$.

- **Column (or coordinate descent) iteration method:** It 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 as follows

$$x_{k+1} = x_k + \frac{c_{j_k}^T (b - A x_k)}{\|c_{j_k}\|^2} e_{j_k}. \tag{18}$$

  If we denote the residual of the k-th iteration as $r_k = b - A x_k$, the expression of (18) can be written as follows:

$$x_{k+1} = x_k + \frac{c_{j_k}^T r_k}{\|c_{j_k}\|^2} e_{j_k}, \tag{19}$$

  where the residual also follows an iterative scheme:

$$r_{k+1} = b - A x_{k+1} = b - A x_k - A \frac{c_{j_k}^T r_k}{\|c_{j_k}\|^2} e_{j_k} = r_k - \frac{c_{j_k} c_{j_k}^T}{\|c_{j_k}\|^2} r_k. \tag{20}$$

3.4.2. Quantum Row Iteration Method

The quantum algorithm of Shao assumes the existence of a QRAM [23] which is responsible for preparing a row a_t in the form of a quantum state and is equivalent to a unitary operator V_t such that:

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

All the rows of A are supposed to be scaled to $\|a_t\| = 1$ for the purpose of their representation as quantum states. The solution unit vector approximations $|x_k\rangle$ are assumed to be included in quantum states $|X_k\rangle$ such that

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

Here and in what follows, we use the notation $|\cdots\rangle$ as a placeholder for an unspecified quantum state, whose explicit form is not relevant to the argument. The basic idea of the algorithm 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

$$|0\rangle\left(\beta\sqrt{p}\,|0\rangle\,|x_k\rangle+\gamma\,|1\rangle\,|a_t\rangle\right)+\beta\sqrt{1-p}\,|1\rangle\,|0\rangle\,|\cdots\rangle\,. \tag{21}$$

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

$$U_t=\begin{bmatrix} I_n-|a_t\rangle\langle a_t| & |a_t\rangle\langle a_t| \\ |a_t\rangle\langle a_t| & I_n-|a_t\rangle\langle a_t| \end{bmatrix}=I_2\otimes(I_n-|a_t\rangle\langle a_t|)+X\otimes(|a_t\rangle\langle a_t|). \tag{22}$$

Through equality $V_t\,|0\rangle=|a_t\rangle\Rightarrow|a_t\rangle\langle a_t|=V_t\,|0\rangle\langle 0|\,V_t^\dagger$ (22) becomes

$$U_t=(I_2\otimes V_t)(I_2\otimes(I_n-|0\rangle\langle 0|)+X\otimes|0\rangle\langle 0|)\Big(I_2\otimes V_t^\dagger\Big). \tag{23}$$

Using (22) and applying the operator U_t on the first part of the quantum state of (21), the quantum state which is calculated below is produced

$$\begin{aligned} &U_t(\beta\sqrt{p}\,|0\rangle\,|x_k\rangle+\gamma\,|1\rangle\,|a_t\rangle)= \\ &=|0\rangle\otimes(\beta\sqrt{p}(I_n-|a_t\rangle\langle a_t|)\,|x_k\rangle+\gamma\,|a_t\rangle\langle a_t|\,|a_t\rangle)+ \\ &+|1\rangle\otimes(\beta\sqrt{p}\,|a_t\rangle\langle a_t|\,|x_k\rangle+\gamma(I_n-\langle a_t|\,|a_t\rangle)\,|a_t\rangle)= \\ &=|0\rangle\otimes(\beta\sqrt{p}(I_n-|a_t\rangle\langle a_t|)\,|x_k\rangle+\gamma\,|a_t\rangle)+|1\rangle\otimes(\beta\sqrt{p}\,\langle a_t|x_k\rangle\,|a_t\rangle) \end{aligned} \tag{24}$$

From (17) it can be observed that in terms of quantum states we have

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

and this expression resembles the first part of (24) if the values of β,γ are chosen appropriately. For example, $\beta=\|x_k\|$ and $\gamma=b_t\sqrt{p}$.

Following this intuitive explanation of the concept of the quantum row iteration algorithm, the analytical sequence of its steps is presented next.

1. An initial approximation $|x_0\rangle$ of the unit vector of the solution is selected. We set $k=0$ and a variable $v_k=1$ to express the state of the quantum system as follows:

$$|X_k\rangle=\frac{\|x_k\|}{v_k}\,|0\rangle^{\otimes k}\,|x_k\rangle+|0^\perp\rangle^{\otimes k}\,|\cdots\rangle\,. \tag{25}$$

   Here $|0^\perp\rangle^{\otimes k}$ denotes a tensor product of states orthogonal to $|0\rangle$. The explicit form of these states is not needed, as they serve only to complete the expression.

2. The row with index $t_k\in\{1,\cdots,n\}$, 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 $|00\rangle\rightarrow\left(\beta_{t_k}\,|0\rangle+\gamma_{t_k}\,|1\rangle\right)|0\rangle$ 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:

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

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

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

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

$$|X_{k+1}\rangle = |0\rangle^{\otimes(k+1)} \otimes \left(\frac{\beta_{t_k}\|x_k\|}{v_k}\left(I_n - |a_{t_k}\rangle\langle a_{t_k}|\right)|x_k\rangle + \gamma_{t_k}|a_{t_k}\rangle \right) + |0^{\perp}\rangle^{\otimes(k+1)}|\cdots\rangle \quad (26)$$

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, Ref. (26) can alternatively be written as follows

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

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

It can be observed that the produced state has the form of (25), thus an iteration of the algorithm is completed at this point. To proceed to the next iteration of the algorithm, we set $k = k+1$ and go to step 2.

3.4.3. Quantum Column Iteration Method

Similarly to the row iterative method, for the column iterative method, the existence of a unitary operator S_j which prepares the column vectors $|c_j\rangle$ is assumed

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

These columns are scaled to $\|c_j\| = 1$ so that they can be represented as quantum states. The algorithm initially selects an approximation x_0, calculates the initial residual $r_0 = b - Ax_0$ and then updates the respective quantum states $|x_k\rangle$ and $|r_k\rangle$ as follows during each iteration

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

$$|r_{k+1}\rangle \propto \|r_k\|(I_n - |c_{t_k}\rangle\langle c_{t_k}|)|r_k\rangle \quad (27)$$

$|x_k\rangle$'s and $|r_k\rangle$'s are included in quantum states $|X_k\rangle$ and $|R_k\rangle$ respectively of the form

$$|R_k\rangle = \|r_k\||0\rangle^{\otimes k}|r_k\rangle + |0^{\perp}\rangle^{\otimes k}|\cdots\rangle, \quad |X_k\rangle = \frac{\|x_k\|}{k+1}|0\rangle^{\otimes 2k}|x_k\rangle + |0^{\perp}\rangle^{\otimes 2k}|\cdots\rangle.$$

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

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

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

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

This resembles the update of $|x_k\rangle$ in the row iteration algorithm if we set $v_k = 1$. For the update of $|x_k\rangle$ it is assumed that the approximations 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 these quantum states are combined in a state of the form:

$$|\phi_1\rangle = \beta|00\rangle|\tilde{X_k}\rangle + \gamma|10\rangle|\tilde{R_k}\rangle \text{ with } \beta^2+\gamma^2 = 1.$$

A W_{t_k} operator is applied, 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 exchanging the second with the third qubit, to produce:

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

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

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

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

The first part of $|\phi_2\rangle$ is similar to (27), but to compute $|x_{k+1}\rangle$ it is essential 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

$$|\phi_3\rangle = (I_4 \otimes G_k \otimes I_n) |\phi_2\rangle = |000\rangle \left(\frac{c\beta\|x_k\|}{v} |x_k\rangle + s\gamma\|r_k\| |t_k\rangle \langle t_k| S_{t_k} |r_k\rangle\right) + |0^\perp\rangle^{\otimes 3} |\cdots\rangle . \quad (28)$$

Combining (27) and (28), the expected quantum state of $|X_{k+1}\rangle$ has the form:

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

Comparing this state with (28) and setting $v = k+1$, it is evident that $\frac{c\beta}{k+1} = s\gamma = \frac{1}{k+2}$. In [20] $c = \beta = \sqrt{\frac{k+1}{k+2}}$ and $s = \gamma = \sqrt{\frac{1}{k+2}}$ is proposed.

At this point, the reader should have a sufficient understanding of how the quantum column iteration algorithm works. Next, we summarize and analyze its main steps.

1. The algorithm begins with an initial approximation $|x_0\rangle$ of the unit vector of the solution. 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 $|R_k\rangle$, $|\psi_0\rangle$ can alternatively be expressed 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 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 $|\psi_3\rangle$ of the form

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

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

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

$$+ |0^{\perp}\rangle^{\otimes(2k+2)} |\cdots\rangle = |X_{k+1}\rangle .$$

4. The state of $|R_k\rangle$ is updated using the same operators that are used for the update of $|X_k\rangle$ in the row iteration algorithm, as explained above. We set $k = k + 1$ and go to step 2 to proceed to the next iteration.

3.4.4. Summarizing the Iteration Algorithms

The row and column iteration algorithms are an innovative idea for a quantum linear solver. Their main advantage compared to the classical iterative methods is 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. 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 $\mathrm{O}\big(\kappa_s^2 (\log n) \log \frac{1}{\epsilon}\big) \approx \mathrm{O}(T \log n)$, where $\kappa_s = \|A\|_F \|A^{-1}\|$ is the scaled condition number and $\|A\|_F$ is the Frobenius norm of A. 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 $\mathrm{O}\big(\kappa_s^2 n \log \frac{1}{\epsilon}\big)$.

On the other hand, the main drawback of these algorithms is 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.

3.5. Variational Quantum Linear Solver

3.5.1. Outline of the Algorithm

All the above described quantum linear solvers exhibit 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. 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) [24], 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 that are needed.

In this subsection, we focus on a short description of the structure of the VQLS algorithm which is summarized in Figure 5:

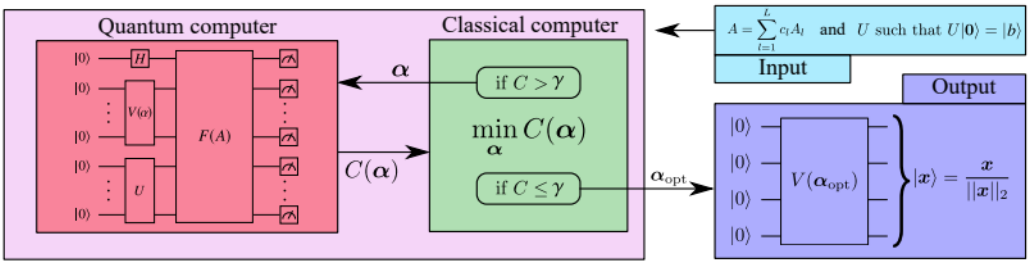


Figure 5. Diagram of the VQLS algorithm ([24]).

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

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

To find $|x\rangle$ that is proportional to the solution $\vec{x}$, an ansatz is used for the combination of gates $V(a)$ which are required 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.

3.5.2. Implementation Details

In this subsection, more details about the implementation of the VQLS algorithm are discussed. First, the form of the cost function needs to be defined, so that it measures the component of $|\Phi\rangle = A|x\rangle$ which is orthogonal to $|b\rangle$. Four different cost functions are proposed in [24]

1. Two global cost functions consisting of the
   - $\hat{C_G} = Tr(|\Phi\rangle\langle\Phi|(I - |b\rangle\langle b|)) = \langle x|H_G|x\rangle$ where $H_G = A^\dagger(I - |b\rangle\langle b|)A$
   - and its normalized version $C_G = \hat{C_G}/\langle\Phi|\Phi\rangle$.
2. Two local cost functions consisting of the function
   - $\hat{C_L} = \langle x|H_G|x\rangle$ where $H_L = A^\dagger U\Big(1 - \frac{1}{n}\sum_{j=1}^{n}|0_j\rangle\langle 0_j| \otimes I_{\bar{j}}\Big)U^\dagger A$ and where $|0_j\rangle$ is the zero state on the j-th qubit, whereas $I_{\bar{j}}$ is the identity matrix which excludes the j-th qubit.
   - and its normalized version $C_L = \hat{C_L}/\langle\Phi|\Phi\rangle$.

All the above cost functions are suitable for the solution of a $n \times n$ system, but 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.

$V(a)$ can be mathematically expressed as

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

where $k_1, k_2, \dots, k_l$ specify the gate types and their positions in the quantum circuit. In cases where $k_1 = k_2 = \cdots = k_l$, only a single gate type is used throughout the circuit, which is overly restrictive. More generally, one can consider a fixed-structure ansatz, where the sequence $(k_1, \dots, k_l)$ is kept constant during training and only the parameters a are optimized, or a variable-structure ansatz, where the sequence itself may change. In addition, different training strategies can be applied: one may optimize all parameters simultaneously or proceed layer by layer, optimizing parameters by circuit layers.

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. Ref. [24] proposes 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.

3.5.3. Summarizing the VQLS Algorithm

The performance of the VQLS algorithm was evaluated in [24] by solving up to 1024×1024 linear systems using Rigetti's Quantum Cloud Services. Thus, the complexity of the algorithm which we present here is based on experimental data and not theoretical formulas. 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, despite the fact that it is implemented in the currently available flawed quantum computers.

3.6. Random Walks Solver

3.6.1. Classical Random Walks

Linear algebraic methods that use random walks can handle linear systems of 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 $A = I - H$. We know that, if $\|H\| < 1$, then

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

so the solution vector can be calculated as follows

$$\vec{x} = A^{-1}\vec{b} = (I - H)^{-1}\vec{b} = \sum_{i=0}^{\infty} H^i\vec{b}. \tag{29}$$

Random walks $\gamma = (i_0, i_1, \cdots, i_k)$ which are required for these methods and terminate after k steps are implemented using Markov chains expressed with a $n \times n$ matrix P such that

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

To construct the matrix P and to define how the corresponding random walk terminates

- Neumann and Ulam [25] 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. [26] proposed that $P_{ij} = \frac{|H_{ij}|}{\sum_j |H_{ij}|}$. 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. Then an approximation of the i_0-th element of the solution vector is calculated as follows

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

3.6.2. Outline of the Algorithm

A hybrid classical-quantum linear solver that utilizes random walks implemented in a quantum computer was proposed in [27] where the reader is referred for a comprehensive explanation of this method. Next, we provide a general description of its structure.

Given a $n \times n$ linear system, $A\vec{x} = \vec{b}$, the matrix A needs to be expressed as:

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

where P is a Markov chain transition matrix that satisfies the relations $P_{ij} \geq 0$ and $\sum_j P_{ij} = 1$. Consequently, (29) is transformed as follows

$$\vec{x} = \sum_{s=0}^{\infty} \gamma^s P^s \vec{b}. \tag{30}$$

To approximate the I_0-th element of x, the following truncated version of (30) is used

$$x_{I_0}^{(c)} = \sum_{s=0}^{c} \gamma^s \sum_{I_1, I_2, \cdots, I_s = 1}^{n} P_{I_0 I_1} \cdots P_{I_{s-1} I_s} b_{I_s}. \tag{31}$$

The evaluation of (31) is based on a quantum random walk on a graph consisting of n nodes. Given a tolerance of ϵ, c should be selected to be approximately $\log\left(\frac{1}{\epsilon}\right) \log\left(\frac{1}{\gamma}\right)$, as it inserts an error of $\mathrm{O}(\gamma^c)$ in the calculation of $\vec{x}$.

Contrary to the previous quantum solvers that were presented in this paper, the form of this algorithm does not require the vectors $\vec{x}$ and $\vec{b}$ to be represented as quantum states. Only the matrix A – or equivalently the matrix P – needs to be appropriately inserted in quantum registers. [27] suggests the Hamming cube structure [28] to encode the n-node graph that corresponds to the transition matrix P. This graph uses $m = \log_2 n$ qubits 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.

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 one 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 operator U is the following general single qubit gate

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

The parameters ϕ and λ are phase angles that represent components of a single qubit rotation. Their ranges are typically defined as $\phi \in [0, 2\pi)$ and $\lambda \in [0, 2\pi)$.

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 $i_{-1} = 0$. 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 paper and the interested reader is referred to [27].

It can be proved that the transition matrix P cannot be written as a tensor product of independent 2×2 matrices, which means that the qubits of the graph are entangled, hence the probability that the one flips depends on the probability that the other one flips.

3.6.3. Summarizing the Random Walks Solver

The proposed random walks solver exploits the probabilistic nature of the quantum states in order to implement a Markov chain structure. It is noteworthy that it does not have extravagant space requirements, as $1 + \log n$ qubits are sufficient. In addition, according to [27] the time complexity for the computation of an element of the solution vector is $\mathrm{O}(\log n)$. The only parameter taken into consideration is the size of the linear system, however, the algorithm still seems to outperform the classical methods, especially when only a part of the solution is required. In addition, contrary to the other quantum solvers, it is important to note that the output of this solver is classical data, so it is more accessible and can easily be extracted.

3.7. Synopsis and Comparison of the Algorithms

As Table 1 summarizes, all the quantum solvers presented above exhibit supremacy compared to the best-performing classical linear solvers.

Table 1. Comparison of the quantum solvers.

Algorithm	Complexity	Ref.	Type	Limitations
HHL	$\mathrm{O}(s^2\kappa^2 \log n/\epsilon)$	[13]	eigen-decomposition	phase estimation, matrix density, condition number, Hamiltonians
WZP	$\mathrm{O}(\kappa^2 \log n \|A\|_F/\epsilon)$	[18]	eigen-decomposition	phase estimation, condition number
Row and column iteration	$\mathrm{O}\big(\kappa_s^2 \log n \log \frac{1}{\epsilon}\big)$	[20]	iterative	ancilla qubits, condition number
VQLS	$\mathrm{O}(\kappa \log n \log \frac{1}{\epsilon})$	[24]	hybrid	ansatz option, condition number
Random walks	$\geq \mathrm{O}(\log n)$ per x_i	[27]	hybrid, Monte Carlo	scalability

All of these algorithms theoretically achieve an exponential speed-up. However, a lot depends on the condition number κ of the matrix A which sometimes scales polynomially with the size of the linear system. If A is ill-conditioned, the quantum algorithms may be slower than the classical methods. This problem can partially be solved using several approaches like preconditioners, but 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 precision parameter ϵ should also be considered. If very accurate solutions are required, then an arbitrary increase in the runtime of the algorithms should be expected.

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 VQLS algorithm, have been tested, so this confirms that these solvers are not limited to theoretical studies that can not be practically implemented in quantum computers. Nevertheless, asymptotic complexity is not the only factor that determines their viability, and it is equally important to examine the trade-offs between theoretical guarantees and hardware feasibility.

Algorithms such as HHL and WZP rely on quantum phase estimation and deep controlled operations; while they enjoy strong asymptotic guarantees, their circuit depth and noise sensitivity make them infeasible on today's NISQ devices. By contrast, hybrid solvers like VQLS are explicitly designed for hardware efficiency, since they use shallow variational circuits and shift part of the computational burden to a classical optimizer. This reduces qubit and gate overhead but introduces other challenges: measurement noise can accumulate due to the repeated sampling required, convergence heavily depends on the choice of ansatz, and optimizer robustness is not guaranteed. Thus, the comparison highlights a fundamental balance: algorithms with phase estimation achieve stronger guarantees in theory but lack near-term feasibility, whereas VQLS is practical and moderately robust under current hardware constraints, provided that careful ansatz design and optimizer selection are employed.

4. Experimental Results

In this Section we propose an implementation of the hybrid quantum-classical VQLS algorithm we discussed in Section 3.5. The requisite mathematical formulas required are the following:

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

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 $|b\rangle$. The cost function 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 (See Section 2.2)

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

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

Given that all involved operators exclusively include real numbers, we note that

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

In all our experiments, the coding for the quantum computation routines relies on the Qiskit framework which allows access to both simulators and cloud-based quantum processors. For the purposes of our study, the Aer simulators has been used. Inspired by the approach described in [29], we solve 8×8 linear systems utilizing the VQLS algorithm. Our main contributions to [29] are the following improvements.

1. We construct and code quantum operator blocks, to reduce the qubit requirements for the inner product calculations.
2. We utilize different optimizers included in the scipy Python library and compare them to discover the one who produces the optimal a and by extension the best ansatz $V(a)$.

3. We propose an alternative ansatz structure to the ones in [29] and [24], tailored to real-valued problem instances. In our simulations this ansatz achieves consistently faster and more stable convergence while using reduced circuit complexity. Although classically simulable, this design illustrates how problem-specific ansatz choices can substantially enhance the practical performance of VQLS.

4.1. Discussion of the Results

Ref. [29] adapts Hadamard test circuits (extending the 2-register design in Section 2.2, Fig. 1) by adding an ancilla for controlled operations, enabling efficient block encoding without full decomposition. Coding the $V(a)$, U, and A_i operators by constructing "block operators" as illustrated in Figure 6, this additional qubit can be omitted. In cases of more complex linear systems, more than one ancilla qubits may be omitted.

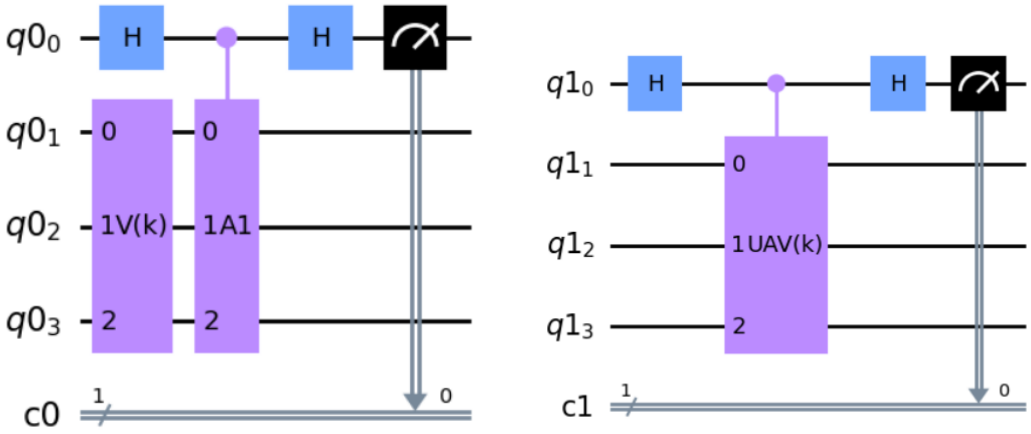


Figure 6. Inner product circuits for $\langle\Phi|\Phi\rangle$ (**left**) and $|\langle b|\Phi\rangle|^2$ (**right**).

The ansatz may take many different forms [24] and an indicative fixed ansatz structure is presented which includes alternating R_y gates with controlled-Z gates, as depicted on the left scheme of Figure 10. The question which arises is which optimizer is the most appropriate for the calculation of the optimal values of a, if the ansatz has the above form. To address this, we 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, while $U = H^{\otimes 3}$. The optimizers which we experiment with stem from the scipy Python library and an interested reader can discover details about their application in [30]. The convergence behavior of these optimizers is shown in Figure 7.

It is evident that the best-performing algorithm is the COBYLA optimizer which performs constrained optimization using linear approximation when the derivative of the target function is unknown. The Powell optimizer produces comparable results, but it requires many more iterations to achieve converges and its behavior is very unstable. Furthermore, the solution vector it calculates fails to sufficiently converge to the actual solution, as the residual plot of Figure 8 indicates.

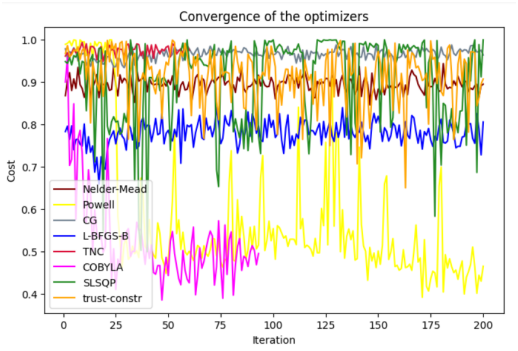


Figure 7. The convergence of the optimizers.

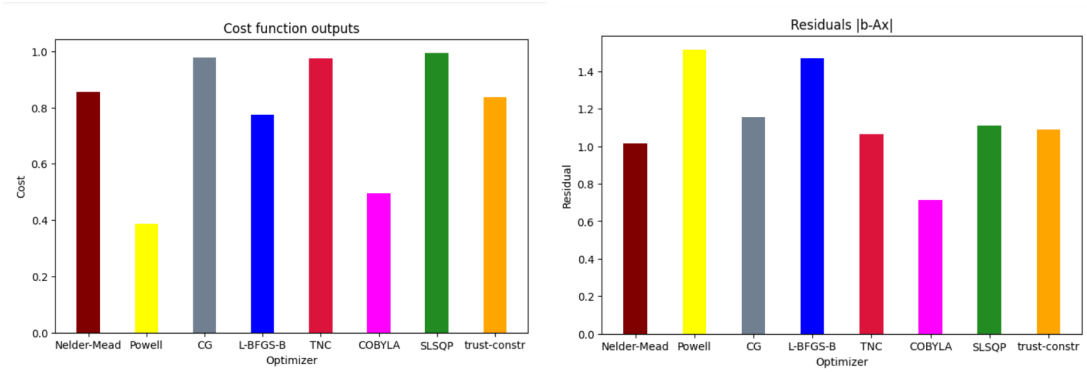


Figure 8. The cost function of the optimizers (**left**) and the corresponding residuals |b-Ax| (**right**).

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 Figure 9—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.

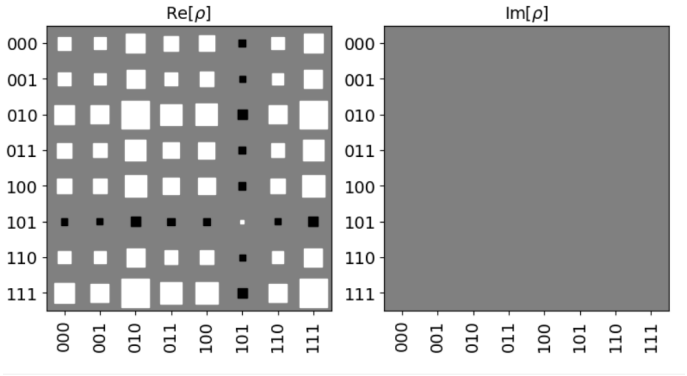
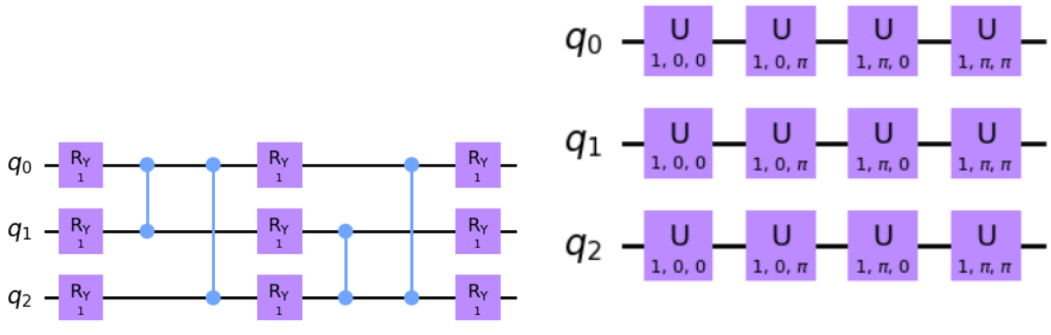


Figure 9. Density matrix of the COBYLA quantum solution vector.

Further improving the performance of the VQLS algorithm largely depends on the structure of the ansatz. Thus, we propose an ansatz depicted in Figure 10 consisting of four gates for each qubit ($U(\theta, 0, 0)$, $U(\theta, 0, \pi)$, $U(\theta, \pi, 0)$ and $U(\theta, \pi, \pi)$), where U is the general single qubit unitary gate. 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. We note that the proposed U-ansatz does not include entangling gates.



(a) (b)

Figure 10. The initial ansatz structure (R_y, CZ gates) on the left and the proposed ansatz structure (U gates) on the right.

The values of θ are the parameters that 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. We solve the aforementioned linear system using this ansatz, and present in Figure 11 the resulting density matrix.

The results indicate a clear improvement in the performance of the algorithm since the cost function converges much more efficiently. The same applies to most of the linear systems which we have tested. Due to computational limitations, error bars were not computed for the figure. However, the results reflect averages over 10 independent runs, and the consistent trends across runs suggest robustness of the observed reductions. For instance, for the linear system $A = 0.3\mathbb{Z}_1 + 0.4\mathbb{Z}_2$, we find comparable results, as shown in Figure 12.

Future extensions could incorporate real-preserving entangling operations to enhance expressivity while maintaining compatibility with real-valued problem instances. 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 that have been examined in practice and in theory.

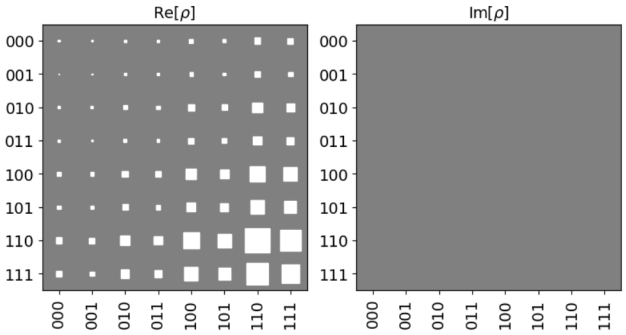


Figure 11. Density matrix of the quantum solution vector produced by the proposed ansatz.

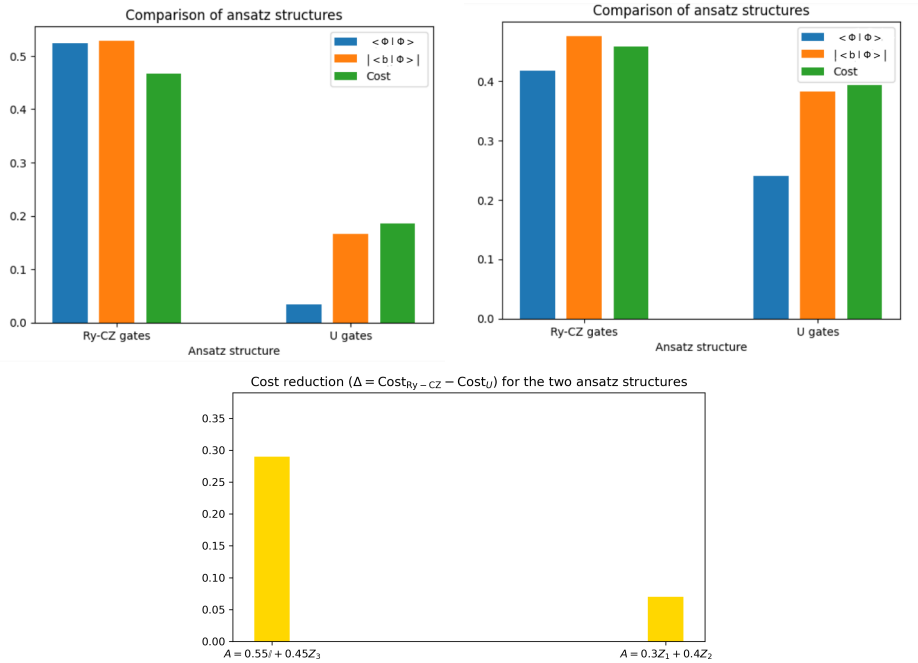


Figure 12. Comparison of the ansatz structures $A = 0.55\mathbb{I} + 0.45Z_3$ (left) and $A = 0.3Z_1 + 0.4Z_2$ (right). The metrics shown are $\langle\Phi|\Phi\rangle$, $|\langle b|\Phi\rangle|$, and the cost $C = 1 - \frac{|\langle b|\Phi\rangle|^2}{\langle\Phi|\Phi\rangle}$. Here $\langle\Phi|\Phi\rangle$ indicates the norm of $A\,|x(a)\rangle$, while $|\langle b|\Phi\rangle|$ measures the raw overlap with $|b\rangle$. Since these two quantities are not meaningful in isolation, the cost is the definitive convergence criterion. The additional panel reports the cost reduction $\Delta = \text{Cost}_{\text{Ry-CZ}} - \text{Cost}_U$, which is the quantity used to evaluate the improvement achieved by the U-gates ansatz. The trends are averaged over 10 runs. While statistical uncertainties are not quantified due to resource limitations, the consistent reductions suggest improved convergence for the U-gates ansatz.

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 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, we attempt 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. It remains to be seen if these methods can be effectively extended for large-scale systems as our current experimentation indicates. The convergence trends in Figure 12 are based on averaged data without statistical error quantification due to resource constraints. Future work with larger-scale simulations could incorporate confidence intervals to confirm statistical significance.

An alternative approach will very well be through the algorithms considered in [31]. It seems that these randomized approaches could be mapped to quantum computer architectures in a very effective and natural way. Such a method is presented in Section 3.6 above. A comprehensive study of the whole ecosystem of these randomized approaches is beyond the scope of this paper and will be presented elsewhere. We should note that several of these methods have been proposed long time and ignored by the majority of researchers. We believe that this was due to the fact that they could not be properly mapped on the von Neuman computing model and effectively implemented on the existing classical computers. We strongly believe that they have the potential to exhibit their supremacy in quantum computing systems.

Apart from the above-mentioned algorithms, there is a plethora of other related publications [10,32–52] which for various reasons we do not comment in our paper. The interested reader can refer to them in order to acquire a global view of the whole research ecosystem of the quantum solvers for linear algebraic systems.

Author Contributions
N.P. Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data;Writing—review & editing. M.V. Conceptualization,Characterization, Supervision, Writing—review & editing,Experimental design. All authors have read and agreed to the published version of the manuscript.

Funding
The authors received no financial support for the research, authorship, and/or publication of this article.

Conflicts of Interest Statement
The authors claim that they do not have any Conflict of Interest.

Data Availability Statement
All necessary data are included into the manuscript.

References

1. Ray LaPierre (2021). Shor algorithm. *Introduction to Quantum Computing*, pages 177–192.
2. McKinsey & Company. Quantum technology monitor. Technical report, April 2023.
3. John von Neumann and H. H. Goldstine (1947). Numerical inverting of matrices of high order. *Bulletin of the American Mathematical Society*, 53(11):1021–1099.
4. Alan Turing (1948). Rounding-off erros in matrix processes. *The Quarterly Journal of Mechanics and Applied Mathematics*, 1(1):287–308.
5. G. A. Hedayat (1993). Numerical linear algebra and computer architecture: An evolving interaction. Technical Report UMCS-93-1-5, University of Manchester.
6. Jack Dongarra (2002). Performance of various computers using standard linear equations software. *Computer Architecture News*, 20, 11.

7. Jessie Henderson, John Kath, John Golden, Allon Percus and Daniel O'Malley (2024). Addressing quantum's "fine print" with efficient state preparation and information extraction for quantum algorithms and geologic fracture networks. *Scientific Reports*, 14, 02.
8. John Golden, Daniel O'Malley and H. Viswanathan (2022). Quantum computing and preconditioners for hydrological linear systems. *Scientific Reports*, 12, 12.
9. Mauro E. S. Morales, Lirandë Pira, Philipp Schleich, Kelvin Koor, Pedro C. S. Costa, Dong An, Alán Aspuru-Guzik *et al*. (2025). Quantum linear system solvers: A survey of algorithms and applications.
10. Eleanor Rieffel and Wolfgang Polak (2000). An introduction to quantum computing for non-physicists. *ACM Computing Surveys (CSUR)*, 32(3):300–335.
11. Michael A. Nielsen and Isaac L. Chuang (2010). *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press.
12. Richard Cleve, Artur Ekert, Chiara Macchiavello and Michele Mosca (1998). Quantum algorithms revisited. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):339–354.
13. Aram W. Harrow, Avinatan Hassidim and Seth Lloyd (2009). Quantum algorithm for solving linear systems of equations. *Phys. Rev. Lett.*, 15(103).
14. Lov Grover and Terry Rudolph (2002). Creating superpositions that correspond to efficiently integrable probability distributions. *arXiv preprint quant-ph/0208112*.
15. Gerhard Hegerfeldt and Dirk Sondermann (1996). 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.
16. 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 (2013). Experimental quantum computing to solve systems of linear equations. *Physical review letters*, 110(23):230501.
17. Leonard Wossnig, Zhikuan Zhao and Anupam Prakash (2018). Quantum linear system algorithm for dense matrices. *Physical review letters*, 120(5):050502.
18. Iordanis Kerenidis and Anupam Prakash (2016). Quantum recommendation systems. *arXiv preprint arXiv:1603.08675*.
19. Mario Szegedy (2004). Quantum speed-up of markov chain based algorithms. In *45th Annual IEEE symposium on foundations of computer science*, pages 32–41. IEEE, 2004.
20. Changpeng Shao and Hua Xiang (2020). Row and column iteration methods to solve linear systems on a quantum computer. *Physical Review A*, 101(2):022322.
21. Yoshiyuki Saito, Xinwei Lee, Dongsheng Cai and Nobuyoshi Asai (2021). An iterative improvement method for hhl algorithm for solving linear system of equations. *arXiv preprint arXiv:2108.07744*.
22. Debasish Roy and Sambo Raj Chandra (2024). Quantum iterative algorithm for linear systems of equation. In *Science and Information Conference*, pages 560–575. Springer, 2024.
23. Vittorio Giovannetti, Seth Lloyd and Lorenzo Maccone (2008). Quantum random access memory. *Physical review letters*, 100(16):160501.
24. Carlos Bravo-Prieto, Ryan LaRose, M. Cerezo, Yigit Subasi, Lukasz Cincio and Patrick J. Coles (2023). Variational Quantum Linear Solver. *Quantum*, 7:1188.
25. George Forsythe and Richard Leibler (1950). Matrix inversion by a monte carlo method. *Mathematics of Computation*, 4(31):127–129.
26. IT Dimov, TT Dimov and TV Gurov (1998). A new iterative monte carlo approach for inverse matrix problem. *Journal of Computational and Applied Mathematics*, 92(1):15–35.
27. Chih-Chieh Chen, Shiue-Yuan Shiau, Ming-Feng Wu and Yuh-Renn Wu (2019). Hybrid classical-quantum linear solver using noisy intermediate-scale quantum machines. *Scientific reports*, 9(1):16251.
28. Richard Hamming (1950). Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160.
29. The variational quantum linear solver. https://learn.qiskit.org/course/ch-applications/the-variational-quantumlinear-solver. Accessed: 2023-06-14.
30. Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau *et al*. (2020). Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272.
31. Per-Gunnar Martinsson and Joel Tropp (2020). Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572.

32. Do Diep, Do Giang and Phan Phu (2018). Application of Quantum Gauss-Jordan Elimination Code to Quantum Secret Sharing Code. *International Journal of Theoretical Physics*, 57.
33. Koji Nagata, Tadao Nakamura, Han Geurdes, J Batle, Ahmed Farouk, Do Diep and Santanu Patro (2018). Efficient Quantum Algorithms of Finding the Roots of a Polynomial Function. *International Journal of Theoretical Physics*, 57.
34. Stefan Heinrich (2003). From Monte Carlo to quantum computation. *Math. Comput. Simul.*, 62:219–230.
35. Thomas Wong (2022). *Introduction to Classical and Quantum Computing*. Rooted Grove, Omaha, Nebraska.
36. Mihir Bhaskar, Stuart Hadfield, Anargyros Papageorgiou and Iasonas Petras (2016). Quantum algorithms and circuits for scientific computing. *Quantum Information and Computation*, 16(3&4):197–236.
37. Stefan Heinrich (2006). Numerical Analysis on a Quantum Computer. In Ivan Lirkov, Svetozar Margenov, and Jerzy Waśniewski, editors, *Large-Scale Scientific Computing*, pages 28–39, Berlin, Heidelberg, Springer Berlin Heidelberg.
38. Chi Zhang (2011). *Quantum Algorithms and Complexity for Numerical Problems*. PhD thesis, Columbia University.
39. Ngoc Do, Do Diep, Do Giang and Nguyen Minh (2017). Quantum Gauss-Jordan Elimination and Simulation of Accounting Principles on Quantum Computers. *ViAsM16.01*, 56.
40. Hefeng Wang and Hua Xiang (2019). A quantum eigensolver for symmetric tridiagonal matrices. *Quantum Information Processing*, 18(3).
41. Giacomo Nannicini (2020). An introduction to quantum computing, without the physics. *SIAM Review*, 62(4).
42. Siddhartha Srivastava and Veera Sundararaghavan (2019). Box algorithm for the solution of differential equations on a quantum annealer. *Physical Review A*, 99(5):52355.
43. Lin Lin (2022). Lecture Notes on Quantum Algorithms for Scientific Computation. *arXiv preprint arXiv:1909.05820*, 1.
44. Matthias Möller and Cornelis Vuik (2017). On the Impact of Quantum Computing Technology on Future Developments in High-Performance Scientific Computing. *Ethics and Inf. Technol.*, 19(4):253–269, 12.
45. Guillermo González, Rahul Trivedi and J. Ignacio Cirac (2021). Quantum algorithms for powering stable Hermitian matrices. *Physical Review A*, 103(6).
46. Stuart Hadfield (2018). Quantum Algorithms for Scientific Computing and Approximate Optimization. *arxiv.org*, 5.
47. Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe and Seth Lloyd (2017). *Quantum machine learning*, 9.
48. Juan José García Ripoll (2021). Quantum-inspired algorithms for multivariate analysis: From interpolation to partial differential equations. *Quantum*, 5.
49. Piotr Gawron, Dariusz Kurzyk and Łukasz Pawela (2018). QuantumInformation.jl—A Julia package for numerical computation in quantum information theory. *PLoS ONE*, 13(12).
50. Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward *et al*. (2022) The variational quantum eigensolver: a review of methods and best practices. *Physics Reports*, 986:1–128.
51. Lin Lin and Yu Tong (2020). Optimal polynomial based quantum eigenstate filtering with application to solving quantum linear systems. *Quantum*, 4:361.
52. Andrew Childs, Robin Kothari and Rolando Somma (2017). Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950.