Quantum Algorithm for Solving Tri-Diagonal Linear Systems of Equations

Master thesis

submitted in partial fulfillment of
the requirements for the degree of

Master of Science in Mathematics

ETH Zürich

Almudena Carrera Vázquez

Supervisors:
Dr. Stefan Wörner[1]
Prof. Dr. Ralf Hiptmair[2]

November 27, 2018

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

[1]IBM Research, Zürich
[2]Seminar for Applied Mathematics, ETH Zürich

Abstract

Numerical simulations, optimisation problems, statistical analysis and computer graphics are only a few examples from the wide range of real-life applications which rely on solving large systems of linear equations. The best classical methods can approximate the solution of sparse systems in time $\mathcal{O}(\sqrt{N}\kappa)$, where N denotes the number of unknowns and κ its condition number. In 2009, A. Harrow, A. Hassidim and S. Lloyd (HHL) proposed a quantum algorithm with a running time of $poly(\log N, \kappa)$ under the assumptions of the availability of efficient methods for loading the data, Hamiltonian simulation and extracting the solution. This thesis presents implementations for the missing oracles and analyzes the overall performance of the algorithm. A complete implementation of the HHL algorithm running in $poly(\log N, \kappa)$ is given for the case of a special class of tri-diagonal symmetric matrices arising from Finite Difference methods for two-point Boundary Value Problems. Full analyses of the mathematical approximations obtained and the circuit depths are also included.

Contents

1 Introduction

Systems of linear equations arise naturally in many real-life applications in a wide range of areas. An interdisciplinary example is the study of the flow of some quantity through a network. It can be the flow of traffic in a grid of city streets, the current flow through electrical circuits, or the distribution of products from manufacturers to consumers through a network of wholesalers and retailers. In particular, symmetric matrices arise more often in applications than any other major class of matrices[44, Chapter-7], as it occurs in many cases with the solution of Partial Differential Equations. Another practical application involving a system associated to a symmetric matrix is the analysis of the images from a satellite. An effective method to suppress redundant information and provide in one or two composite images most of the information is the Principal Component Analysis, which tries to find a linear combination of the images. The size of the systems arising from these type of situations is usually very large. However, even the best classical general purpose method, the conjugate gradient, has a runtime complexity of $\mathcal{O}(Ns\kappa\log(1/\epsilon))$, where N denotes the size of the system, s the maximum number of non-zero entries in a row or column, κ the condition number and ϵ the precision [33]. Therefore, areas where the amount of data to be processed is growing need faster methods.

A promising aspirant to this end is Quantum Computing. At the time of writing, there is a big gap between the theory and the existing hardware. The technology needed for quantum computations is still in its infancy, although it is already possible to run experiments in small chips. However, for small computations, classical computers outperform their quantum counterparts. The promising potential in the field of Quantum Computing lies in the computation of large amounts of data. Nonetheless, the hope is that one day the technology will be mature enough to run complex algorithms. In the meantime, the development of the theory will continue so that everything is ready when that day arrives.

This thesis had two main goals. One, to investigate from a mathematical point of view an algorithm proposed by A. Harrow, A. Hassidim and S. Lloyd ([1]) to solve systems of linear equations with a quantum computer. In contrast with the running time of the best classical algorithm, their algorithm has a running time complexity of $\mathcal{O}(\log(N)s^2\kappa^2/\epsilon)$. The other goal of the thesis was a practical implementation with the Qiskit open source software from IBM ([42]) Qiskit is based on Python and allows to run either local simulations, or simulations on a real device. In the end, a complete implementation of the algorithm was achieved and the mathematical results complemented with the data from the simulations.

The structure of this thesis is as follows. Section 2 is a short introduction to quantum computing from a mathematical point of view. Section 3 is a review of the existing work on solving systems of linear equations with a quantum computer. Section 4 gives an overview of the complete algorithm. Sections 5 to 8 discuss the different stages of the algorithm: state preparation, Hamiltonian simulation, eigenvalue inversion and observables. Finally, an overall mathematical analysis is given in Section 9, and Section 11 gives some guidelines for possible further work followed by a conclusion.

2 Axiomatic quantum computing

The aim of this section is to provide enough mathematical background for the reader. The material presented is a combination of Sections 2-3 of [6], and Chapter 2.1.-2.2. of [3]. The former, incorporating explanations of the basic algorithms, is still a Mathematics paper and does not require previous knowledge of Physics. The book by Nielsen and Chuang is regarded as the standard text in the subject and encompasses both Quantum Computing and Quantum Information.

Since interest in Quantum Computing might come from different fields, the idea is to not assume a deep knowledge of Mathematics while giving only the strictly necessary definitions. The structure is as follows. Section 2.1 presents the tools from linear algebra, then Section 2.2 introduces the basic elements of quantum computing as mathematical entities, and Section 2.3 is dedicated to quantum measurements. The notation adopted to denote vectors and operators throughout this work is that of Quantum Mechanics, as it is conventional in Quantum Computing. Thus, $|v\rangle$ will denote a vector in a vector space. In this field it is also common to start counting from 0 and this convention has been preserved even in the mathematical definitions for the sake of consistency.

2.1 Linear algebra preliminaries

We begin by defining the mathematical framework for developing the theory of Quantum Computing. We will be interested in complex vector spaces.

Definition 2.1. Given a complex vector space $\mathbb{C}^n$, $n \in \mathbb{N}$, a **ket** is a column vector $|v\rangle \in \mathbb{C}^n$. A **bra** $|v\rangle \in (\mathbb{C}^n)^*$ is a vector in the dual space and can be thought as the transpose conjugate of $|v\rangle$.

Example 1.
Let $|v\rangle = \begin{pmatrix} 1-i \\ 2 \\ -\frac{i}{2} \\ 0 \end{pmatrix} \in \mathbb{C}^4$. Then its bra is given by $\langle v| = \begin{pmatrix} 1+i, & 2, & \frac{i}{2}, & 0 \end{pmatrix}$.

With this notation, the **braket** $\langle y|x\rangle$ of two vectors $|x\rangle, |y\rangle \in \mathbb{C}^n$ denotes the standard inner product in $\mathbb{C}^n$. Thus, if $|x\rangle = (x_0, x_1, ..., x_{n-1})^T$ and $|y\rangle = (y_0, y_1, ..., y_{n-1})^T$, then

$$\langle y|x\rangle := (|y\rangle, |x\rangle) = \begin{pmatrix} y_0^*, & y_1^*, & ..., & y_{n-1}^* \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix} = \sum_{i=0}^{n-1} y_i^* x_i \in \mathbb{C}^n, \tag{1}$$

where for $z \in \mathbb{C}$, z^* denotes the complex conjugate.

Definition 2.2. Given two complex vector spaces $V \subset \mathbb{C}^m$ and $W \subset \mathbb{C}^m$, $m, n \in \mathbb{N}$, with bases $|e_0\rangle$,$|e_1\rangle$, ...,$|e_{m-1}\rangle$ and $|f_0\rangle, |f_1\rangle, ..., |f_{n-1}\rangle$ respectively, the **tensor product** $V \otimes W$ is another complex vector space of dimension mn. The tensor product space is equipped with a bilinear operation $\otimes : V \times W \to V \otimes W$. The vector space $V \otimes W$ has basis $|e_i\rangle \otimes |f_j\rangle \, \forall i = 0, 1, ..., m-1, j = 0, 1, ..., n-1$.

For a tensor product of vectors many times we will omit the $\otimes$ symbol, thus $|v\rangle\,|w\rangle := |v\rangle \otimes |w\rangle$.

Definition 2.3. Given $A \in \mathbb{C}^{m\times n}, B \in \mathbb{C}^{p\times q}$, the **Kronecker product** $A \otimes B$ is the matrix defined as:

$$D := A \otimes B = \begin{pmatrix} a_{00}B & \cdots & a_{0n-1}B \\ a_{10}B & \cdots & a_{2n}B \\ \vdots & & \vdots \\ a_{m-10}B & \cdots & a_{m-1n-1}B \end{pmatrix}. \tag{2}$$

If we choose the standard basis over the vector spaces $\mathbb{C}^{m\times n}$ and $\mathbb{C}^{p\times q}$, then the bilinear operation $\otimes$ of the tensor product $\mathbb{C}^{m\times n} \otimes \mathbb{C}^{p\times q}$ is simply the Kronecker product.

Proposition 2.1. *Let $A, B \in \mathbb{C}^{m\times m} C, D \in \mathbb{C}^{n\times n}$ be linear transformations on $\mathbb{C}^m$ and $\mathbb{C}^n$ respectively, $|u\rangle, |v\rangle \in \mathbb{C}^m$, $|w\rangle, |x\rangle \in \mathbb{C}^n$, and $a, b \in \mathbb{C}$. The tensor product satisfies the following properties:*

(i) $(A \otimes C)(B \otimes D) = AB \otimes CD$.

(ii) $(A \otimes C)(|u\rangle \otimes |w\rangle) = A\,|u\rangle \otimes C\,|w\rangle$.

(iii) $(|u\rangle + |v\rangle) \otimes |w\rangle = |u\rangle \otimes |w\rangle + |v\rangle \otimes |w\rangle$.

(iv) $|u\rangle \otimes (|w\rangle + |x\rangle) = |u\rangle \otimes |w\rangle + |u\rangle \otimes |x\rangle$.

(v) $a\,|u\rangle \otimes b\,|w\rangle = ab\,|u\rangle \otimes |w\rangle$.

(vi) $(A \otimes C)^\dagger = A^\dagger \otimes C^\dagger$. *(See Definition 2.8.)*

The vector spaces used in quantum computing are of the form $\left(\mathbb{C}^2\right)^{\otimes n}$, where

$$\left(\mathbb{C}^2\right)^{\otimes n} := \underbrace{\mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2}_{n \text{ times}} \quad \text{and} \quad n \in \mathbb{N}. \tag{3}$$

This notation is the same for operators. We now specify the basis elements and their notation for these spaces.

Definition 2.4. The standard basis for $\mathbb{C}^2$ is denoted by $|0\rangle_1 = \begin{pmatrix}1\\0\end{pmatrix}$ and $|1\rangle_1 = \begin{pmatrix}0\\1\end{pmatrix}$. The standard basis for $\left(\mathbb{C}^2\right)^{\otimes n}$, which has 2^n elements, is denoted by $|0\rangle_n, |1\rangle_n, ..., |2^n - 1\rangle_n$.

Example 1. We write explicitly the basis elements of $\left(\mathbb{C}^2\right)^{\otimes 2} = \mathbb{C}^2 \otimes \mathbb{C}^2$:

$$|0\rangle := |0\rangle_2 = |0\rangle \otimes |0\rangle = |00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |1\rangle := |1\rangle_2 = |0\rangle \otimes |1\rangle = |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \tag{4}$$

$$|2\rangle := |2\rangle_2 = |1\rangle \otimes |0\rangle = |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad |3\rangle := |3\rangle_2 = |1\rangle \otimes |1\rangle = |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{5}$$

$$\tag{6}$$

Remarks. Example 1 introduces the alternative but equivalent notations for the basis states. Throughout this work we will switch between them accordingly to the context and without further warning.

(i) In many cases, when n is known, the subscript is dropped and we write $|i\rangle$ in place of $|i\rangle_n$ for the i^{th} basis state.

(ii) There are situations (namely the *qubits*, which will be introduced in Section 2.2) where it is more convenient to switch to a binary notation for the basis states, e.g. $|010\rangle$ denotes $|2\rangle_3$.

The outer product in the braket notation will be a convenient way for writing matrices and calculating products with other matrices or vectors. We now give the definition.

Definition 2.5. Let V and W be linear subspaces of $\mathbb{C}^n$ and $\mathbb{C}^m$, respectively, where $n, m \in \mathbb{N}$. Let $|v\rangle \in V$ and $|w\rangle \in W$. Then the **outer product** $(|w\rangle\langle v|) : V \to W$ is a linear operator whose action is defined by

$$(|w\rangle\langle v|)(|v'\rangle) := |w\rangle\langle v|v'\rangle = \langle v|v'\rangle |w\rangle \quad \text{for} \quad |v'\rangle \in V. \tag{7}$$

Definition 2.6. Let $V \subset \mathbb{C}^n, n \in \mathbb{N}$, be a complex vector space. Then I_V denotes the identity operator in V. The notation $I_k, k \in \mathbb{N}$ will be used to denote the identity operator in $\mathbb{C}^{2^k}$, i.e. the $2^k \times 2^k$ identity matrix.

Remark. For $k = 1$ the subscript will be often dropped, therefore in this text $I := I_1$.

Example 2.

$$I := I_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad I_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{8}$$

Theorem 2.1 (Completeness relation)**.** *Let V be a complex inner product space and let $|v_0\rangle, |v_1\rangle, ..., |v_{n-1}\rangle$ be an orthonormal basis for V. Then*

$$I_V = \sum_{i=0}^{n-1} |v_i\rangle\langle v_i|. \tag{9}$$

Proof. Let $|w\rangle \in V$ be an arbitrary vector. It can be written $|w\rangle = \sum_{i=0}^{n-1} w_i |v_i\rangle$ for some $w_i \in \mathbb{C}$. . Note that $\langle v_i|w\rangle = w_i$, therefore

$$\left(\sum_{i=0}^{n-1} |v_i\rangle \langle v_i|\right) |w\rangle = \sum_{i=0}^{n-1} |v_i\rangle \langle v_i|w\rangle \sum_{i=0}^{n-1} |v_i\rangle w_i = |w\rangle . \tag{10}$$

□

We will be mostly interested in the equation

$$I_k = \sum_{i=0}^{2^k-1} |i\rangle \langle i| . \tag{11}$$

Definition 2.7. Let V be a complex vector space and $A : V \to V$ a linear operator on V (i.e. a square matrix). A **diagonal representation** for A is

$$A = \sum_{i=0}^{n-1} \lambda_i |v_i\rangle \langle v_i| , \tag{12}$$

where the vectors $|v_i\rangle$,$0 \leq i \leq n-1$ form an orthonormal set of eigenvectors for A with corresponding eigenvalues λ_i. An operator is said to be **diagonalisable** if it has a diagonal representation.

Definition 2.8. Let A be a linear operator on a Hilbert space, V. Then, its **adjoint** or **Hermitian conjugate**, $A^\dagger$, exists and is the unique operator on V such that

$$(|v\rangle , A|w\rangle) = \left(A^\dagger |v\rangle , |w\rangle\right) \qquad \forall |v\rangle , |w\rangle \in V. \tag{13}$$

The above introduces the *dagger* notation from quantum mechanics. Which, in matrix representation, is no other than the transpose conjugate of a matrix: $A^\dagger := (A^*)^T$.

Definition 2.9. Let A be a linear operator on a Hilbert space, V. Then A is said to be

(i) **normal** when $AA^\dagger = A^\dagger A$,

(ii) **Hermitian** or **self-adjoint** when $A = A^\dagger$ (a normal matrix is hermitian iff its eigenvalues are real),

(iii) **unitary** when $AA^\dagger = I$.

The next theorem is taken from Section-2.1.7 of [3] and stated without proof.

Theorem 2.2 (Spectral decomposition)**.** *Any normal operator A on a vector space V is diagonal with respect to some orthonormal basis for V . Conversely, any diagonalisable operator is normal.*

We can now define functions of normal matrices.

Definition 2.10. Let $A \in \mathbb{C}^{n\times n}$ be a normal matrix with eigenvalues λ_i and respective orthonormal eigenvectors $|v_i\rangle$. Let $f : \mathbb{C} \to \mathbb{C}$. Then

$$f(A) := \sum_{i=0}^{n-1} f(\lambda_i) |v_i\rangle \langle v_i| . \tag{14}$$

Example 2. In particular, with A as in Definition 2.10,

$$e^{iAt} = \sum_{i=0}^{n-1} e^{i\lambda_i t} \left|v_i\right\rangle \left\langle v_i\right| . \tag{15}$$

The last piece of linear algebra we will need is the condition number of a matrix associated to a linear system of equations $A\left|x\right\rangle = \left|b\right\rangle$. Roughly speaking, this quantity is the rate at which the solution, $\left|x\right\rangle$, will change with respect to a change in $\left|b\right\rangle$. Before addressing the condition number, we will need the following definitions which can be found in Chapter 5 from [2].

Definition 2.11. The **p-norm**, for $p \geq 1$, of $\left|x\right\rangle \in \mathbb{C}^n, n \in \mathbb{N}$, is defined as $\|\left|x\right\rangle\|_p = \left(\sum_{i=0}^{n-1} |x_i|^p\right)^{1/p}$.

Definition 2.12. A vector norm,$\|\cdot\|$, that is defined on $\mathbb{C}^p$ for $p = m, n$, induces a matrix norm on $\mathbb{C}^{m\times n}$ by setting

$$\|A\| = \max_{\|\left|x\right\rangle\|=1} \|A\left|x\right\rangle\| \quad \text{for} A \in \mathbb{C}^{m\times n}, \left|x\right\rangle \in \mathbb{C}^{n\times 1}. \tag{16}$$

It is called the **induced matrix norm**.

Remark. An equivalent definition for the induced matrix norm is given by the expression

$$\|A\| = \sup_{\left|x\right\rangle\neq 0} \frac{\|A\left|x\right\rangle\|}{\|\left|x\right\rangle\|}. \tag{17}$$

Proposition 2.2. *An induced matrix norm is compatible with its underlying vector norm in the sense that*

$$\|A\left|x\right\rangle\| \leq \|A\|\|\left|x\right\rangle\|. \tag{18}$$

Now we can define the condition number. The following is taken from Chapter 3 from [4], which also contains more on the topic.

Definition 2.13. The **condition number** of a matrix $A \in \mathbb{C}^{n\times n}$is defined as

$$\kappa(A) = \|A\|\left\|A^{-1}\right\|, \tag{19}$$

where $\|\cdot\|$ is an induced matrix norm.

The condition number depends on the choice of norm. In this report we use the 2-norm, in which case, and for A normal, it is also defined as

$$\kappa(A) = \frac{|\lambda_{\max}(A)|}{|\lambda_{\min}(A)|}. \tag{20}$$

Here $\lambda_{\max}$ and $\lambda_{\min}$ denote the maximal and minimal eigenvalues of A respectively. The following result will be useful in the error analysis of the state preparation later in this text.

Proposition 2.3. *Let $A \in \mathbb{C}^{n\times n}$ be a nonsingular matrix and $\left|x\right\rangle, \left|b\right\rangle, \left|\epsilon\right\rangle \in \mathbb{C}^n, n \in \mathbb{N}$. Then*

$$\frac{\left\|A^{-1}\left|\epsilon\right\rangle\right\|}{\|A^{-1}\left|b\right\rangle\|} \leq \kappa(A)\frac{\|\left|\epsilon\right\rangle\|}{\|\left|b\right\rangle\|}. \tag{21}$$

2.2 Building blocks of Quantum Computing

A qubit is to a quantum computer what a bit is to a classical one. In a similar way that a computer has a register made of bits, the quantum computer has a register made of qubits. We now define these.

Definition 2.14. Mathematically, a **qubit** is represented by a unitary vector $|q\rangle \in \mathbb{C}^2$. It can be written as

$$|q\rangle = \alpha |0\rangle + \beta |1\rangle, \quad \text{where} \quad |\alpha|^2 + |\beta|^2 = 1 \quad \text{and} \quad \alpha, \beta \in \mathbb{C}. \tag{22}$$

For the following definition we encounter something that is common from the field of Computer Science: indexes run starting from 0.

Definition 2.15. The **state of n qubits**, $|q_0\rangle, |q_1\rangle, ..., |q_{n-1}\rangle$, is a unitary vector $|\psi\rangle \in \mathbb{C}^{2^n}$:

$$|\psi\rangle = |q_{n-1}\rangle \otimes \cdots \otimes |q_1\rangle \otimes |q_0\rangle. \tag{23}$$

It can be represented in the standard basis as

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle \quad \text{where} \quad \alpha_i \in \mathbb{C} \quad \text{and} \quad \sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1. \tag{24}$$

The **state of a register** consisting of m qubits is the state of those m qubits.

Remark. It is crucial to highlight the word *unitary* in the definitions of qubit, state of n qubits and state of a register. Throughout this work, whenever we work with the state of the register at a particular stage of an algorithm, it will always be assumed normalised.

Sometimes, however, it is more illustrative to work with $C|\psi\rangle$, where C is a constant, or to omit the normalisation factors altogether. Nonetheless, the reader should acknowledge the implicit normalisation.

When drawing a quantum circuit, wires represent qubits. In this report we use the convention that, with the notation from Definition 2.15, the bottom wire represents $|q_0\rangle$ while the top wire is for $|q_{n-1}\rangle$. Thus, for example, Eq. 23 is the state of the register drawn in Figure 1.

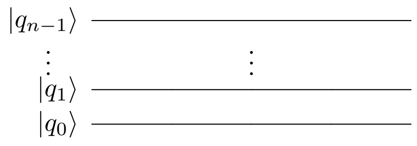


Figure 1: Representation of a register of n qubits.

Definition 2.16. We say that n qubits are in a **basis state** if their state $|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$ is such that $\exists k : |\alpha_k| = 1, \alpha_i = 0 \quad \forall i \neq k$. Otherwise, we say that they are in a **superposition**.

Furthermore, any linear combination $\sum_i \alpha_i |\psi_i\rangle$ is said to be a **superposition of the states** $|\psi_i\rangle$ with amplitude α_i for the state $|\psi_i\rangle$.

Definition 2.17. A quantum state $|\psi\rangle \in \mathbb{C}^{2^n}$ is **decomposable** if it can be expressed as a tensor product $|\psi_1\rangle \otimes \cdots \otimes |\psi_k\rangle$ of $k > 2$ quantum states on $n_1, ..., n_k$ qubits respectively, with the property that $n_1 + ... + n_k = n$.

Definition 2.18. Let $n, k \in \mathbb{N}$. A quantum state $|\psi\rangle \in \mathbb{C}^{2^n}$ is a **product state** if it is decomposable into the tensor product of n single-qubit quantum states. Otherwise, it is **entangled**.

Quantum gates are the counterpart of logical gates in a classical computer. They must satisfy certain conditions to abide by the laws of quantum mechanics, which state that the evolution of quantum systems is described by unitary operators. Remarkably, this means that the possible operations on qubits will be reversible, which could have boded dubious any computational speed up from this field already from the beginning. Fortunately, C. H. Bennett showed in 1973 in [5] that it is possible to make any computation reversible, and that it is possible to do so with only a polynomial overhead in time and space.

Theorem 2.3 (Bennett, 1973)**.** *For every standard one-tape Tuning machine S, there exists a three-tape reversible, deterministic Turing machine R such that if I and P are strings on the alphabet of S, containing no embedded blanks, then S halts on I if and only if R halts on $(I; B; B)$, and $S : I \to P$ if and only if $R : (I; B; B) \to (I; B; P)$.*

Furthermore, if S has f control states, N quintuples and a tape alphabet of z letters, including the blank, R will have $2f + 2N + 4$ states. $4N + 2z + 3$ quadruples and tape alphabets of z, $N + 1$, and z letters, respectively. Finally. if in a particular computation S requires ν steps and uses s squares of tape, producing an output of length λ, then R will require $4\nu + 4\lambda + 5$ steps, and use s, $\nu + 1$, and $\lambda + 2$ squares on its three tapes, respectively. (Where $\nu >> s$, the total space requirement can be reduced to less than $2\sqrt{\nu s}$).)

We now give the formal definition for quantum gates.

Definition 2.19. A **quantum gate** on n qubits is a unitary matrix $U \in \mathbb{C}^{2^n \times 2^n}$.

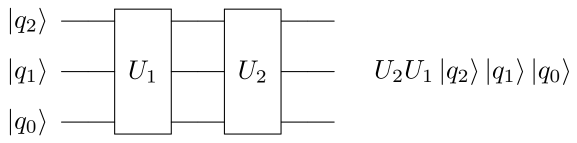


Figure 2: Quantum gates on three qubits.

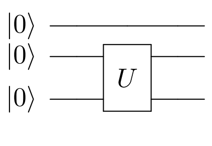
Operations on qubits are represented as gates, which in the circuit are drawn as boxes, as shown in Figure 2. Circuits are read from left to right, but when writing the mathematical expression corresponding to a circuit, gates are written from right to left. For example, in the circuit from Figure 2, U_1 is the first gate to be applied on the initial state $|\psi\rangle = |q_2\rangle |q_1\rangle |q_0\rangle$.

Remark. Identity gates are usually omitted when drawing a circuit, and gates on the same vertical line become a tensor product in a mathematical equation.

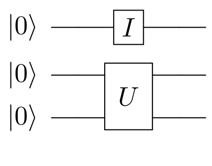
Example 3. The circuits drawn in Figure 3 are equivalent. The mathematical expression describing their action is

$$(I \otimes U)\,(|0\rangle \otimes |0\rangle \otimes |0\rangle) = |0\rangle \otimes U\,|00\rangle\,. \tag{25}$$

However, the preferred notation will be that of Figure 3a.



(a) Unitary gate and implicit identity gate.



(b) Unitary gate and identity gate.

Figure 3: Two equivalent circuits.

Of particular interest are one-qubit gates, that is unitary matrices in $\mathbb{C}^{2\times 2}$. Among those, the *Pauli gates* and the *Hadamard gate* should be highlighted.

Definition 2.20. The **Pauli gates** are the following single-qubit gates:

$$X = \text{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}; \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}; \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{26}$$

The Pauli matrices exponentiated give rise to three important operators.

Definition 2.21. The **rotation operators** about the $\hat{x}, \hat{y}$ and $\hat{z}$ axes are defined by the equations

$$R_x(\theta) := e^{-i\theta X/2} = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) X = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}, \tag{27}$$

$$R_y(\theta) := e^{-i\theta Y/2} = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) Y = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}, \tag{28}$$

$$R_z(\theta) := e^{-i\theta Z/2} = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) Z = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}. \tag{29}$$

Example 4. The X or NOT gate is the quantum counterpart to the logical NOT gate. Its action is to flip the state of a qubit:

$$X\,|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle\,, \tag{30}$$

$$X\,|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle\,. \tag{31}$$

Definition 2.22. The **Hadamard gate** is the one-qubit gate defined by

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{32}$$

Lemma 2.4. *Let $\bullet_n$ denote the bitwise dot product. Then the action of $H^{\otimes n}$ on the n-qubit state $|x\rangle_n$ is*

$$H^{\otimes n} |x\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} (-1)^{k \bullet_n x} |k\rangle_n . \tag{33}$$

In particular, if $|x\rangle_n = |0\rangle_n$, applying $H^{\otimes n}$ yields a uniform superposition of basis states:

$$H^{\otimes n} |0\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle_n . \tag{34}$$

Proof. We first compute specifically the action of H on one qubit.

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) , \tag{35}$$

$$H|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) . \tag{36}$$

This can be summarised as

$$H|x\rangle_1 = \frac{1}{\sqrt{2}} (|0\rangle + (-1)^x |1\rangle) = \frac{1}{\sqrt{2}} \sum_{k=0}^{1} (-1)^{k \cdot x} |k\rangle_1 . \tag{37}$$

Therefore, writing the binary string $x \coloneqq x_{n-1}...x_1 x_0$ for $x_i \in \{0,1\}$, one has

$$H^{\otimes n} |x\rangle_n = \bigotimes_{i=0}^{n-1} \frac{1}{\sqrt{2}} (|0\rangle + (-1)^x |1\rangle) = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} (-1)^{k \bullet_n x} |k\rangle_n \tag{38}$$

□

Remark. When it is clear from the context, we will write a simple dot instead of $\bullet_n$ to refer to the bitwise dot product, i.e. in those cases $k \cdot x \coloneqq k \bullet_n x$.

The power of $H^{\otimes n}$ is that further operations will be simultaneously applied to all basis states. The following example shows this for the two-qubit case.

Example 5. Let U be a two-qubit gate. And consider the circuit from Figure 4.

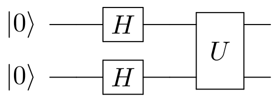


Figure 4: Two Hadamards and a two-qubit gate.

Mathematically, the effect of this circuit is

$$U(H \otimes H)(|0\rangle \otimes |0\rangle) = U\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \tag{39}$$

$$= \frac{1}{2} \sum_{i=0}^{3} U |i\rangle . \tag{40}$$

Another two important single-qubit gates are the phase gate (S) and the $\pi/8$ gate (T):

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}; \quad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i/4} \end{pmatrix}. \tag{41}$$

The analogous quantum *if* logical operation is a *controlled* gate. The prototypical controlled operation is the CNOT or controlled-NOT gate (CX for short), which flips the target qubit if the control qubit is $|1\rangle$.

Definition 2.23. The **CNOT** gate (**CX**) is a two-qubit gate with two input qubits: the control qubit and the target qubit. Let $\oplus$ denote addition modulus 2, $|c\rangle$ the control qubit and $|t\rangle$ the target qubit. Then the action of the CNOT gate is defined as

$$|c\rangle |t\rangle \mapsto |c\rangle |t \oplus c\rangle. \tag{42}$$

The matrix representation is given by

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{43}$$

And the circuit representation is shown in Figure 5, with the top line representing the control qubit and the bottom the target qubit.

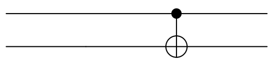


Figure 5: CNOT gate.

One can also define controlled operations in which the action takes place when the control qubit is $|0\rangle$. This is represented by an empty circle as shown in figure 6.

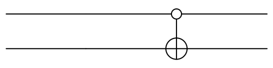


Figure 6: CNOT gate controlled by 0.

Multiple controls are allowed, as well as controlled multi-qubit gates. For example, the circuit from Figure 7 is a representation of a two-qubit gate U that will be applied on qubits $|q_2\rangle$ and $|q_1\rangle$ if $|q_0\rangle = 0$ and $|q_3\rangle = 1$.

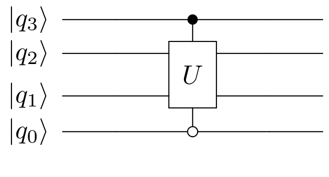


Figure 7: Multi-controlled two-qubit gate.

Real quantum computing devices can only implement operations from a finite set of gates. The following theorem is proved in Section 4.5.3. of [3] and it states that we can approximate any unitary operation with a small set of gates, provided these are well-chosen.

Definition 2.24. Let U be a unitary operation and V an approximation to U. The **approximation error** is defined as

$$\max_{\||\psi\rangle\|=1} \|(U-V)\,|\psi\rangle\|. \tag{44}$$

Definition 2.25. The **standard set** of universal gates consists of Hadamard, phase, controlled-NOT (CNOT), and $\pi/8$ gates.

The following theorem explains why the standard set is called universal.

Theorem 2.5. *An arbitrary single qubit gate may be approximated to an accuracy ϵ using $\mathcal{O}\left(\log^2(1/\epsilon)\right)$ gates from the standard set.*

There exist other combinations of gates which as a set are universal, however we are interested in the standard, since it is the one available in Qiskit software from IBM ([42]) used to implement this project.

The following are the most important gate decompositions that we will use in the upcoming sections for the gate count analyses of the different algorithms. These are taken from Chapter 4.3. of [3].

- Toffoli: 6CNOTs +7T +2H +1S = 6·CNOTs+10 · {one-qubit gate}.

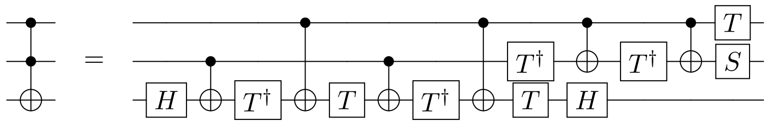


Figure 8: Toffoli gate and its decomposition.

- C–U: 2·CNOTs + 4 · {one-qubit gate}.

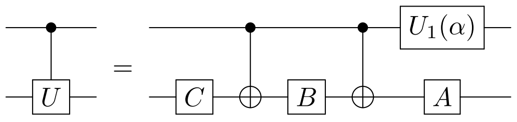


Figure 9: One-controlled single-qubit gate and its decomposition. α, A, B and C satisfy $U = e^{i\alpha}AXBXCX, ABC = I$.

- C^n–U: $(12n-10)$·CNOTs+$(20n-16)$·{one-qubit gate}+$(n-1)$·{ancilla}.

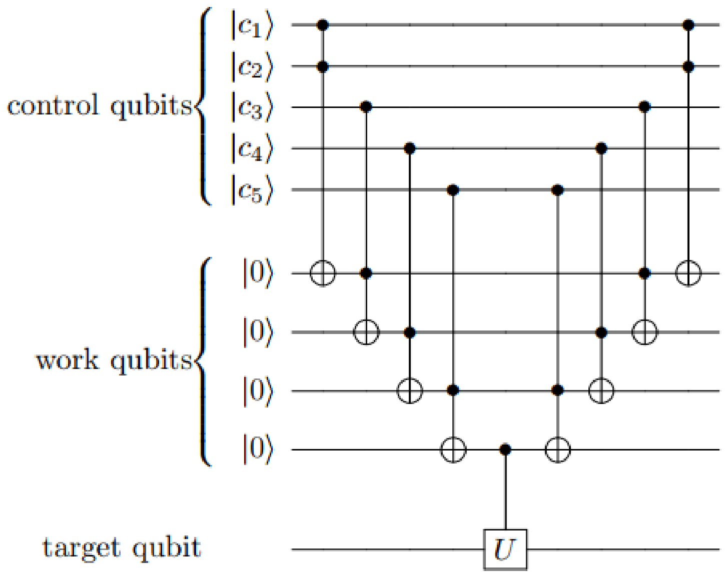


Figure 10: Multi-controlled single-qubit gate and its decomposition for $n = 5$.

2.3 Measurements

Measurements are the procedure to obtain information from a quantum state. In this case, however, there is no classical analogue. The typical explanation is that when a person performs a measurement on a quantum state $|\psi\rangle = \sum_i \alpha_i |i\rangle$, it causes it to *collapse* into one of the $|i\rangle$ with probability $|\alpha_i|^2$ (this means that the person instead of observing the quantum state $|\psi\rangle$ will see $|i\rangle$).

There is no good understanding yet as to why it happens or what does it really mean. There are different interpretations, and discussions on the topic are likely to deviate towards a more philosophical ground. Nonetheless, there does exist a general agreement: that the mathematics used to describe this bewildering event predict well the outcome of experiments.

Definition 2.26. A **collection of measurement operators on n qubits** is a set $\{M_m\} \subset \mathbb{C}^{2^n \times 2^n}, m \in I \subset \mathbb{N}$, such that its elements satisfy the *completeness equation*

$$\sum_m M_m^\dagger M_m = I_n \tag{45}$$

Definition 2.27. A **quantum measurement** by a collection of measurement operators $\{M_m\}$ on n qubits is a map

$$\{|\psi\rangle \in \mathbb{C}^{2^n} : \langle\psi|\psi\rangle = 1\} \to \{|\phi\rangle \in \mathbb{C}^{2^n} : \langle\phi|\phi\rangle = 1\} \tag{46}$$

that sends a quantum state on n qubits $|\psi\rangle$ to the n-qubit state

$$\frac{M_m |\psi\rangle}{\sqrt{\langle\psi| M_m^\dagger M_m |\psi\rangle}} \tag{47}$$

with probability

$$p(m) = \langle\psi| M_m^\dagger M_m |\psi\rangle \,. \tag{48}$$

The state from Eq. 47 is referred to as the post-measurement state with m being the **outcome of the measurement**.

Remark. The completeness equation expresses the fact that the probabilities sum up to one:

$$\sum_m p(m) = \sum_m \langle\psi| M_m^\dagger M_m |\psi\rangle = 1. \tag{49}$$

We will refer to quantum measurements as measurements. Sometimes we are interested in measuring only a few qubits from a quantum state. In that case the measurement operators can be tensored with identity matrices to obtain measurement operators in the state space. The next example illustrates the case for one qubit measurements.

Example 6. An important example of a measurement is the *measurement of a qubit in the computational basis.* This is a measurement of a single qubit with two outcomes defined by the two measurement operators $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$. Suppose that we have the state

$$|\psi\rangle = \alpha_0 |00\rangle + \alpha_1 |01\rangle + \alpha_2 |10\rangle + \alpha_3 |11\rangle \,, \tag{50}$$

where $\sum_i |\alpha_i|^2 = 1$ and $\alpha_i \in \mathbb{C}$. To measure the last qubit in the computational basis we would use the collection of measurement operators $\{M_0 = I \otimes |0\rangle\langle 0|\,, M_1 = I \otimes |1\rangle\langle 1|\}$. First, we check that the completeness equation is satisfied:

$$\begin{aligned} M_0^\dagger M_0 &= (I \otimes |0\rangle\langle 0|)^\dagger (I \otimes |0\rangle\langle 0|) = \left(I^\dagger \otimes (|0\rangle\langle 0|)^\dagger\right)(I \otimes |0\rangle\langle 0|) && (51)\\ &= (I \otimes |0\rangle\langle 0|)^2 = I \otimes |0\rangle\langle 0| = M_0, && (52)\\ M_1^\dagger M_1 &= (I \otimes |1\rangle\langle 1|)^\dagger (I \otimes |1\rangle\langle 1|) = \left(I^\dagger \otimes (|1\rangle\langle 1|)^\dagger\right)(I \otimes |1\rangle\langle 1|) && (53)\\ &= (I \otimes |1\rangle\langle 1|)^2 = I \otimes |1\rangle\langle 1| = M_1. && (54) \end{aligned}$$

Therefore,

$$\begin{aligned} M_0^\dagger M_0 + M_1^\dagger M_1 &= M_0 + M_1 = I \otimes |0\rangle\langle 0| + I \otimes |1\rangle\langle 1| && (55)\\ &= I \otimes (|0\rangle\langle 0| + |1\rangle\langle 1|) = I_2. && (56) \end{aligned}$$

Then the probability of obtaining a measurement outcome of 0 is

$$\begin{aligned} p(0) &= \langle\psi| M_0^\dagger M_0 |\psi\rangle = \langle\psi| M_0 |\psi\rangle && (57)\\ &= \left(\sum_{i=0}^{3} \alpha_i^* \langle i|\right)(\alpha_0 |00\rangle + \alpha_2 |10\rangle) && (58)\\ &= |\alpha_0|^2 + |\alpha_2|^2. && (59) \end{aligned}$$

Similarly, one obtains $p(1) = |\alpha_1|^2 + |\alpha_3|^2$. And the respective post-measurement states are

$$\frac{\alpha_0 |00\rangle + \alpha_2 |10\rangle}{\sqrt{|\alpha_0|^2 + |\alpha_2|^2}} \quad \text{and} \quad \frac{\alpha_1 |01\rangle + \alpha_3 |11\rangle}{\sqrt{|\alpha_1|^2 + |\alpha_3|^2}}. \tag{60}$$

The symbol representing the measurement of one-qubit in the computational basis is a meter, as shown in Figure 11.

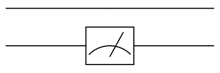


Figure 11: Measurement of the last qubit in the computational basis.

The measurement of one qubit plays an important role in many algorithms. There are situations in which we use a single qubit to store whether the run of a part of an algorithm has been successful or not. Then we measure the qubit and decide what to do depending on the outcome. If it has been what we labelled as success, we tell the program to proceed. Otherwise we repeat the process until we see the flag for success.

3 Literature review

In this section we discuss the existing literature on solving systems of linear equations and related problems. Here we make the same division into subproblems that was encountered during the implementation of the algorithm and which is followed throughout the report.

First we give an overview of the quantum algorithms to solve systems of linear equations. Then we address state preparation, which will be needed to load the right hand side of the equation into the quantum register. Afterwards we discuss the literature on Hamiltonian simulation (finding a circuit whose effect is e^{iAt}) applied to the problem of solving systems of linear equations. The section finishes with a review of the existing works on quantum arithmetics.

3.1 Solving systems of linear equations

The main idea came from the paper by Harrow, Hassidim and Lloyd (HHL) [1], and much of the subsequent literature on the topic uses their algorithm as a base and tries to improve the running time or use of resources. At the core of the algorithm is a procedure called Quantum Phase Estimation, which is a quantum algorithm that estimates the eigenvalues of an operator and stores them in the register. Then, quantum arithmetics can be used to invert these values and obtain the solution to the system. For an accuracy of ϵ, however, the running time of the Quantum Phase Estimation increases as $\mathcal{O}(1/\epsilon)$. The total runtime complexity of the algorithm is $\mathcal{O}(\log(N)s^2\kappa^2/\epsilon)$.

In [8], the authors propose to circumvent the limits from phase estimation by approximating the inverse of the matrix via unitaries. They propose two methods, one more general applying to any Hamiltonian which can be efficiently simulated and the other more efficient but applying only for sparse hamiltonians. The improved algorithm would run in time poly(log(1/epsilon)). Their method was used as a subroutine in a study of the HHL algorithm applied to the finite element method for differential equations[36].

All the papers mentioned focus on sparse matrices, which are defined as matrices with few non-zero entries. However, there are also proposals to deal with dense matrices, such as [9].

The running time of the HHL algorithm scales with the condition number of the matrix, which poses a problem for ill-conditioned matrix. [10] introduces the idea to first apply a preconditioner from the class of sparse approximate inverse (SPAI). The reason for this class rather than an arbitrary preconditioner is that we only have local knowledge of A and sparsity should be preserved after precondition for the hamiltionian simulation. This class of preconditioners was originally introduced in [31] for parallel computing.

Recently, a hybrid algorithm was proposed in [11]. Here the register is measured after phase estimation to obtain the eigenvalues classically and then find a suitable circuit to perform the inversion of these values.

Finally, in the paper by Harrow, Hassidim and Lloyd, there is a proof of the optimality of their algorithm based on complexity classes. It can be found in Section-5 of [1].

3.2 State preparation

Our motivation to work on this problem will be to bring the initial quantum register into a state representing the vector $|b\rangle$ on the right hand side of the equation. Nonetheless, it is a broader problem: many quantum algorithms rely of having a procedure to prepare the initial register in a determinate state. The problem is defined as, starting from $|0\rangle_n$, how to obtain an target state $|\psi\rangle_n$.

One of the first proposals ([7]) was to prepare states whose amplitudes are given by a probability distribution which is efficiently classically computable.The idea is that if something is efficiently computable by a classical computer, one can take the classical circuit and apply it simultaneously to the basis states. The drawback is that there is still much work to do on performing arithmetic operations with quantum computers.

Soklakov and Schack offer in [12] an improvement of this method, where the function does not need to be efficiently integrable. They part from the assumption that there are oracles defining the state (i.e. giving the amplitudes and phases of each basis state) and propose a state preparation algorithm similar to [7] by approximating the state. However, it is a nontrivial assumption that these oracles are given. But it could be applied in conjunction to an algorithm to obtain those oracles.

In [13] they propose a method which assumes positive real amplitudes and encoded classical information about the state. With this information and via controlled rotations they achieve the desired quantum state. Although not being able to efficiently prepare all states, the method will work for a family of states where it is possible to efficiently compute a function of the amplitudes.

On the other hand, [14] proposed dividing the vector representing the state in 2-blocks and performing rotations on each block. Since these are controlled rotations, it requires an exponential number of gates. Conveniently, their method is already integrated in Qiskit ([42]) and can be readily used. Because the goal was to solve a system of linear equations more efficiently than on a classical computer, we tried to be thrifty when it came to gates and sought for an alternative method.

Another method through rotations is proposed in [15]. It is a more general algorithm to transform any state $|a\rangle$ into $|b\rangle$. The idea is to apply a set of rotations to equalise the phases, and then another set of rotations to obtain a fixed state (say, $|0\rangle$). Finally, apply the inverse operations to obtain the target state.

In [16] they propose to split the state into even parts and use the method by [15] to prepare a 2-qubit state. The idea is to copy the just prepared state into the other half of the register via CNOTS. Then unitary operation to simultaneously transform the copied basis states of the second half of the register into Schmidt basis states. For more than 4 qubits, it uses the same idea but relies on being able to prepare a k-qubit state. So this paper rather than a method to prepare an arbitrary state provides a method to decrease the number of gates.

3.3 Hamiltonian simulation

The Hamiltonian simulation problem is defined as, given an $n \times n$ Hermitian matrix A, find a circuit whose effect is e^{iAt}. We will need it later for the Quantum Phase Estimation stage of the HHL algorithm (Section 4.2).

Mostly all approaches make a subdivision into two further problems: decomposition into a sum of sparse Hamiltonians and recombination in the simulation.

[17] propose decomposing the given matrix into a sum of one-sparse matrices (that is, at most one non-zero entry per row and column) and then simulating each of them. However, if two matrices do not commute then the product of exponentials is just an approximation. So one wants to minimise the number of one-sparse matrices used in the decomposition. The key idea to find a good decomposition is by associating the matrix to a graph and finding a colouring, this will be explained in more detail in Section 6.1.

A very detailed description of how to simulate general 1-sparse Hamiltonians can be found in Chapter-4 of [18].

On the other hand, in [19] they part from the assumption that the simulation of 1-sparse hamiltonians is given and instead focus on a more efficient algorithm. They study higher order Suzuki methods and show their scaling is optimal in a blackbox setting (i.e. not exploiting the structure of the problem). They also give a graph-theory based algorithm to decompose the Hamiltonian into a sum of 1-sparse which minimizes the number of black-box calls.

An approach different from Lie-Trotter-Suzuki is given by [20]. They show how to implement a linear combination of unitaries with high probability, and study the approximation via multi-product formulas instead of Lie-TrotterSuzuki, since the former require less exponentials.

As to how to decompose H, [22] say that by taking the components of the sum to be graphs whose connected components are stars translates into an improved complexity of the simulation.

Another method is via quantum walks. For a discrete-quantum walk, a step corresponds to a unitary operation moving amplitudes between adjacent vertices. First [21] and improved in [23] by focusing on applying it to Hamiltonian simulation assuming black-boxes for the Hamiltonian they show how to obtain an operator which corresponds to the quantum walk of the Hamiltonian. One of the advantages of this method is that it applies to non-sparse Hamiltonians, although works better when sparse.

In [24] they study the trade-off between increasing the number of Trotter steps to gain accuracy in the approximation to the exponential, and the physical inaccuracy added from increasing the circuit depth.

3.4 Quantum arithmetics

Many quantum algorithms rely on being able to implement arithmetic functions. A quantum circuit is, however, reversible by definition and therefore so must be the circuits for quantum arithmetics. The price tag of these kind of circuits is a large number of qubits to remember the intermediate results [25]. Therefore, using traditional reversible circuits translates into obtaining very expensive quantum circuits from an inexpensive classical algorithm.

Research on the topic focuses on reducing the auxiliary memory needed or the number of computational steps. [26] focuses on the former and give circuits for addition, multiplication and exponentiation. They save memory by reversing some computations with different computations.

A useful operation is the so called analog (data as amplitudes) to digital (data as qubit strings) conversion and vice versa. This is, respectively,

$$\lambda \mapsto \left|\tilde{\lambda}\right\rangle_m \quad \text{and} \quad \left|\tilde{\lambda}\right\rangle_m \mapsto \lambda, \tag{61}$$

where $\lambda \in \mathbb{R}$ and $\tilde{\lambda}$ is the m-bit string that best approximates λ. The above conversions are described in [27] and are not to be confused with [28], whose paper gives a method to encode a smooth wave function into the amplitudes and phases of a quantum state.

These are very useful techniques which can be widely applied to different parts of quantum algorithms. For example, later we will use digital-to-analog conversion in our state preparation and in the conditioned rotation of the eigenvalues. In a more recent paper [37], they show how to evaluate polynomials and give the implementation of a few useful functions (such as trigonometric). They adapt ideas from high-performance classical libraries to a reversible fixedpoint domain. Their circuits were developed so that they can be added to existing software (e.g. Quipper). The paper also gives resource analyses for each operation. Worth to mention is that there is also investigation on how to implement elliptic curve arithmetic [38, 39, 40]. In [25], the authors propose to compute arithmetic in the amplitudes instead, therefore reducing the number of ancilla qubits that would be otherwise needed. Floating point quantum arithmetics is explored in [41], where they show it is viable. They study different choices of floating point representations in the number of qubits.

Another approach is introduced in [29] and later used in [30]. Arguing that physical properties cannot be measured to high precision, and that it is the reason digital computers replaced analog, they propose a digital representation of a state. The latter means to encode the amplitudes of the state in binary in an additional register. Some advantages of this method are that it allows for controlled arithmetics and that operations that would not be unitary become so.

4 Solving systems of linear equations with a quantum computer

Let $A \in \mathbb{C}^{N\times N}, N \in \mathbb{N}$ be a Hermitian matrix ($A^\dagger = A$) and $|x\rangle, |b\rangle \in \mathbb{C}^N$ be complex vectors. The problem of solving a system of linear equations can be described as

$$A|x\rangle = |b\rangle . \tag{62}$$

Let κ denote the condition number of A and ϵ the accuracy of the calculated solution. For a classical computer, the best general method requires $\mathcal{O}(N^3 s\kappa \log(1/\epsilon))$ running time, although it returns the full solution. The HHL is a quantum algorithm to estimate a function of $|x\rangle$ in time $\mathcal{O}(\log(N)s^2\kappa^2 \log(1/\epsilon))$ (Table 1 from [33]). Where, in both cases, s is the maximum number of nonzero entries in the same row or column of A, and κ denotes its condition number. After running the HHL, the entries of the vector solution to the system, $|x\rangle$, come encoded as the amplitudes of the final state of the register. Therefore one does not have direct access to them rather to a function of it. Nevertheless, in many situations that is what one is interested in.

From now on, assume that $|x\rangle$ and $|b\rangle$ are unit vectors. Since A is Hermitian, it has a spectral decomposition

$$A = \sum_{j=0}^{N-1} \lambda_j |u_j\rangle \langle u_j| , \quad \lambda_j \in \mathbb{R}, \tag{63}$$

where $|u_j\rangle$ is the j^{th} eigenvector of A with respective eigenvalue λ_j. Then,

$$A^{-1} = \sum_{j=0}^{N-1} \lambda_j^{-1} |u_j\rangle \langle u_j| , \tag{64}$$

and the right hand side of (62) can be written in the eigenbasis of A,

$$|b\rangle = \sum_{j=0}^{N-1} b_j |u_j\rangle , \quad b_j \in \mathbb{C}. \tag{65}$$

It is useful to keep in mind that the objective of the HHL is to exit the algorithm with the readout register in the state

$$|x\rangle = A^{-1}|b\rangle = \sum_{j=0}^{N-1} \lambda_j^{-1} b_j |u_j\rangle . \tag{66}$$

Note that here we already have an implicit normalisation constant since we are talking about the state of a register.

4.1 Description of the HHL

Here we give a brief outline the key steps of the algorithm, and explain them in more detail later. The first register, $|0\rangle_{n_l}$ will be used to store a binary representation of the eigenvalues of A. The second register, $|0\rangle_{n_b}$ will contain the vector solution, and from now on, $N = 2^{n_b}$. There is an extra register, for the ancilla qubits. These are qubits used as intermediate steps in the individual computations, such as the decomposition of the $C^n - U$ gate from Section 2.2, but will be ignored since they are set to $|0\rangle$ at the beginning of each computation and restored back to the $|0\rangle$ state at the end of the individual operation.

All registers start in the $|0\rangle$ state at the beginning of an algorithm. In the following description of the algorithm, we assume that all computations have been exact.

(i) Load the data $|b\rangle \in \mathbb{C}^{2^{n_b}}$. That is, perform the transformation

$$|0\rangle_{n_l} \otimes |0\rangle_{n_b} \mapsto |0\rangle_{n_l} \otimes |b\rangle_{n_b} = \sum_{j=0}^{N-1} |0\rangle_{n_l} \otimes b_j \left|u_j\right\rangle_{n_b} =: \sum_{j=0}^{N-1} b_j |0\rangle_{n_l} \left|u_j\right\rangle_{n_b}. \tag{67}$$

(ii) Apply Quantum Phase Estimation (QPE) with

$$U = e^{iAt} = \sum_{j=0}^{N-1} e^{i\lambda_j t} \left|u_j\right\rangle \left\langle u_j\right|, \tag{68}$$

where t will be specified later in Section 9.1. From Section 4.2, the register is now in the state

$$\sum_{j=0}^{N-1} b_j \left|\lambda_j\right\rangle_{n_l} \left|u_j\right\rangle_{n_b}. \tag{69}$$

(iii) Add an ancilla qubit and apply a rotation conditioned on $|\lambda_j\rangle$,

$$\sum_{j=0}^{N-1} b_j \left|\lambda_j\right\rangle_{n_l} \left|u_j\right\rangle_{n_b} \left(\sqrt{1-\frac{1}{\lambda_j^2}}\,|0\rangle + \frac{1}{\lambda_j}|1\rangle\right). \tag{70}$$

(iv) Apply the inverse of QPE. If the eigenvalues are perfectly estimated, this results (from Section 4.2) in

$$\sum_{j=0}^{N-1} b_j |0\rangle_{n_l} \left|u_j\right\rangle_{n_b} \left(\sqrt{1-\frac{1}{\lambda_j^2}}\,|0\rangle + \frac{1}{\lambda_j}|1\rangle\right). \tag{71}$$

(v) Measure the last qubit in the computational basis. If the outcome is $|1\rangle$, the register is in the post-measurement state

$$\left(\sqrt{\frac{1}{\sum_{j=0}^{N-1} \left|b_j\right|^2 / \left|\lambda_j\right|^2}}\right) \sum_{j=0}^{N-1} \frac{b_j}{\lambda_j} |0\rangle_{n_l} \left|u_j\right\rangle_{n_b}, \tag{72}$$

which up to a normalisation factor corresponds to Eq. 66.

4.2 Quantum Phase Estimation (QPE)

Quantum Phase Estimation is a quantum algorithm which, given a unitary U with eigenvector $|\psi\rangle_m$ and eigenvalue $e^{2\pi i\theta}$, finds θ. We can formally define this as follows.

Definition 4.1. Let $U \in \mathbb{C}^{2^m \times 2^m}$ be unitary and let $|\psi\rangle_m \in \mathbb{C}^{2^m}$ be one of its eigenvectors with respective eigenvalue $e^{2\pi i\theta}$. The **Quantum Phase Estimation** algorithm, abbreviated **QPE**, takes as inputs the unitary gate for U and the state $|0\rangle_n |\psi\rangle_m$ and returns the state $\left|\tilde{\theta}\right\rangle_n |\psi\rangle_m$. Here $\tilde{\theta}$ denotes a binary approximation to θ and the n subscript denotes it has been truncated to n digits.

$$\mathrm{QPE}(U, |0\rangle_n |\psi\rangle_m) = \left|\tilde{\theta}\right\rangle_n |\psi\rangle_m . \tag{73}$$

Example 7. For the HHL we will use QPE with $U = e^{iAt}$, where A is the matrix associated to the system we want to solve. In this case, for the eigenvector $|u_j\rangle_{n_b}$, which has eigenvalue $e^{i\lambda_j t}$, it will output $\left|\tilde{\lambda}_j\right\rangle_{n_l} |u_j\rangle_{n_b}$. Where $\tilde{\lambda}_j$ represents an n_l-bit binary approximation to $\frac{\lambda_j t}{2\pi}$.

The algorithm uses the inverse Quantum Fourier Transform (QFT†), which is defined as follows.

Definition 4.2. The **Quantum Fourier Transform** (**QFT**) on $N = 2^n, n \in \mathbb{N}$, qubits is the map

$$\mathrm{QFT}_N : |x\rangle_n \mapsto \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i \frac{x}{N} y} |y\rangle_n , \tag{74}$$

where $|x\rangle_n \in \mathbb{C}^N$ is a basis state (i.e. $x \in \{0, ..., N-1\}$). Its inverse, the inverse Quantum Fourier Transform (QFT†), is given by the map

$$\mathrm{QFT}_N^\dagger : |x\rangle_n \mapsto \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{-2\pi i \frac{x}{N} y} |y\rangle_n , \tag{75}$$

for $|x\rangle_n \in \mathbb{C}^N$ a basis state.

The full circuit for the Quantum Phase Estimation is shown in Figure 12 Here we analyse the action of the effect of the QPE on the quantum state after

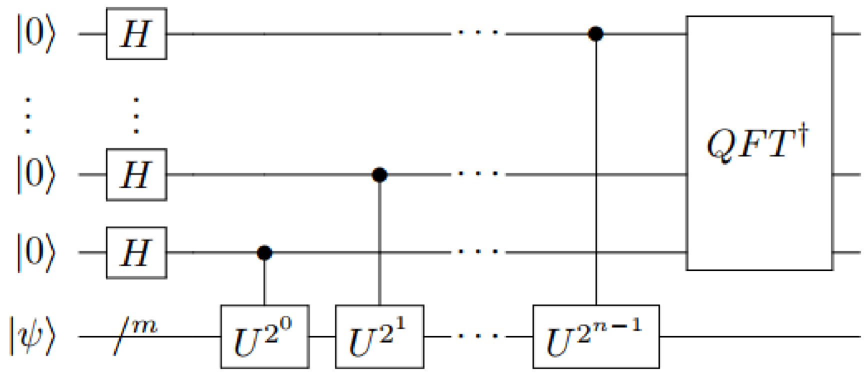


Figure 12: Circuit for the Quantum Phase Estimation

loading the data from Eq. 67. For simplicity, we do it first for one eigenvalue.

(i) $H^{\otimes n_l}$

$$|0\rangle_{n_l} |u_j\rangle_{n_b} \quad \mapsto \quad \frac{1}{2^{n_l/2}} \bigotimes_{l=0}^{n_l-1} (|0\rangle + |1\rangle) |u_j\rangle_{n_b}. \tag{76}$$

(ii) After the controlled powers of U

$$\frac{1}{2^{n_l/2}} \bigotimes_{l=0}^{n_l-1} (|0\rangle + e^{i2^l\lambda_j t} |1\rangle) |u_j\rangle_{n_b} = \frac{1}{2^{n_l/2}} \sum_{l=0}^{2^{n_l}-1} e^{it\lambda_j l} |l\rangle |u_j\rangle_{n_b}. \tag{77}$$

(iii) $\mathrm{QFT}^\dagger$ with $N = 2^{n_l}$

$$\frac{1}{2^{n_l}} \sum_{k,l=0}^{2^{n_l}-1} e^{i(t\lambda_j l - 2\pi lk/2^{n_l})} |k\rangle_{n_l} |u_j\rangle_{n_b}. \tag{78}$$

Writing

$$\alpha_{k|j} = \frac{1}{2^{n_l}} \sum_{l=0}^{2^{n_l}-1} e^{iI(t\lambda_j - 2\pi k/2^{n_l})}, \tag{79}$$

78 simplifies to

$$\sum_{k=0}^{2^{n_l}-1} \alpha_{k|j} |k\rangle_{n_l} |u_j\rangle_{n_b}. \tag{80}$$

If λ_j can be exactly represented by n bits, then $\lambda_j = k/2^{n_l}$ for some $0 \leq k \leq 2^{n_l} = 1$. In this case, $\alpha_{k|j} = 1$ and $\alpha_{k'|j} = 0$ for all $k' \neq k$. And 78 would further simplify, after relabeling $k = k/2^{n_l}$, to

$$|k\rangle_{n_l} |u_j\rangle_{n_b} = |\lambda_j\rangle_{n_l} |u_j\rangle_{n_b}. \tag{81}$$

Therefore, if the eigenvalues can be exactly represented,

$$\mathrm{QPE}\left(e^{iAt}, \sum_{j=0}^{2^{n_l}-1} |0\rangle_{n_l} |u_j\rangle_{n_b}\right) = \sum_{j=0}^{2^{n_l}-1} |\lambda_j\rangle_{n_l} |u_j\rangle_{n_b}. \tag{82}$$

From this analysis one can also see that on the number of qubits used for the eigenvalues register will depend on the accuracy of the eigenvalue approximation, consequently of the whole HHL.

5 Polynomial state preparation

In numerous real life applications, it is required to solve a linear system of equations $A|x\rangle = |b\rangle$ such that $|b\rangle$ is specified by a function. It is a well known result in Mathematics that continuous functions can be approximated by polynomials. The following theorem, stated without proof is Theorem 7.26. from [34].

Theorem 5.1 (Stone-Weierstrass). *If f is a continuous complex function on $[a, b]$, there exists a sequence of polynomials P_n such that*

$$\lim_{n\to\infty} P_n(x) = f(x) \tag{83}$$

uniformly on $[a, b]$. If f is real, the P_n may be taken real.

In this section we focus on how to prepare an arbitrary state with amplitudes given by a polynomial $p : [0, 1] \to [-\pi/2, \pi/2]$ of degree d. That is, for $N = 2^n$,

$$|b\rangle = \sum_{i=0}^{N-1} p\left(\frac{i}{N-1}\right)|i\rangle \quad \text{where} \quad \sum_{i=0}^{N-1} p\left(\frac{i}{N-1}\right)^2 = 1 \quad \text{and} \quad n \in \mathbb{N}. \tag{84}$$

At the moment of writing, state preparation is an open problem, and it is believed that it is not necessarily possible to do it efficiently for the arbitrary case. However, in the description of quantum algorithms, it is often assumed that the required state comes from previous processes or that there is a black box procedure for its preparation. Furthermore, it is also common in the literature to take for granted access to the exact state. Thereby, inaccuracies deriving from using an approximated quantum state are usually overlooked or intentionally omitted in the error analyses of the algorithms.

This section is structured in the following manner. In Section 5.1 we explain the general technique and then give in Section 5.2 the implementation of the state preparation algorithm. In the HHL paper, which is focused on the algorithm itself rather than the oracles for state preparation or hamiltonian simulation, the authors choose to carry out the calculations neglecting any errors from the preparation of the right hand side. Therefore, we use Section 5.3 to complete the error analyses of the HHL by adding the inaccuracies arising from using an approximated right hand side of the equation, as well as to study our method. Section 5.4 is then an analysis of the probability of successfully preparing a state with our proposed circuit. This section ends with a circuit depth analysis of the algorithm.

5.1 General rotations technique

The main technique used in our implementation is based on conditioned rotations and the goal will be to implement a polynomial. We use the rotation around the $\hat{y}$ axis introduced in Definition 2.21:

$$R_y(\theta) := e^{-i\theta Y/2} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}, \quad \text{where} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}. \tag{85}$$

It can also be regarded as the map

$$R_y(2\theta) : |0\rangle \mapsto \cos(\theta)|0\rangle + \sin(\theta)|1\rangle. \tag{86}$$

To achieve the main goal, namely implementing a polynomial, we need to be able to implement sums and products of variables within the amplitudes of a quantum state. For that we will need controlled versions of the $R_y(\theta)$ gate, such as the one showed in Figure 5.1.

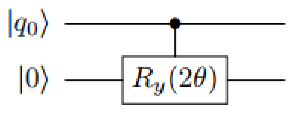


Figure 13: Controlled $R_y(\theta)$.

Proposition 5.1. *A controlled $R_y(\theta)$ gate with control $|q_0\rangle \in \mathbb{C}^2$, $q_0 \in \{0,1\}$, and target $|0\rangle$ is the map*

$$|q_0\rangle |0\rangle \mapsto |q_0\rangle e^{-iq_0\theta Y} |0\rangle . \tag{87}$$

Proof. We calculate the action of the circuit from Figure 5.1 explicitly:

$$|0\rangle |0\rangle \mapsto |0\rangle |0\rangle , \tag{88}$$
$$|1\rangle |0\rangle \mapsto |1\rangle (\cos(\theta) |0\rangle + \sin(\theta) |1\rangle). \tag{89}$$

Which can also be written in one line as

$$|q_0\rangle |0\rangle \mapsto |q_0\rangle (\cos (q_0\theta) |0\rangle + \sin (q_0\theta) |1\rangle) = |q_0\rangle e^{-iq_0\theta Y} |0\rangle . \tag{90}$$

□

To implement products we use multi-controlled $R_y(\theta)$ gates, such as the one from Figure 14.

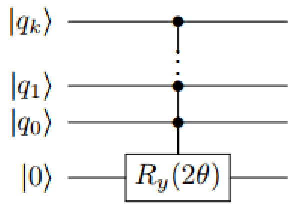


Figure 14: Multi-controlled $R_y(\theta)$ gate.

Proposition 5.2. *A multi-controlled $R_y(\theta)$ gate with controls $|q_k\rangle , ..., |q_0\rangle \in \mathbb{C}^2$ and target $|0\rangle$ is the map*

$$|q\rangle_k |0\rangle \mapsto |q\rangle_k e^{-i\left(\theta \prod_{i=0}^{k} q_i\right)Y} |0\rangle , \tag{91}$$

where $|q\rangle_k := |q_k\rangle \cdots |q_1\rangle |q_0\rangle \in \mathbb{C}^{2^{k+1}}$ and $q_i \in \{0,1\}$, $0 \leq i \leq k$. Here $q \in \{0, ..., 2^{k+1} - 1\}$ has binary representation $q_k...q_1q_0$.

Proof. Similarly as before we calculate explicitly the action of the circuit from Figure 14. Since the gate will be applied only when $q_0 = \cdots = q_k = 1$, we get

$$\begin{aligned} |q\rangle_k |0\rangle &\mapsto |1\cdots 1\rangle_k (\cos(\theta)|0\rangle + \sin(\theta)|1\rangle) && \text{if } q_0 = \cdots = q_k = 1, && (92)\\ |q\rangle_k |0\rangle &\mapsto |q\rangle_k |0\rangle && \text{otherwise .} && (93)\end{aligned}$$

Which can be also written again as

$$\begin{aligned} |q\rangle_k |0\rangle &\mapsto |q\rangle_k \left(\cos\left(\theta \prod_{i=0}^{k} q_i\right)|0\rangle + \sin\left(\theta\prod_{i=0}^{k} q_i\right)|1\rangle\right) && (94)\\ &= |q\rangle_k e^{-i\left(\theta\Pi_{i=0}^{k} q_i\right)Y} |q\rangle_k . && (95)\end{aligned}$$

□

The exponential notation is useful to understand the effect of successive rotations. Addition can be implemented via consecutive controlled $R_y(\theta)$ gates, as shown in calculating the effect from the circuit in Figure 15:

$$\begin{aligned} |q_1\rangle|q_0\rangle|0\rangle &\mapsto |q_1\rangle|q_0\rangle e^{-i\theta_2 q_0 q_1 Y} e^{-i\theta_1 q_0 Y}|0\rangle && (96)\\ &= |q_1\rangle|q_0\rangle e^{-i(\theta_2 q_0 q_1 + \theta_1 q_0)Y}|0\rangle && (97)\\ &= |q_1\rangle|q_0\rangle(\cos(\cdots)|0\rangle + \sin(\theta_2 q_0 q_1 + \theta_1 q_0)|1\rangle), && (98)\end{aligned}$$

where $q_0, q_1 \in \{0,1\}$.

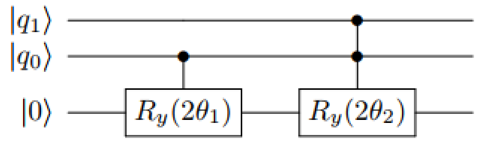


Figure 15: Two controlled $R_y(\theta)$ gates.

Recall that adding Hadamards at the beginning of a circuit allows to calculate simultaneously an operation on all basis states. Thus, with the circuit represented in Figure 16 we can achieve the operation

$$\begin{aligned} |0\rangle|0\rangle|0\rangle &\mapsto \frac{1}{2}\sum_{q=0}^{3}|q\rangle_2(\cos(\ldots)|0\rangle + \sin(\theta_2 q_0 q_1 + \theta_1 q_0)|1\rangle) && (99)\\ &= \frac{1}{2}\sum_{q=0}^{3}|q\rangle_2(\cos(p(q))|0\rangle + \sin(p(q))|1\rangle), && (100)\end{aligned}$$

where $p(q) = \theta_2 q_0 q_1 + \theta_1 q_0$ is a polynomial, $q_0, q_1 \in \{0,1\}$ and $q_1 q_0$ is the binary representation of q.

5.2 Implementation

Let $n_b \in \mathbb{N}$ denote the size of the register which will contain the target state, and $N = 2^{n_b}$. Here we only consider the quantum register on n_b qubits that will contain $|b\rangle$ and an ancilla qubit for the rotation appended at the end. The outline of the algorithm is as follows:

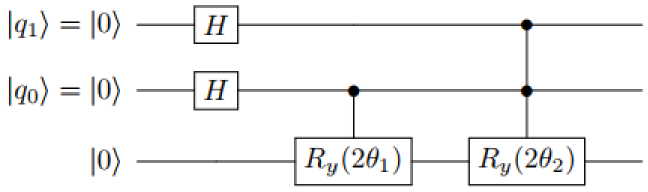


Figure 16: Two Hadamards before controlled $R_y(\theta)$ gates.

(i) Initialise the register in the state $|0\rangle_{n_b} |0\rangle$.

(ii) Apply Hadamards to the first n_b qubits, i.e. the operator $H^{\otimes n_b} \otimes I$. This gives

$$|0\rangle_{n_b} |0\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle_{n_b} |0\rangle . \tag{101}$$

(iii) Apply a rotation conditioned on the $|i\rangle$'s. Ideally, the state is now

$$\frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle_{n_b} \left(\sqrt{1-p(i)^2}\,|0\rangle + p(i)\,|1\rangle\right) . \tag{102}$$

(iv) Measure the appended qubit in the computational basis. An outcome of 1 means we have the post-measurement state

$$|b\rangle_{n_b} = \sum_{i=0}^{N-1} p(i)\,|i\rangle_{n_b} , \tag{103}$$

after dropping the ancilla qubit since after the measurement we know it is $|1\rangle$.

In reality, the state we can efficiently achieve in Step (iii) due to the nature of the quantum rotation gates, is

$$\frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle_{n_b} (\cos(p(i))\,|0\rangle + \sin(p(i))\,|1\rangle). \tag{104}$$

Exploiting the near linearity of the sine function when the argument is close to 0, we prepare instead $c\,|b\rangle$. Then for $c \in \mathbb{R}$ small enough we have that

$$\sum_{i=0}^{N-1} \sin(cp(i))\,|i\rangle_{n_b} \approx \sum_{i=0}^{N-1} cp(i)\,|i\rangle_{n_b} = c\,|b\rangle\,n_b. \tag{105}$$

We first show how to do it specifically for the base case: a two-qubit state $|b\rangle$ and a polynomial of degree 2. And then indicate how to generalise the implementation for an arbitrary number of qubits and arbitrary degree of the polynomial.

Thus, we look for the circuit for $p(x) = a_2x^2 + a_1x + a_0$, where $x \in \{0, 1, 2, 3\}$ is represented by two qubits and $a_i \in \mathbb{C}$, $0 \leq i \leq 2$. In this case we have for $q_i \in \{0, 1\}$,

$$x = 2q_1 + q_0 \qquad \text{and} \qquad q_i^2 = q_i.$$

So

$$p(x) = p(2q_1 + q_0) = (4a_2 + 2a_1)q_1 + (a_2 + a_1)q_0 + 4a_2q_1q_0 + a_0. \tag{106}$$

The rotation from Eq. 104 can then be achieved with the circuit from Figure 17.

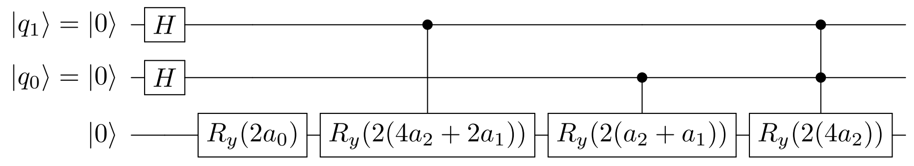


Figure 17: Circuit preparing a state with amplitudes given by the polynomial from Eq. 106.

For the general case, let $|b\rangle_{n_b}$ be an n_b-qubit state and $p(x) = \sum_{i=0}^{d} a_i x^i$, a polynomial of degree d. In this context, $x \in \{0, 1, \ldots, 2^{n_b} - 1\}$ and for $q_j \in \{0, 1\}$,

$$x = \sum_{j=0}^{n_b-1} 2^j q_j \quad \text{and} \quad q_j^2 = q_j. \tag{107}$$

Substituting into $p(x)$,

$$p\left(\sum_{j=0}^{n_b-1} 2^j q_j\right) = \sum_{i=0}^{d} a_i \left(\sum_{j=0}^{n_b-1} 2^j q_j\right)^i \tag{108}$$

$$= \sum_{i=0}^{d} a_i \sum_{k_0+\ldots+k_{n_b-1}=i} \binom{i}{k_0, \cdots, k_{n_b-1}} \prod_{j=0}^{n_b-1} 2^{jk_j} q_j. \tag{109}$$

The circuit is similar to that for the base case. Each product of q_j's determines a controlled rotation, with controls the $|q_j\rangle$ appearing in that product and argument determined by its coefficient in Eq. 109.

To summarise, the key steps of the Polynomial State Preparation algorithm (PSP) are:

(i) Hadamards to obtain a superposition of basis states.

(ii) Controlled R_y rotations to implement $p(x)$.

(iii) Measurement of the ancilla qubit. Repeat steps (i) and (ii) until observing $|1\rangle$. Then, the post-measurement state of the register on outcome 1 is $|b\rangle$.

We will now proceed to analyse the additional error induced in the overall performance of the HHL from using an approximated initial state, and the number of gates needed to implement the circuit.

5.3 Error analysis

Let $\|\cdot\|$ denote the 2-norm for the remaining of this section. The following is Theorem 1 from [1], and it assumes efficient and accurate methods for the state preparation and Hamiltonian simulation. We will use $|x\rangle$ to denote the exact solution to the system of linear equations and $|\tilde{x}\rangle$ to denote the quantum state returned by the HHL representing an approximation to the solution $|x\rangle$.

Theorem 5.2 (HHL). *Take $t = O(\kappa/\epsilon_H)$, and let $|x\rangle$ and $|\tilde{x}\rangle$ as defined above. Then*

$$\||x\rangle - |\tilde{x}\rangle\| < \epsilon_H. \tag{110}$$

Definition 5.1. We will denote by $\mathrm{HHL}(A, |b\rangle) := |\tilde{x}\rangle$ the normalised state returned by the HHL algorithm assuming exact procedures for preparing $|b\rangle$ and simulating e^{iAt}. And ϵ_H as defined in Theorem 5.2, will be the accuracy of the algorithm with these assumptions. Thus,

$$\||x\rangle - \mathrm{HHL}(A, |b\rangle)\| < \epsilon_H, \tag{111}$$

where $|x\rangle$ denotes the exact answer.

Lemma 5.3. *Let $\left|\tilde{b}\right\rangle$ denote the approximated state from the Polynomial State Preparation algorithm, c the small parameter used in the procedure and κ the condition number of A. Then*

$$\left\||x\rangle - \mathrm{HHL}\left(A, \left|\tilde{b}\right\rangle\right)\right\| = \mathcal{O}\left(\kappa c^2 + \epsilon_H\right). \tag{112}$$

Corollary 5.3.1. *For an overall accuracy of ϵ, we can set the parameters in the algorithm so that $\epsilon_H < \epsilon/2$ and take*

$$c < \sqrt{\frac{\epsilon}{2\kappa}}. \tag{113}$$

For the proof of Lemma 5.3 we will need the following result.

Proposition 5.3. *Let $|v\rangle \in \mathbb{C}^n$ be unitary and $c \in \mathbb{R}$ be a small, positive real number. Then*

$$\sum_{i=0}^{n-1} \sin^2(cv_i) = c^2 + \mathcal{O}(c^4) \quad \text{for } c \to 0. \tag{114}$$

Proof. The Taylor Series expansion for the sine function is given by

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots. \tag{115}$$

Thus, for $c \to 0$,

$$\sin^2(cv_i) = c^2 v_i^2 + \mathcal{O}\left(c^4\right). \tag{116}$$

Hence

$$\sum_{i=0}^{n-1} \sin^2(cv_i) = \sum_{i=0}^{n-1} c^2 v_i^2 + \mathcal{O}\left(c^4\right) \quad \text{for } c \to 0. \tag{117}$$

□

We now prove Lemma 5.3.

Proof. Let $p : [0,1] \to [-\pi/2, \pi/2]$ be a polynomial, let $n_b \in \mathbb{N}$, $N := 2^{n_b}$ and let

$$|b_0\rangle = \frac{1}{N}\sum_{i=0}^{N-1} p\left(\frac{i}{N-1}\right)|i\rangle . \tag{118}$$

Finally, since quantum states are normalised, let $|b\rangle := |b_0\rangle / \||b_0\rangle\|$ be the state we want to prepare. Instead, what we obtain is

$$\left|\tilde{b}\right\rangle = \frac{\sum_i \sin(cb_i)|i\rangle}{\sqrt{\sum_i \sin^2(cb_i)}} = \frac{\sum_i cb_i|i\rangle + \sum_i \mathcal{O}(c^3 b_i^3)|i\rangle}{\sqrt{\sum_i \sin^2(cb_i)}}, \tag{119}$$

where b_i denotes the i^{th} component of the vector $|b\rangle$ and $c \in \mathbb{R}$ is small. After simplifying using Proposition 5.3, this expression can be rewritten as

$$\left|\tilde{b}\right\rangle = |b\rangle + |\epsilon\rangle, \quad \text{where} \quad \||\epsilon\rangle\| = c^2 + \mathcal{O}\left(c^4\right) \quad \text{for } c \to 0. \tag{120}$$

The overall error taking into account the initial state approximation is given by the quantity

$$\left\||x\rangle - \mathrm{HHL}\left(A, \left|\tilde{b}\right\rangle\right)\right\| = \left\|\frac{A^{-1}|b\rangle}{\|A^{-1}|b\rangle\|} - \left(\frac{A^{-1}\left|\tilde{b}\right\rangle}{\left\|A^{-1}\left|\tilde{b}\right\rangle\right\|} + \mathcal{O}(\epsilon_H)\right)\right\| \tag{121}$$

$$\leq \left\|\frac{A^{-1}|b\rangle}{\|A^{-1}|b\rangle\|} - \frac{A^{-1}\left|\tilde{b}\right\rangle}{\left\|A^{-1}\left|\tilde{b}\right\rangle\right\|}\right\| + \mathcal{O}(\epsilon_H), \tag{122}$$

where we used the triangle inequality to derive the second line.

Expanding the first term of the right hand side of Eq. 122 gives

$$\begin{aligned}
\left\|\frac{A^{-1}|b\rangle}{\|A^{-1}|b\rangle\|} - \frac{A^{-1}\left|\tilde{b}\right\rangle}{\left\|A^{-1}\left|\tilde{b}\right\rangle\right\|}\right\| &= \left\|\frac{A^{-1}|b\rangle}{\|A^{-1}|b\rangle\|} - \frac{A^{-1}|b\rangle}{\left\|A^{-1}\left|\tilde{b}\right\rangle\right\|} - \frac{A^{-1}|\epsilon\rangle}{\left\|A^{-1}\left|\tilde{b}\right\rangle\right\|}\right\| \\
&\leq \left\|\frac{A^{-1}|b\rangle\left(\left\|A^{-1}\left|\tilde{b}\right\rangle\right\| - \|A^{-1}|b\rangle\|\right)}{\|A^{-1}|b\rangle\|\left\|A^{-1}\left|\tilde{b}\right\rangle\right\|}\right\| + \frac{\|A^{-1}|\epsilon\rangle\|}{\left\|A^{-1}\left|\tilde{b}\right\rangle\right\|} \\
&\leq \frac{\left|\left\|A^{-1}\left|\tilde{b}\right\rangle\right\| - \|A^{-1}c|b\rangle\|\right|}{\left\|A^{-1}\left|\tilde{b}\right\rangle\right\|} + \frac{\|A^{-1}|\epsilon\rangle\|}{\left\|A^{-1}\left|\tilde{b}\right\rangle\right\|} \\
&\leq \frac{\left|\|A^{-1}|b\rangle\| - \|A^{-1}|\epsilon\rangle\| - \|A^{-1}|b\rangle\|\right|}{\left\|A^{-1}\left|\tilde{b}\right\rangle\right\|} + \frac{\|A^{-1}|\epsilon\rangle\|}{\left\|A^{-1}\left|\tilde{b}\right\rangle\right\|} \\
&= 2\frac{\|A^{-1}|\epsilon\rangle\|}{\left\|A^{-1}\left|\tilde{b}\right\rangle\right\|} \\
&\leq 2\kappa\left[c^2 + \mathcal{O}(c^4)\right].
\end{aligned} \tag{123}$$

The last inequality comes from the definition of the condition number written as the ratio of the relative error in the solution to the relative error in $\left|\tilde{b}\right\rangle$. Namely,

$$\kappa = \max \frac{\frac{\left\| A^{-1} |\epsilon\rangle \right\|}{\left\| A^{-1} |\tilde{b}\rangle \right\|}}{\frac{\left\| |\epsilon\rangle \right\|}{\left\| |\tilde{b}\rangle \right\|}}.$$

I.e.,

$$\frac{\left\| A^{-1} |\epsilon\rangle \right\|}{\left\| A^{-1} \left|\tilde{b}\right\rangle \right\|} \leq \kappa \frac{\left\| |\epsilon\rangle \right\|}{\left\| \left|\tilde{b}\right\rangle \right\|} = \kappa \frac{\left\| |\epsilon\rangle \right\|}{\left\| |b\rangle + |\epsilon\rangle \right\|} \leq \kappa \frac{\left\| |\epsilon\rangle \right\|}{1} = \kappa \left[c^2 + \mathcal{O}(c^4) \right]$$

using 120 and that $|b\rangle$ is normalised.

Combining 122 and 123 gives the bound

$$\left\| |x\rangle - \mathrm{HHL}\left(A, \left|\tilde{b}\right\rangle\right) \right\| = \mathcal{O}\left(\kappa c^2 + \epsilon_H\right). \tag{124}$$

□

5.4 Success probability analysis

In Step (iv) we saw that having prepared the right initial state depends upon obtaining a 1 after measuring the last qubit in 104. We proceed now to study the relation between the probability of preparing the desired state and the choice of c. Thus, suppose we are in the state

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle_{n_b} \left(\cos(cp(i/N-1)) |0\rangle + (\sin(cp(i/N-1)) |1\rangle\right). \tag{125}$$

The following definition is Definition 3.6. from [32].

Definition 5.2. Let $f : [0,1] \to [a,b]$ where $a, b \in \mathbb{R}$. The L^2-norm of f is defined as

$$\|f\|_{L^2_{[0,1]}} = \left(\int_0^1 |f(x)|^2 dx \right)^{1/2}. \tag{126}$$

Lemma 5.4. *Let $\mathbb{P}[|1\rangle]$ denote the probability of successfully preparing the state and $p : [0,1] \to [-\pi/2, \pi/2]$ the polynomial from the state preparation. Then*

$$\mathbb{P}[|1\rangle] = \mathcal{O}\left(c^2 \|p\|^2_{L^2_{[0,1]}}\right). \tag{127}$$

Set c as in Corollary 5.3.1. Then the number of expected repetitions of the PSP algorithm behaves as $\mathcal{O}(\kappa/\epsilon)$ for $\epsilon \to 0$.

Proof. With the notation from Section 2.2, we take the collection of measurement operators

$$M_0 = I_{n_b} \otimes |0\rangle\langle 0| \quad \text{and} \quad M_1 = I_{n_b} \otimes |1\rangle\langle 1|. \tag{128}$$

With $|\psi\rangle$ as in Eq. 125,

$$M_1 |\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle_{n_b} (\sin(cp(i/N-1)) |1\rangle). \tag{129}$$

Therefore, the probability of outcome 1 on measuring the last qubit is given by

$$\begin{aligned}\mathbb{P}[|1\rangle] = \langle\psi| M_1^\dagger M_1 |\psi\rangle &= \frac{1}{N}\sum_{i=0}^{N-1} \sin^2(cp(i/N-1)) \quad (130)\\ &= \frac{c^2}{N}\sum_{i=0}^{N-1} p^2\left(\frac{i}{N-1}\right) + \mathcal{O}\left(c^4\right) = \mathcal{O}\left(c^2 \|p\|^2_{L^2_{[0,1]}}\right). \quad (131)\end{aligned}$$

Here we used that

$$\lim_{N\to\infty} \frac{1}{N}\sum_{i=0}^{N-1} p^2\left(\frac{i}{N}\right) = \int_0^1 p^2(x)dx = \|p\|^2_{L^2_{[0,1]}}. \tag{132}$$

Therefore, the number of expected repetitions of the state preparation algorithm before observing $|1\rangle$ behaves as $\mathcal{O}(1/c^2)$ for $c \to 0$. Which, with $c = \mathcal{O}(\sqrt{\epsilon/\kappa})$ behaves like $\mathcal{O}(\kappa/\epsilon)$ for $\epsilon \to 0$. □

5.5 Gate analysis

The complexity of a quantum algorithm is usually quantified in the number of gates it requires. Let n_b and d denote, respectively, the size of the register and the degree of the polynomial as before, and let $N = 2^{n_b}$.

We will assume $d < n_b$. This is a reasonable assumption because we are interested in the asymptotic growth of the number of gates in the circuit as n_b increases. Since we keep the domain of the polynomial fixed, and only increase the number of discretisation points as the size of the system grows, the degree of the polynomial needed to approximate the function specifying $|b\rangle$ depends on the accuracy desired and not on n_b.

Table 1 shows the asymptotic number of CNOT gates, general single-qubit gates and ancilla qubits needed in terms of N.

CNOTs	one-qubit gates	ancilla
$\mathcal{O}(polylog(N))$	$\mathcal{O}(polylog(N))$	$d-1$

Table 1: Gate count for the Polynomial State Preparation algorithm.

The controlling sequences of qubits for the R_y gates are given by the monomials in the expansions of each of $(q_0 + \ldots + q_{n_b-1})^k$, $0 \leq k \leq d$. Therefore, the total number of controlled R_y, where the number of control bits ranges from 0 to d, is

$$\sum_{k=0}^{d} \binom{n_b}{k}. \tag{133}$$

In Section 2.2 we saw that a k-controlled R_y gate can be decomposed into $(k-1)$ ancilla qubits, $(20k-18)$ one-qubit gates and $(12k-10)$ CNOTs. Since the maximum number of controls is d, the circuit will require $d-1$ extra ancilla qubits.

Proposition 5.4. *The number of CNOTs and one-qubit gates needed grows as $\mathcal{O}(polylog(N))$, where the power in "polylog"depends on d.*

Proof. We first proceed to calculate the number of one-qubit gates. We have to count the ones arising from the controlled rotations and an extra non-controlled R_y gate. Therefore, this quantity is given by

$$1+\sum_{k=1}^{d}\binom{n_b}{k}(20(k-1)+2) \leq 1+\sum_{k=1}^{d}\frac{n_b^k}{k!}20k \leq 20\sum_{k=0}^{d}\frac{\log(N)^k}{(k-1)!} \tag{134}$$

$$= \mathcal{O}(polylog(N)). \tag{135}$$

Similarly, the number of CNOTs is given by

$$\sum_{k=1}^{d}\binom{n_b}{k}(12(k-1)+2) = \mathcal{O}(polylog(N)). \tag{136}$$

□

Finally, in from Lemma 5.4, we know that the expected number of repetitions of the circuit is $\mathcal{O}(\kappa/\epsilon)$, where ϵ is the desired accuracy for the complete HHL algorithm. Therefore, Table 2, gives the expected total gate count.

CNOTs	one-qubit gates	ancilla
$\mathcal{O}(polylog(N)\kappa/\epsilon)$	$\mathcal{O}(polylog(N)\kappa/\epsilon)$	$d-1$

Table 2: Expected total gate count for the Polynomial State Preparation algorithm.

6 Hamiltonian simulation of tri-diagonal symmetric matrices

Let $A \in \mathbb{R}^{2^{n_b} \times 2^{n_b}}$, $n_b \in \mathbb{N}$, be a tri-diagonal symmetric matrix. That is, of the form

$$A = \begin{pmatrix} a & b & 0 & 0 \\ b & a & b & 0 \\ 0 & b & a & b \\ 0 & 0 & b & a \end{pmatrix}, \quad a, b \in \mathbb{R}. \tag{137}$$

In this section we will show how to simulate e^{iAt}, $t \in \mathbb{R}$, when A has this structure. These type of matrices are well studied. Their eigenvalues are given by

$$\lambda_j = a - 2b \cos\left(\frac{j\pi}{2^{n_b}+1}\right), \tag{138}$$

with respective eigenvectors

$$v_j = \left(\sin\left(\frac{1j\pi}{2^{n_b}+1}\right), \ldots, \sin\left(\frac{2^{n_b}j\pi}{2^{n_b}+1}\right)\right), \tag{139}$$

for $1 \leq j \leq 2^{n_b}$.

We use the method presented in [17] for the simulation of sparse matrices already introduced in the Literature Review section. It consists of three steps, namely:

(i) Find a decomposition $A = \sum H_i$, where each H_i is a 1-sparse matrix.

(ii) Find an efficient implementation for each e^{iH_i}.

(iii) Use a "Lie-Trotter" approximation to e^{iAt}.

6.1 Implementation

Decomposition into sum of 1-sparse matrices

For arbitrary matrices A, B, the equality $e^{A+B} = e^A e^B$ holds if and only if A and B commute. Since multiples of the identity operator commute with any other matrix, we can take the first term in our decomposition to be

$$H_1 = \begin{pmatrix} a & 0 & & 0 \\ 0 & \ddots & \ddots & \\ & \ddots & & 0 \\ 0 & & 0 & a \end{pmatrix}. \tag{140}$$

And focus for the remainder of this subsection in $A - H_1$. Even if this is a simple case, we can use it to illustrate the general method. Since we will have to simulate individually each element of the decomposition, and each noncommuting term will add an error, we look for a decomposition keeping minimal the number of terms while still being able to efficiently implement each $e^{iH_i t}$.

Before explaining how to find such decomposition in general, the following definitions will be useful.

Definition 6.1. A **graph** G is a pair $G = (V, E)$ where V is a set of **vertices** and E is a (multi)set of unordered pairs of vertices. The elements of E are called **edges**. We write $V(G)$ for the set of vertices and $E(G)$ for the set of edges of a graph.

Definition 6.2. Let $G = (V, E)$ be a graph with $V = \{v_1, \dots, v_n\}$. The adjacency matrix $A = A(G)$ is the $n \times n$ symmetric matrix defined by

$$a_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{141}$$

In general, it is possible to view an $n \times n$ matrix A as the adjacency matrix of a graph by taking as vertex set $\{1, \dots, n\}$ and drawing an edge between i and j whenever $(A)_{ij}$ is nonzero.

Definition 6.3. A **k-edge-colouring** of G is a labelling $f : E(G) \to \{1, \dots, k\}$; the labels are "colours". A **proper k-edge-colouring** is a k-edge-colouring such that edges sharing a vertex receive different colours.

Remark. Given a k-edge-colouring for G, if we colour the nonzero a_{ij} entries of $A(G)$ with the colour of (v_i, v_j), then no colour can appear more than once in any row nor column.

This would give a decomposition

$$A(G) = \sum_c A_c(G), \tag{142}$$

where $A_c(G)$ denotes the matrix consisting of the entries of $A(G)$ which were coloured in c and all other entries set to 0. Furthermore, by the remark, each $A_c(G)$ is 1-sparse.

Going back to our $A - H_1$, the idea now is to view it as the adjacency matrix of a graph and find a proper edge-colouring for it. In this particular case, the corresponding graph and two-edge-colouring are given in Figure 18.

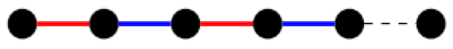


Figure 18: A two-edge-colouring for the graph $A - H_1$.

The H_i then correspond to the adjacency matrices of the subgraphs specified by each colour with weight b. That is,

$$H_2 = \begin{pmatrix} 0 & b & & & & \\ b & 0 & & & & \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & 0 & b \\ & & & & b & 0 \end{pmatrix} \quad \text{and} \quad H_3 = \begin{pmatrix} 0 & & & & & & \\ & 0 & b & & & & \\ & b & 0 & & & & \\ & & & \ddots & & & \\ & & & & 0 & b & \\ & & & & b & 0 & \\ & & & & & & 0 \end{pmatrix}.$$

An advantage of this method is that the decomposition can be found algorithmically, and by Vizing's theorem [35, Theorem 5.3.2.], the number of summands is $\leq s+1$. Where s denotes the sparsity of A. More precisely the theorem states that for a simple graph of maximum degree d a $d+1$ colouring can be efficiently found, and viewing A as an adjacency graph gives $s \leq d$.

Simulation of $e^{iH_i t}$

Here we show efficient implementations for the simulation of each term in the sum, and give their corresponding circuit. We will start with the case $A \in \mathbb{C}^{4\times 4}$ and then extend it to higher dimensions. Thus, we have

$$H_1 = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & a & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & a \end{pmatrix}, \quad H_2 = \begin{pmatrix} 0 & b & 0 & 0 \\ b & 0 & 0 & 0 \\ 0 & 0 & 0 & b \\ 0 & 0 & b & 0 \end{pmatrix}, \quad H_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & b & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Let I_k denote the $2^k \times 2^k$-identity matrix. We will use the following quantum gates:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad U_1(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix} \quad \text{and} \quad R_x(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}.$$

Computing $e^{iH_i t}$ in each case yields the following circuits.

(1)

$$e^{iH_1 t} = \begin{pmatrix} e^{iat} & 0 & 0 & 0 \\ 0 & e^{iat} & 0 & 0 \\ 0 & 0 & e^{iat} & 0 \\ 0 & 0 & 0 & e^{iat} \end{pmatrix} = I_1 \otimes \begin{pmatrix} e^{iat} & 0 \\ 0 & e^{iat} \end{pmatrix} \tag{143}$$

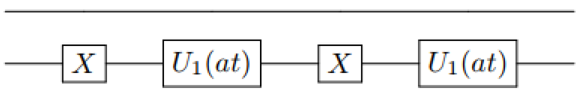


Figure 19: Circuit for implementing $e^{iH_1 t}$.

(2)

$$e^{iH_2 t} = \begin{pmatrix} \cos bt & i\sin bt & 0 & 0 \\ i\sin bt & \cos bt & 0 & 0 \\ 0 & 0 & \cos bt & i\sin bt \\ 0 & 0 & i\sin bt & \cos bt \end{pmatrix} = I_1 \otimes R_x(-2bt) \tag{144}$$

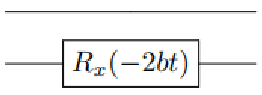


Figure 20: Circuit for implementing $e^{iH_2 t}$.

(3)

$$e^{iH_3t} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos bt & i\sin bt & 0 \\ 0 & i\sin bt & \cos bt & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{145}$$

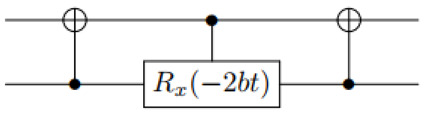


Figure 21: Circuit for implementing e^{iH_3t}.

For a general $n \in \mathbb{N}$ and $A \in \mathbb{C}^{2^{n_b} \times 2^{n_b}}$, the circuits for H_1 and H_2 do not change since they correspond, respectively, to

$$I_{2^{n_b-1}} \otimes \begin{pmatrix} e^{iat} & 0 \\ 0 & e^{iat} \end{pmatrix} \quad \text{and} \quad I_{2^{n_b-1}} \otimes R_x(-2bt). \tag{146}$$

However, the general case for H_3 will make use of a number of CNOTs polynomial in n_b. The full circuit for it is given in Figure 22

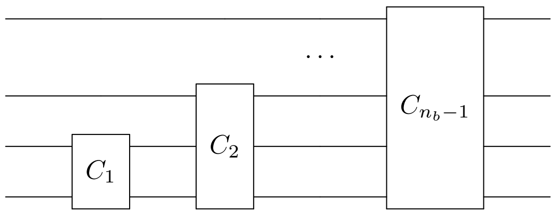


Figure 22: Schematic representation of the circuit implementing e^{iH_3t} for the general dimension case.

The circuit for each C_j-block, $1 \le j \le n_b - 1$, is shown in Figure 23.

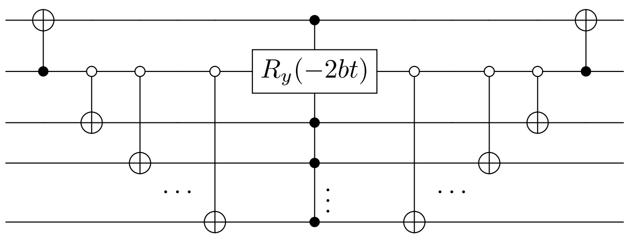


Figure 23: Detailed circuit for C_j.

Approximation of e^{iAt}

Let A and B be Hermitian operators, and t be real, then the Trotter formula [3, Theorem 4.3.] states that

$$e^{i(A+B)t} = \lim_{n\to\infty} \left(e^{iAt/n} e^{iBt/n}\right)^n. \tag{147}$$

Higher order approximations can be derived from this formula for simulating $e^{i(A+B)t}$. We will use the following result [18, Lemma 4.8].

Lemma 6.1 (Trotter formula.)**.** *Let $H = A + B$ with H, A, B Hermitian and $\|A\|, \|B\|, \|H\| \leq \Delta$. Let $m \in \mathbb{N}$ and $t \in \mathbb{R}$ with $m > t$. Then*

$$e^{iHt/m} = e^{iAt/2m} e^{iBt/m} e^{iAt/2m} + \mathcal{O}(\Delta t/m)^3. \tag{148}$$

This result combined with Eq. 147, and with the notation from the previous subsections, gives

$$e^{iAt} = \lim_{m\to\infty} e^{iH_1 t} \left(e^{iH_2 t/2m} e^{iH_3 t/m} e^{iH_2 t/2m}\right)^m. \tag{149}$$

In the next section we will analyse the relation between the error tolerance and the choice of m. One last thing to note is that for the QPE we will need controlled applications of all the circuits mentioned in this section. Having that in mind, to decrease the number of gates we can instead implement

$$e^{iH_1 t} e^{-iH_2 t/2m} \left(e^{iH_2 t/m} e^{iH_3 t/m}\right)^m e^{iH_2 t/2m}. \tag{150}$$

Mathematically, 149 and 150 are the same, but the latter requires fewer gates. The reason is that when drawing $e^{iH_2 t/2m} e^{iH_3 t/m} e^{iH_2 t/2m}$ as a circuit, we have to build the gates from Figure 20, then Figure 21, and again Figure 20. Then to obtain the circuit for Eq. 149, we have to repeat this construction m times. The construction for simulating Eq. 150 is similar, but this time the initial composition consists only on the circuit from Figure 21 and then Figure 20.

6.2 Error analysis

In this subsection we will analyse the approximation we use for the Hamiltonian simulation to give a formula for the number of Trotter steps. Later on, we will see that the main source of error for the overall algorithm will be the inversion of eigenvalues and not the Hamiltonian simulation.

With the notation from Section 6.1,let

$$\tilde{V} := e^{iH_1 t} \left(e^{iH_2 t/2m} e^{iH_3 t/m} e^{iH_2 t/2m}\right)^m \tag{151}$$

denote the approximation to the Hamiltonian simulation, and let $V := e^{iAt}$. Let $\|\cdot\|$ denote the 2-norm as before, and let a, b be the coefficients of the matrix A as in the previous subsections. Let $\mathrm{HHL}\left(\tilde{A}, |b\rangle\right)$ denote the final state of the algorithm assuming an exact procedure for preparing $|b\rangle$ and simulating $\tilde{V}$. Then the following results hold.

Lemma 6.2. *With the notation above,*

$$\left\|V - \tilde{V}\right\| = \mathcal{O}\left(\frac{t^3 b^3}{2m^2}\right) \quad \textit{for } t/m \to 0. \tag{152}$$

Corollary 6.2.1. *Let ϵ_A denote the target tolerance for the Hamiltonian simulation procedure, then we need to choose*

$$m > \sqrt{\frac{t^3 b^3}{2\epsilon_A}}. \tag{153}$$

Lemma 6.3. *Let ϵ_A as before and let ϵ_H denote the error from the HHL assuming exact oracles for state preparation and Hamiltonian simulation. Then*

$$\left\||x\rangle - HHL\left(\tilde{A}, |b\rangle\right)\right\| = \mathcal{O}\left(\epsilon_A + \epsilon_H\right) \quad \textit{for } \epsilon_A, \epsilon_H \to 0, \tag{154}$$

where $|x\rangle$ denotes the exact solution.

We will prove first Lemma 6.2.

Proof. Let

$$U(t) := e^{i(H_2+H_3)t} \quad \text{and} \quad \tilde{U}(t) := e^{iH_2 t/2} e^{iH_3 t} e^{iH_2 t/2}. \tag{155}$$

We want to bound the quantity

$$\left\|V - e^{iH_1 t}\tilde{U}^m(t/m)\right\|. \tag{156}$$

From the Taylor series expansion one finds that

$$\tilde{U}(t/m) = e^{i(H_2+H_3)t/m} + E^{(3)}_{t/m}, \tag{157}$$

where

$$E^{(3)}_{t/m} := \frac{1}{6}\left[\left[H_2, H_3\right], \frac{1}{4}H_2 + \frac{1}{2}H_3\right]\left(\frac{it}{m}\right)^3 + \mathcal{O}\left(\frac{t}{m}\right)^4 \quad \text{for } t/m \to 0. \tag{158}$$

Here $[H_2, H_3] = H_2H_3 - H_3H_2$ denotes the matrix commutator. Both $U(t)$ and $\tilde{U}(t)$ are unitary operators, hence of norm 1. Since $U(t) = U(t/m)^m$, and using the Cauchy-Schwarz and triangle inequalities gives

$$\left\|U(t) - \tilde{U}(t/m)^m\right\| = \left\|\left(U(t/m) - \tilde{U}(t/m)\right)\left(U(t/m)^{m-1} - \ldots - \tilde{U}(t/m)^{m-1}\right)\right\| \tag{159}$$

$$\leq m\left\|U(t/m) - \tilde{U}(t/m)\right\| = m\left\|E^{(3)}_{t/m}\right\| \tag{160}$$

$$\leq \frac{t^3}{m^2}C_E + \mathcal{O}\left(\frac{t}{m}\right)^4 \quad \text{for } t/m \to 0, \tag{161}$$

where

$$C_E = \frac{1}{6}\left\|\left[\left[H_2, H_3\right], \frac{1}{4}H_2 + \frac{1}{2}H_3\right]\right\|. \tag{162}$$

Using that

$$\|[H_2, H_3]\| = \|H_2H_3 - H_3H_2\| \leq 2\|H_2\|\|H_3\|, \tag{163}$$

and that $\|H_2\|, \|H_3\| \leq b$, we calculate

$$C_E \leq \frac{1}{3}\|[H_2, H_3]\|\left\|\frac{1}{4}H_2 + \frac{1}{2}H_3\right\| \tag{164}$$

$$\leq \frac{b^3}{2}. \tag{165}$$

Therefore,

$$\left\|V - e^{iH_1t}\tilde{U}^m(t/m)\right\| = \mathcal{O}\left(\frac{t^3b^3}{2m^2}\right) \quad \text{for } t/m \to 0. \tag{166}$$

□

However, the Quantum Phase Estimation uses powers of e^{iAt}. The next result will be useful to find the right number of Trotter steps for the implementation.

Proposition 6.1. *Let m and ϵ_A as before. Let $\tilde{V}'$ denote an approximation obtained using m' steps. Then for $m' = m\lfloor\sqrt{k}\rfloor$, it holds*

$$\left\|V^k - \tilde{V}'^k\right\| = \mathcal{O}\left(\epsilon_A\right) \quad \textit{for } \epsilon_A \to 0. \tag{167}$$

Proof. The proof is analogous as the one for Lemma 6.2, but instead of Eq. 159 we have

$$\left\|U(t)^k - \left(\tilde{U}(t/m)^m\right)^k\right\| = \left\|U(tk) - \tilde{U}(t/m)^{km}\right\| \tag{168}$$

$$\leq km\left\|U(t/m) - \tilde{U}(t/m)\right\| = km\left\|E^{(3)}_{t/m}\right\| \tag{169}$$

$$= \mathcal{O}\left(k \cdot \frac{t^3b^3}{m^2}\right) \quad \text{for } t/m \to 0. \tag{170}$$

□

Now we can prove Lemma 6.3.

Proof. We will denote U_1 the unitary matrix corresponding to the application of the powers of V; U_2 the unitary matrix corresponding to the $\text{QFT}^\dagger$, inversion of eigenvalues and QFT; and U_3 to the application of powers of the inverse of V. Similarly, $\tilde{U}_1$ and $\tilde{U}_3$ will denote the matrices corresponding to the same parts of the algorithm but using $\tilde{V}$ (U_2 is the same in both cases).

Then we can write

$$\text{HHL}(A, |b\rangle) = U_3U_2U_1\,|b\rangle\,, \tag{171}$$

and

$$\text{HHL}(\tilde{A}, |b\rangle) = \tilde{U}_3U_2\tilde{U}_1\,|b\rangle\,. \tag{172}$$

We can express $\tilde{V}$ as

$$\tilde{V} = V + E, \quad \text{where} \quad \|E\| < \epsilon_A. \tag{173}$$

Then, for $N = 2^{n_l}, n_l \in \mathbb{N}$, and using Proposition 6.1,

$$\tilde{U}_1 = \frac{1}{\sqrt{N}} \left(\sum_{k=0}^{N-1} |k\rangle \langle k| \otimes \left(V^k + E\right) \right) = U_1 + \frac{1}{\sqrt{N}} \left(\sum_{k=0}^{N-1} |k\rangle \langle k| \right) \otimes E. \quad (174)$$

Similarly,

$$\tilde{U}_3 = \frac{1}{\sqrt{N}} \left(\sum_{k=0}^{N-1} |k\rangle \langle k| \otimes \left(V^{-k} + E\right) \right) = U_1 + \frac{1}{\sqrt{N}} \left(\sum_{k=0}^{N-1} |k\rangle \langle k| \right) \otimes E. \quad (175)$$

Thus, expanding and using that the U_i are unitary,

$$\left\| U_3 U_2 U_1 - \tilde{U}_3 U_2 \tilde{U}_1 \right\| = \mathcal{O}\left(\left\| \frac{1}{\sqrt{N}} \left(\sum_{k=0}^{N-1} |k\rangle \langle k| \right) \otimes E \right\| \right) \quad (176)$$

$$= \mathcal{O}(\|E\|) = \mathcal{O}(\epsilon_A) \quad \text{for } \epsilon_A \to 0. \quad (177)$$

Finally,

$$\left\| |x\rangle - \mathrm{HHL}(\tilde{A}, |b\rangle) \right\| \leq \quad (178)$$

$$\| |x\rangle - \mathrm{HHL}(A, |b\rangle) \| + \left\| \mathrm{HHL}(A, |b\rangle) - \mathrm{HHL}(\tilde{A}, |b\rangle) \right\| \quad (179)$$

$$= \mathcal{O}(\epsilon_H + \epsilon_A) \quad \text{for } \epsilon_A, \epsilon_H \to 0. \quad (180)$$

□

6.3 Gate analysis

This subsection is dedicated to analyse the gate cost of the procedure described above and to investigate how the number of gates escalates with the dimension of the matrix.

Let $n_l \in \mathbb{N}$ denote the size of the register storing the representation of the eigenvalues, $n_b \in \mathbb{N}$ denote the number of qubits used to represent the solution, $N = 2^{n_b}$ the size of the matrix and m the exponent of the Trotter formula as above. For the quantum phase estimation (Section 4.2), we require $2^0, 2^1, \ldots, 2^{n_l-1}$ controlled applications of e^{iAt}, with Trotter exponent $m(i) = m\lfloor\sqrt{2^i}\rfloor$ for the i^{th} iteration. The circuit for the i^{th} controlled application, with the notation from the previous subsections, is shown in Figure 24. We can apply $e^{iH_2t/2m}$ uncontrolled because if $|q_i\rangle = |0\rangle$, with the notation from the Figure, then the terms inside the dashed box are not applied and therefore the uncontrolled gates cancel each other.

Proposition 6.2. *Let $f(n_l)$ denote the number of repetitions of the terms inside the dashed box from Figure 24, and m the number of Trotter steps. Then*

$$f(n_l) \leq 3m \left(\frac{2^{3\frac{n_l+1}{2}} - 1}{7} \right). \quad (181)$$

Proof. The Trotter step at the i^{th} iteration is

$$m(i) = m\lfloor\sqrt{2^i}\rfloor = m2^{\lfloor i/2 \rfloor}. \quad (182)$$

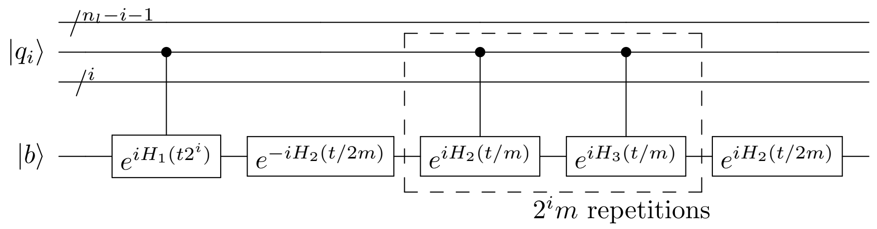


Figure 24: Circuit for e^{iAt2^i} controlled by $|q_i\rangle$ from the eigenvalues register with $m(i) = m$.

Then,

$$f(n_l) = \sum_{i=0}^{n_l-1} 2^i m \lfloor \sqrt{2^i} \rfloor = m \sum_{i=0}^{n_l-1} 2^i 2^{\lfloor i \rfloor} \leq m \sum_{i=0}^{\frac{n_l-1}{2}} 2^i \left(2^{2i} + 2^{2i+1}\right) \tag{183}$$

$$= 3m \sum_{i=0}^{\frac{n_l-1}{2}} 2^{3i} = 3m \left(\frac{2^{3\frac{nl+1}{2}} - 1}{7} \right). \tag{184}$$

□

In Section 7 we will show that $n_l = \mathcal{O}(\log_2(1/\epsilon))$, therefore $f(n_l) = \mathcal{O}(m/\epsilon)$, where ϵ is the tolerance of the complete HHL. As for the polynomial state preparation, we show first the table with the different gate counts for the general case and afterwards the calculations, where the decompositions used are those from Section 4.2.

	CNOTs	one-qubit gates	ancilla
H_1	$6n_l$	$8n_l$	0
H_2	0	n_l	0
H_2, H_3	$\mathcal{O}(m\log_2^2(N)/\epsilon)$	$\mathcal{O}(m\log_2^2(N)/\epsilon)$	$n_b - 1$
H_2	0	n_l	0
Total	$\mathcal{O}(m\log_2^2(N)/\epsilon)$	$\mathcal{O}(m\log_2^2(N)/\epsilon)$	$n_b - 1$

Table 3: Gate count for the Hamiltonian simulation.

We begin analysing the controlled applications of H_1. Its controlled version consists on two CNOTs and two controlled one-qubit gates, and we use it a total of n_l times. A controlled one-qubit gate can be decomposed into 2 CNOTs and 4 one-qubit gates. Therefore, the total cost of H_1 is

$$\begin{array}{c} n_l(2\text{CNOTs} + 2(2\text{CNOTs} + 4\{\text{ one-qubit gate }\})) = \\ 6n_l \cdot \text{CNOTs} + 8n_l \cdot \{\text{ one-qubit gate }\} \end{array}. \tag{185}$$

Since we use the non-controlled version of H_2, it only needs one-qubit gates, which after n_l iterations gives a total of n_l one-qubit gates.

Proposition 6.3. *The total cost of the terms inside the dashed box from Figure 24 is*

$$\mathcal{O}(m\log_2^2(N)/\epsilon), \quad \textit{for } \epsilon \to 0, \tag{186}$$

CNOTs and one-qubit gates.

Proof. The cost of a controlled H_2 is

$$2n_l \cdot \text{ CNOTs } + 4n_l \cdot \{\text{ one-qubit gate }\}. \tag{187}$$

To calculate the cost of H_3 we need to take into account each C_j block, $1 \leq j \leq n_b - 1$. For the controlled version of the circuit shown in Figure 23, instead of adding an extra control to each gate, we can just add the extra control to the rotation gate. This is to reduce the number of gates, and can be done because the CNOTs will cancel each other if the rotation is not applied.

Thus, the C_j controlled block will need $2j$ CNOTs and a j-controlled R_x. The latter can be decomposed into $(j-1)$ ancilla, $20(j-1)+4$ one qubit gates and $12(j-1)+2$ CNOTs. Giving a total of

$$(14j - 10) \cdot \text{CNOTs} + (20j - 16) \cdot \{\text{ one-qubit gate }\}. \tag{188}$$

Summing over all C_j gives the total cost of one controlled application of H_3,

$$\sum_{j=1}^{n_b-1} (14j - 10) \cdot \text{CNOTs} + (20j - 16) \cdot \{\text{ one-qubit gate }\} = \tag{189}$$

$$= \begin{cases} (n_b - 1)(7(n_b - 2) - 10) \cdot \text{ CNOTs} \\ +(n_b - 1)(10(n_b - 2) - 16) \cdot \{\text{ one-qubit gate }\} \end{cases} \tag{190}$$

Using that $n_b = \log_2(N)$ and Proposition 6.2 gives the result. □

7 Inversion of Eigenvalues

Let $n_l \in \mathbb{N}$ denote the size of the register storing the binary representation of the eigenvalues, $n_b \in \mathbb{N}$ the size of the register containing the solution and $N = 2^{n_b}$. After the QPE, and assuming no errors in the computations, the computer is in the state

$$\sum_{j=0}^{N-1} b_j \left|0\right\rangle \left|\tilde{\lambda}_j\right\rangle_{n_l} \left|u_j\right\rangle_{n_b}, \tag{191}$$

Here $\tilde{\lambda}_j$ is the n_l-bit representation of $\frac{\lambda_j t}{2\pi}$, and λ_j denotes the j^{th} eigenvalue of our matrix A for $\lambda_{\min} \coloneqq \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_{N-1} \eqqcolon \lambda_{\max}$.

We are now in Step (iii) from Section 4.1. The goal is to find a circuit performing the simultaneous transformation

$$\sum_{j=0}^{N-1} b_j \left|0\right\rangle \left|\tilde{\lambda}_j\right\rangle_{n_l} \left|u_j\right\rangle_{n_b} \mapsto \sum_{j=0}^{N-1} b_j \left(\sqrt{1-\frac{1}{\tilde{\lambda}_j^2}}\left|0\right\rangle + \frac{1}{\tilde{\lambda}_j}\left|1\right\rangle\right) \left|\tilde{\lambda}_j\right\rangle_{n_l} \left|u_j\right\rangle_{n_b}. \tag{192}$$

Let $\left|\tilde{x}\right\rangle$ denote the approximated solution to the system of linear equations. Then, after restoring the eigenvalues register back to $\left|0\right\rangle_{n_l}$ by applying the inverse QPE, if we measure the first qubit in the computational basis and obtain 1, we know that the post-measurement state is

$$\sum_{j=0}^{N-1} \frac{b_j}{\tilde{\lambda}_j} \left|1\right\rangle \left|0\right\rangle_{n_l} \left|u_j\right\rangle_{n_b} = \left|\tilde{x}\right\rangle. \tag{193}$$

To achieve this, we can use the Polynomial State Preparation procedure and approximate the function

$$f(x) = \arcsin\left(\frac{t}{2\pi x}\right), \quad x \in \left[0, 2^{n_l - 1}\right], \quad t \leq \frac{2\pi}{\lambda_{\max}}. \tag{194}$$

Thus, we take a polynomial $p : \left[0, 2^{n_l-1}\right] \to [-\pi/2, \pi/2]$ such that

$$p(x) \approx \arcsin\left(\frac{t}{2\pi x}\right). \tag{195}$$

Then, applying the PSP algorithm to the n_l register results in

$$\sum_{j=0}^{N-1} b_j \left|0\right\rangle \left|\tilde{\lambda}_j\right\rangle_{n_l} \left|u_j\right\rangle_{n_b} \mapsto \sum_{j=0}^{N-1} b_j \left(\cos\Big(p(\tilde{\lambda}_j)\Big)\left|0\right\rangle + \sin\Big(p(\tilde{\lambda}_j)\Big)\left|1\right\rangle\right) \left|\tilde{\lambda}_j\right\rangle_{n_l} \left|u_j\right\rangle_{n_b} \tag{196}$$

$$\approx \sum_{j=0}^{N-1} b_j \left(\cos(\cdots)\left|0\right\rangle + \frac{t}{2\pi\tilde{\lambda}_j}\left|1\right\rangle\right) \left|\tilde{\lambda}_j\right\rangle_{n_l} \left|u_j\right\rangle_{n_b} \tag{197}$$

$$\approx \sum_{j=0}^{N-1} b_j \left(\cos(\cdots)\left|0\right\rangle + \frac{1}{\lambda_j}\left|1\right\rangle\right) \left|\tilde{\lambda}_j\right\rangle_{n_l} \left|u_j\right\rangle_{n_b}, \tag{198}$$

where in the derivation of the last line we used that $\tilde{\lambda}_j$ is the n_l-bit representation of $\frac{\lambda_j t}{2\pi}$. Finally, after measuring the ancilla qubit in 198 and obtaining an outcome of 1, the post-measurement state is approximately

$$\sum_{j=0}^{N-1} \frac{b_j}{\lambda_j} |1\rangle \left|\tilde{\lambda}_j\right\rangle_{n_l} |u_j\rangle_{n_b}, \tag{199}$$

which after restoring the n_l-register back to 0, represents an approximation to the exact solution of the system $|x\rangle$.

7.1 Error analysis

Let $A \in \mathbb{C}^{2^{n_b} \times 2^{n_b}}$ be a Hermitian matrix and $N = 2^{n_b}$. Let λ_j denote the j^{th} eigenvalue, with $\lambda_{\min} := \lambda_0$ and $\lambda_{\max} = \lambda_{N-1}$. Let $n_l = \log_2(\epsilon)$ denote the size of the register for the lambdas, where ϵ is the error we want to allow for the HHL.

Definition 7.1. We denote by $\tilde{\lambda}_j$ the integer in the interval $[0, 2^{n_l} - 1]$ such that $\frac{\tilde{\lambda}_j}{2^{n_l}}$ is the best n_l-bit binary approximation to $\frac{\lambda_j t}{2\pi}$ which is less than $\frac{\lambda_j t}{2\pi}$.

For now and until stated otherwise we will relabel $\frac{\lambda_j t}{2\pi}$ as λ_j . Then

$$0 \leq \delta_j := \lambda_j - \frac{\tilde{\lambda}_j}{2^{n_l}} \leq 2^{-n_l} \tag{200}$$

With the following proposition we show how to pick t to obtain the best approximation.

Proposition 7.1. *Let δ_j, $\tilde{\lambda}_j$ and λ_j as above. Then*

$$\left| \frac{1}{\lambda_j} - \frac{2^{n_l}}{\tilde{\lambda}_j} \right| \leq \frac{2^{n_l}}{\tilde{\lambda}_j^2}. \tag{201}$$

Proof. From the definition of δ_j we have that

$$\left| \frac{1}{\lambda_j} - \frac{2^{n_l}}{\tilde{\lambda}_j} \right| = \left| \frac{\tilde{\lambda}_j - 2^{n_l}\lambda_j}{\lambda_j \tilde{\lambda}_j} \right| = \left| \frac{2^{n_l}\delta_j}{\lambda_j \tilde{\lambda}_j} \right|. \tag{202}$$

On the other hand,

$$\lambda_j \tilde{\lambda}_j = \left(\delta_j + \frac{\tilde{\lambda}_j}{2^{n_l}} \right) \tilde{\lambda}_j \geq \frac{\tilde{\lambda}_j^2}{2^{n_l}}. \tag{203}$$

Therefore, using that $\delta_j \leq 2^{-n_l}$,

$$\left| \frac{1}{\lambda_j} - \frac{2^{n_l}}{\tilde{\lambda}_j} \right| \leq \frac{2^{n_l}\delta_j 2^{n_l}}{\tilde{\lambda}_j^2} \leq \frac{2^{n_l}}{\tilde{\lambda}_j^2}. \tag{204}$$

□

Hence, the approximation is better the closer the $\tilde{\lambda}_j$ are to 2^{n_l}.

8 Observables

After running the algorithm, the solution to the system of linear equations will be encoded in the amplitudes of the final state $|\tilde{x}\rangle$. In this section we will study how to obtain information about the solution from this quantum state.

Let n_b be the size of the register containing the solution, $N = 2^{n_b}$ and n_l the size of the eigenvalues register. After restoring the n_l-register back to $|0\rangle$, but before measuring whether the inversion of eigenvalues has been successful, the state is

$$|\psi\rangle = \sum_{j=0}^{N-1} b_j \left(\sqrt{1 - \frac{1}{\tilde{\lambda}_j^2}} |0\rangle + \frac{1}{\tilde{\lambda}_j} |1\rangle \right) |0\rangle_{n_l} |u_j\rangle_{n_b} . \tag{192}$$

For this section, $|x\rangle$ and $|\tilde{x}\rangle$ will denote, respectively, the exact and approximated solutions to the system before normalisation, and $\|\cdot\|$ the 2-norm.

The following result states that the norm of the solution is given by the probability of seeing a 1 in the conditioned rotation of the eigenvalues.

Proposition 8.1. *Let $\mathbb{P}[|1\rangle]$ denote the probability of measuring $|1\rangle$ for the inversion of eigenvalues. Then*

$$\mathbb{P}[|1\rangle] = \| |\tilde{x}\rangle \|^2. \tag{205}$$

Proof. Since we only measure the ancilla qubit, we take the collection of measurement operators $M_i = |i\rangle \langle i| \otimes I_{n_l + n_b}$ for $i \in \{0, 1\}$. Furthermore, we are looking for an outcome of 1, which means we are interested in the operator M_1. Thus, we calculate

$$M_1 |\psi\rangle = \sum_{j=0}^{N-1} \frac{b_j}{\tilde{\lambda}_j} |1\rangle |0\rangle_{n_l} |u_j\rangle_{n_b} . \tag{206}$$

Therefore,

$$\mathbb{P}[|1\rangle] = \langle\psi| M_1^\dagger M_1 |\psi\rangle = \sum_{j=0}^{N-1} \frac{|b_j|^2}{\left|\tilde{\lambda}_j\right|^2} = \| |\tilde{x}\rangle \|^2. \tag{207}$$

□

We will proceed to show how to compute different functions of the solution vector. The idea is always the same: add some extra gates at the end of the HHL and then measure in the computational basis. By varying the gates and the qubits measured, one can compute different functions.

Average

We can write the solution vector in the standard basis as $|\tilde{x}\rangle = \sum_{i=0}^{N-1} x_i |i\rangle$. The goal will be to compute the quantity

$$\left| \frac{1}{N} \sum_{i=0}^{N-1} x_i \right|. \tag{208}$$

To do so, we can append the circuit shown in Figure 25 at the end of the algorithm and measure the each qubit from the solution register in the computational basis.

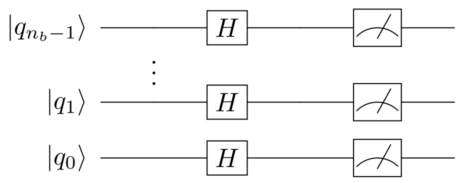


Figure 25: Hadamards to compute the average of the solution.

Proposition 8.2. *Let* $\mathbb{P}\left[|0\rangle\right]$ *denote the probability of measuring all* 0*'s in Figure 25. Then*

$$\mathbb{P}\left[|0\rangle\right] = \left| \frac{1}{N} \sum_{i=0}^{N-1} x_i \right|^2 . \tag{209}$$

Proof. First, we compute explicitly

$$H^{\otimes n_b} \otimes |\tilde{x}\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} x_i \sum_{k=0}^{N-1} (-1)^{k \cdot i} |k\rangle . \tag{210}$$

We will use the collection of measurement operators $M_i = |i\rangle \langle i|$, $i \in \{0, 1, ..., N-1\}$. We are interested in M_0. Thus,

$$M_0 H^{\otimes n_b} \otimes |\tilde{x}\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} x_i |0\rangle . \tag{211}$$

Therefore,

$$\mathbb{P}\left[|0\rangle\right] = \left| \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} x_i \right|^2 . \tag{212}$$

□

Since N is a known quantity, we can then obtain Eq. 208.

Compliance output functional

Let A be an $N \times N$ tri-diagonal symmetric matrix of the form

$$A = \begin{pmatrix} a & b & & 0 \\ b & \ddots & \ddots & \\ & \ddots & & b \\ 0 & & b & a \end{pmatrix}, \quad a, b \in \mathbb{R}, \tag{213}$$

and let

$$F(x) = \langle x| A |x\rangle = a \sum_{i=0}^{N-1} |x_i|^2 + 2b \sum_{i \neq j} \mathrm{Re}\left(x_i x_j^*\right), \tag{214}$$

where $|x\rangle \in \mathbb{C}^N$ and where $\mathrm{Re}(z)$ denotes the real part of $z \in \mathbb{C}$.

In this subsection we will give an algorithm to calculate $F(\tilde{x})$ for a given A with n_b different measurements. The algorithm consists on n_b steps, at each step we use a different measurement. At this point, it will be useful to write the Hadamard gate as

$$H = \frac{1}{\sqrt{2}} \sum_{i,j\{0,1\}} (-1)^{i \cdot j} |i\rangle \langle j| . \tag{215}$$

Let i denote the step of the algorithm, then the detailed description of the algorithm is as follows.

i=1

This step consists of adding a single Hadamard gate on the last qubit as shown in Figure 26. Then measure the last qubit in the computational basis applying the map $0 \mapsto 1, 1 \mapsto -1$. Denote by n_0 the number of times we observe $|0\rangle$ and

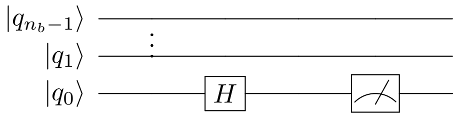


Figure 26: Single Hadamard gate. First step towards calculating $F(x)$.

by n_1 the number of times we observe $|1\rangle$ in the circuit from Figure 26.

Proposition 8.3. *Let n_0 and n_1 as above with $n_0 + n_1 \to \infty$. Then*

$$\mathbb{E}\left(\frac{n_0 - n_1}{n_0 + n_1}\right) = \mathbb{P}[|0\rangle] - \mathbb{P}[|1\rangle] = 2 \sum_{i=0}^{N/2-1} \operatorname{Re}\left(x_{2i} x_{2i+1}^*\right). \tag{216}$$

Proof. The operator associated with the circuit from Figure 26 is $I_{n_b-1} \otimes H$. We will use the collection of measurement operators $M_i = I_{n_b-1} \otimes |i\rangle \langle i|$, for $i \in \{0, 1\}$. Then,

$$M_i \left(I_{n_b-1} \otimes H\right) |\tilde{x}\rangle = \left(I_{n_b-1} \otimes \left(\frac{1}{\sqrt{2}} \sum_{j=0}^{1} (-1)^{i \cdot j} |i\rangle \langle j| \right) \right) |\tilde{x}\rangle \tag{217}$$

$$= \frac{1}{\sqrt{2}} \sum_{k=0}^{N/2-1} ((-1)^0 x_{2k} + (-1)^i x_{2k+1}) |2k + i\rangle . \tag{218}$$

Therefore,

$$\mathbb{P}[|i\rangle] = \frac{1}{2} \sum_{k=0}^{N/2-1} (x_{2k} + (-1)^i x_{2k+1})^* (x_{2k} + (-1)^i x_{2k+1}) \tag{219}$$

$$= \frac{1}{2} \left(\sum_{k=0}^{N-1} |x_i|^2 + (-1)^i 2 \sum_{i=0}^{N/2-1} \operatorname{Re}\left(x_{2i} x_{2i+1}^*\right) \right) . \tag{220}$$

Hence,

$$\mathbb{P}[|0\rangle] + \mathbb{P}[|1\rangle] = 2 \sum_{i=0}^{N/2-1} \operatorname{Re}\left(x_{2i} x_{2i+1}^*\right) . \tag{221}$$

□

i=k

The circuit for the k^{th} step, $k > 1$, is shown in Figure 27. Now we will measure the last k qubits in the computational basis. We want to see $|q_0\rangle = \cdots = |q_{n_b-2}\rangle = |1\rangle$ and apply the same mapping as before for the results of measuring $|q_{k-1}\rangle$.

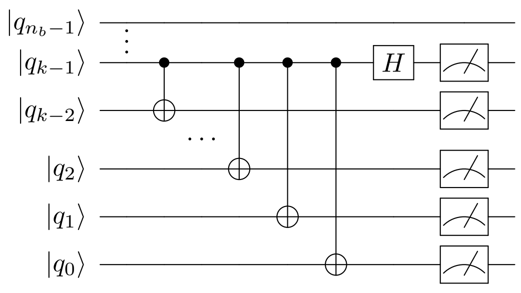


Figure 27: k^{th} step towards calculating $F(x)$.

Proposition 8.4. *Let* $\mathbb{P}_k[|i\rangle]$ *denote the probabilty of outcome* $|q_0\rangle = \cdots = |q_{k-2}\rangle = |1\rangle$ *and* $|q_{k-1}\rangle = |i\rangle$*, for* $i \in \{0, 1\}$*. Then*

$$\mathbb{P}_k[|0\rangle] - \mathbb{P}_k[|1\rangle] = 2 \sum_{i=-1 \mod 2^k}^{N-2} \operatorname{Re}\left(x_i x_{i+1}^*\right). \tag{222}$$

Proof. Note that with the circuit from Figure 27, the outcome $|q_0\rangle = \cdots = |q_{k-2}\rangle = |1\rangle$ can only happen when originally either

$$|q_0\rangle = \cdots = |q_{k-2}\rangle = |1\rangle \quad \text{and} \quad |q_{k-1}\rangle = |0\rangle, \tag{223}$$

or

$$|q_0\rangle = \cdots = |q_{k-2}\rangle = |0\rangle \quad \text{and} \quad |q_{k-1}\rangle = |1\rangle. \tag{224}$$

That is, for those basis state $|i\rangle$ such that $i = 0 \mod 2^k$ or $i = -1 \mod 2^k$. Parting from the point at which we have already measured $|q_0\rangle = \cdots = |q_{k-2}\rangle = |1\rangle$, calculating the probability of $|q_{k-1}\rangle = |0\rangle$ or $|q_{k-1}\rangle = |1\rangle$ can be done as in Proposition 8.3. Therefore,

$$\mathbb{P}_k[|i\rangle] = \frac{1}{2}\left(\sum_{i=0 \text{ or } -1 \mod 2^k}^{N-1} |x_i|^2 + (-1)^i 2 \sum_{i=-1 \mod 2^k}^{N-2} \operatorname{Re}\left(x_i x_{i+1}^*\right)\right), \tag{225}$$

and the result follows. □

Finally,

$$\sum_{k=1}^{n_b} 2 \sum_{i=-1 \mod 2^k}^{N-2} \operatorname{Re}\left(x_i x_{i+1}^*\right) = 2 \sum_{i \neq j} \operatorname{Re}\left(x_i x_j^*\right). \tag{226}$$

Both a and b are known parameters. And $\sum_{i=0}^{N-1} |x_i|^2 = \||\tilde{x}\rangle\|^2$ can be calculated from Proposition 8.1. Therefore, we have a method to compute $F(x)$ with only $n_b = \log_2(N)$ different measurements.

9 HHL error analysis

The goal of this section is to prove the following theorem.

Theorem 9.1. *Let $|x\rangle$ denote the exact solution to the system, and* HHL $\left(\tilde{A}, \left|\tilde{b}\right\rangle\right)$ *the solution returned by the HHL algorithm using approximations for the initial state and Hamiltonian simulation. Then*

$$\left\| |x\rangle - \text{HHL}\left(\tilde{A}, |b\rangle\right) \right\| = \mathcal{O}(\kappa c^2 + \epsilon_A + \epsilon_H) \quad \textit{for } c, \epsilon_A, \epsilon_H \to 0. \tag{227}$$

There are two parameters that we can adjust in the algorithm. In Section 9.1 we show how to choose them to achieve a solution within a target tolerance. For the error analysis we take into account the Polynomial State Preparation approximation given in Section 5, the Hamiltonian simulation shown in Section 6 and assume that the function for the eigenvalues inversion is exact. Throughout this section, A will denote the tri-diagonal symmetric matrix associated to the system of equations we want to solve.

9.1 Parameters

There are two parameters which can be adjusted to improve the performance of the algorithm: t and m.

From Proposition 6.1, we have that for the j^{th} power within the QPE, the number of Trotter steps should be adjusted to $m_j = \left\lfloor m\sqrt{2^j} \right\rfloor$, and $\left\lceil \sqrt{\frac{t^3 b^3}{2\epsilon_A}} \right\rceil$, where ϵ_A is the error from the approximated Hamiltonian simulation.

The values represented in binary in the eigenvalue register are $\frac{\lambda_j t}{2\pi}$ rather than λ_j. Thus, the goal is to find $t \in \mathbb{R}$ such that:

(i) $\frac{\lambda_j t}{2\pi} \in [0,1) \forall j$.

(ii) From Proposition 8.4, the quantities $\frac{\lambda_j t}{2\pi}$ should be as close to 1 as possible.

(iii) $\frac{\lambda_{\min} t}{2\pi}$ should be represented exactly since it has the largest contribution in $\sum \frac{b_j}{\lambda_j} |u_j\rangle$.

The largest value representable in binary with n_l digits is $\left(1 - \frac{1}{2^{n_l}}\right)$. Thus, to satisfy (i),

$$t \leq \frac{2\pi \left(1 - \frac{1}{2^{n_l} t}\right)}{\lambda_{\max}} =: T. \tag{228}$$

Now let $\tilde{\lambda}_{\min} \in [0, 2^{n_l} - 1]$ be such that $\frac{\tilde{\lambda}_{\min}}{2^{n_l}}$ is the best n_l-bit binary approximation to $\frac{\lambda_{\min} T}{2\pi}$ which is less than $\frac{\lambda_{\min} T}{2\pi}$.

To satisfy (iii), we will set the time of the simulation to

$$t = \frac{1}{2^{n_l}} \cdot \frac{\tilde{\lambda}_{\min} 2\pi}{\lambda_{\min}}. \tag{229}$$

9.2 Error analysis

Let $n_l \in \mathbb{N}$ denote the size of the register storing the binary representation of the eigenvalues, $n_b \in \mathbb{N}$ the size of the register containing the solution and $N = 2^{n_b}$.The state of the computer after the Quantum Phase Estimation is

$$\sum_{j=0}^{N-1} b_j \left|u_j\right\rangle \left(\sum_{l=0}^{2^{n_l}-1} \alpha_{l|j} \left|l\right\rangle \right), \tag{230}$$

where

$$\alpha_{l|j} = \frac{1}{2^{n_l}} \sum_{k=0}^{2^{n_l}-1} \left(e^{2\pi i \left(\frac{\lambda_j t}{2\pi} - \frac{t}{2^{n_l}} \right)} \right)^k. \tag{231}$$

The first step towards finding a bound for the overall error of the algorithm will be to bound these coefficients $\alpha_{l|j}$. We can relabel the second register in Eq. 230 so that $\alpha_{l|j}$ denotes the amplitude of $\left| l + \tilde{\lambda}_j \;(\mod 2^{n_l}) \right\rangle$. So now,

$$\alpha_{l|j} := \frac{1}{2^{n_l}} \sum_{k=0}^{2^{n_l}-1} \left(e^{2\pi i \left(\frac{\lambda_j t}{2\pi} - \frac{l+\tilde{\lambda}_j}{2^{n_l}} \right)} \right)^k. \tag{232}$$

With this notation, the best possible scenario occurs when $\alpha_{0|j} = 1 \quad \forall j$ and $\alpha_{l|j} = 0 \quad \forall l \neq 0 \quad \forall j$.

Remark. Note that fixing j, from the fact that all the gates used were unitary, we have the identity

$$\sum_{l=-2^{n_l-1}+1}^{2^{n_l-1}} \left|\alpha_{l|j}\right|^2 = 1. \tag{233}$$

The following result will be needed for the proof of Lemma 9.2.

Proposition 9.1. *Let $\theta \in \mathbb{R}$, then*

$$\left|1 - e^{i\theta}\right| \geq \frac{2|\theta|}{\pi} \text{ for } -\pi \leq \theta \leq \pi. \tag{234}$$

Proof.

$$\begin{aligned} \left|1 - e^{i\theta}\right| &= |(1 - \cos(\theta)) + i \sin(\theta)| = \sqrt{(1 - \cos(\theta))^2 + \sin^2(\theta)} \\ &= \sqrt{2 - 2\cos(\theta)} = \left| 2 \sin\left(\frac{\theta}{2}\right) \right|. \end{aligned} \tag{235}$$

Since $\sin(\theta)$ is concave for $\theta \in [0, \pi/2]$, and it takes the same values as $\frac{2\theta}{\pi}$ at the endpoints of the interval while $\sin\left(\frac{\pi}{4}\right) = \frac{\sqrt{2}}{2} > \frac{1}{2}$, we have that $\sin(\theta) \geq \frac{2\theta}{\pi}$ for $\theta \in [-\pi/2, \pi/2]$. Then, by symmetry of the functions, we obtain that $|\sin(\theta)| \geq \left|\frac{2\theta}{\pi}\right|$ for $\theta \in [-\pi/2, \pi/2]$.

Hence, for $-\pi \leq \theta \leq \pi$,

$$\left|1 - e^{i\theta}\right| = \left| 2 \sin\left(\frac{\theta}{2}\right) \right| \geq \frac{2|\theta|}{\pi}. \tag{236}$$

□

Lemma 9.2. *Let δ_j as defined in Eq. 200 and t as in Eq. 229. For $-2^{n_l-1} < l \leq 2^{n_l-1}$ it holds that*

$$\left|\alpha_{l|j}\right| \leq \frac{1}{2^{n_l+1}\left(\delta_j - l/2^{n_l}\right)}, \tag{237}$$

where $1 \leq j \leq N-1$. Furthermore, $\alpha_{0|0} = 1$ and $\alpha_{l|0} = 0$ for $l \neq 0$.

Proof. Since t is chosen such that $\delta_0 = 0$, we have that

$$\alpha_{0|0} = \frac{1}{2^{n_l}} \sum_{k=0}^{2^{n_l}-1} 1 = 1. \tag{238}$$

And for $l \neq 0$,

$$\alpha_{l|0} = \frac{1}{2^{n_l}} \sum_{k=0}^{2^{n_l}-1} \left(e^{2\pi i \frac{1}{2^{n_l}}}\right)^k = 0. \tag{239}$$

Let now $j > 0$. Since the $\alpha_{l|j}$ are the sum of a geometric series, one has

$$\alpha_{l|j} = \frac{1}{2^{n_l}} \left(\frac{1 - e^{2\pi i \left(\lambda_j - \frac{l+\tilde{\lambda}_j}{2^{n_l}}\right) 2^{n_l}}}{1 - e^{2\pi i \left(\lambda_j - \frac{l+\tilde{\lambda}_j}{2^{n_l}}\right)}} \right) \tag{240}$$

$$= \frac{1}{2^{n_l}} \left(\frac{1 - e^{2\pi i \left(\lambda_j 2^{n_l} - (l+\tilde{\lambda}_j)\right) 2^{n_l}}}{1 - e^{2\pi i \left(\lambda_j - \frac{l+\tilde{\lambda}_j}{2^{n_l}}\right)}} \right) \tag{241}$$

$$= \frac{1}{2^{n_l}} \left(\frac{1 - e^{2\pi i (2^{n_l}\delta_j - l)}}{1 - e^{2\pi i (\delta_j - l/2^{n_l})}} \right). \tag{242}$$

Using that for $\theta \in \mathbb{R}$, $\left|1 - e^{i\theta}\right| \leq 2$,

$$\left|\alpha_{l|j}\right| \leq \frac{2}{2^{n_l}\left|1 - e^{2\pi i (\delta_j - l/2^{n_l})}\right|}. \tag{243}$$

On the other hand, $\left|1 - e^{i\theta}\right| \geq \frac{2|\theta|}{\pi}$ for $-\pi \leq \theta \leq \pi$ by Proposition 234. And $-2^{n_l-1} < l \leq 2^{n_l-1}$ means that $-\pi \leq 2\pi\left(\delta_j - l/2^{n_l}\right) \leq \pi$. So we obtain the result

$$\left|\alpha_{l|j}\right| \leq \frac{1}{2^{n_l+1}\left(\delta_j - l/2^{n_l}\right)}. \tag{244}$$

□

Proposition 9.2. *Let k be an integer with $0 \leq k \leq n_l - 2$ and fix j. Then*

$$\sum_{|l|=2^k}^{2^{k+1}} \left|\alpha_{l|j}\right|^2 \leq \frac{1}{2^{k+2}}. \tag{245}$$

Proof. From Lemma 9.2, and using that $0 \leq 2^{n_l}\delta_j \leq 1$,

$$\sum_{|l|=2^k}^{2^{k+1}} \left|\alpha_{l|j}\right|^2 \leq \sum_{|l|=2^k}^{2^{k+1}} \frac{1}{4\left(2^{n_l}\delta_j - l\right)^2} \tag{246}$$

$$\leq \frac{1}{4}\left(\sum_{l=-2^{k+1}}^{-2^k} \frac{1}{(l-1)^2} + \sum_{l=2^k}^{2^{k+1}} \frac{1}{l^2}\right) \tag{247}$$

$$\leq \frac{1}{2}\sum_{l=2^k}^{2^{k+1}} \frac{1}{l^2} \leq \frac{1}{2}\int_{2^k}^{2^{k+1}} \frac{1}{l^2} dl \tag{248}$$

$$= \frac{1}{2}\left(\frac{1}{2^k} - \frac{1}{2^{k+1}}\right) \tag{249}$$

$$= \frac{1}{2^{k+2}}. \tag{250}$$

□

The following will be the last intermediate result before the proof of Theorem 9.1.

Proposition 9.3. *Fix j. Let λ_j denote the j^{th} eigenvalue of A. Define $f(l) := \frac{2^{n_l}}{l+\tilde{\lambda}_j} \cdot \frac{t}{2\pi}$ for $-2^{n_l-1} < l \leq 2^{n_l-1}$ and define $f(\lambda_j) := 1/\lambda_j$. Then*

$$\frac{2\pi}{t}\left|f\left(\lambda_j\right) - \sum_{l=-2^{n_l-1}+1}^{2^{n_l-1}} \left|\alpha_{l|j}\right|^2 f(l)\right| = \mathcal{O}\left(n_l \frac{2^{n_l}}{\tilde{\lambda}_j^2}\right) \quad \textit{for} \quad j \geq 1, \tag{251}$$

and

$$\left|f\left(\lambda_0\right) - \sum_{l=-2^{n_l-1}+1}^{2^{n_l-1}} \left|\alpha_{l|0}\right|^2 f(l)\right| = 0. \tag{252}$$

Proof. From Proposition 9.2,

$$\frac{2\pi}{t}\left|f\left(\lambda_0\right) - \sum_{l=-2^{n_l-1}+1}^{2^{n_l-1}} \left|\alpha_{l|0}\right|^2 f(l)\right| = \left|f\left(\lambda_0\right) - f(0)\right| = \left|\frac{2\pi}{t\lambda_0} - \frac{2^{n_l}}{\tilde{\lambda}_0}\right| = 0. \tag{253}$$

From now on, let λ_j be a label for $\frac{\lambda_j t}{2\pi}$. For $j \geq 1$, using the identity from Eq. 233,

$$\frac{2\pi}{t}\left|\frac{t}{2\pi\lambda_j} - \sum_{l=-2^{n_l-1}+1}^{2^{n_l-1}} \left|\alpha_{l|j}\right|^2 f(l)\right| = \sum_{l=-2^{n_l-1}}^{2^{n_l-1}} \left|\alpha_{l|j}\right|^2 \left|\frac{1}{\lambda_j} - \frac{2^{n_l}}{l+\tilde{\lambda}_j}\right| \tag{254}$$

$$\leq \left|\alpha_{0|j}\right|^2 \left|\frac{1}{\lambda_j} - \frac{2^{n_l}}{\tilde{\lambda}_j}\right| + \sum_{|l|=1}^{2^{n_l-1}} \left|\alpha_{l|j}\right|^2 \left|\frac{1}{\lambda_j} - \frac{2^{n_l}}{l+\tilde{\lambda}_j}\right|. \tag{255}$$

First, from Proposition 7.1,

$$\left|\alpha_{0|j}\right|^2 \left|\frac{1}{\lambda_j} - \frac{2^{n_l}}{\tilde{\lambda}_j}\right| \leq \frac{2^{n_l}}{\tilde{\lambda}_j^2}. \tag{256}$$

And using Proposition 9.2,

$$\sum_{|l|=1}^{2^{n_l-1}} \left|\alpha_{l|j}\right|^2 \left|\frac{1}{\lambda_j} - \frac{2^{n_l}}{l+\tilde{\lambda}_j}\right| \leq \sum_{k=0}^{n_l-2} \sum_{|l|=2^k}^{n_l-2} \left|\alpha_{l|j}\right|^2 \left|\frac{1}{\lambda_j} - \frac{2^{n_l}}{l+\tilde{\lambda}_j}\right| \tag{257}$$

$$\leq \sum_{k=0}^{n_l-2} \left|\frac{1}{\lambda_j} - \frac{2^{n_l}}{2^{k+1}+\tilde{\lambda}_j}\right| \sum_{|l|=2^k}^{2^{k+1}} \left|\alpha_{l|j}\right|^2 \tag{258}$$

$$\leq \sum_{k=0}^{n_l-2} \frac{1}{2^{k+2}} \left|\frac{1}{\lambda_j} - \frac{2^{n_l}}{2^{k+1}+\tilde{\lambda}_j}\right| = \sum_{k=0}^{n_l-2} \frac{1}{2^{k+2}} \left|\frac{2^{k+1}+\tilde{\lambda}_j - 2^{n_l}\lambda_j}{\lambda_j\left(2^{k+1}+\tilde{\lambda}_j\right)}\right| \tag{259}$$

$$= \sum_{k=0}^{n_l-2} \frac{1}{2^{k+2}} \left|\frac{2^{k+1} - 2^{n_l}\delta_j}{\lambda_j\left(2^{k+1}+\tilde{\lambda}_j\right)}\right| \leq \sum_{k=0}^{n_l-2} \frac{1}{2^{k+2}} \left|\frac{2^{k+1}}{\lambda_j\left(2^{k+1}+\tilde{\lambda}_j\right)}\right| \tag{260}$$

$$\leq \frac{1}{2} \sum_{k=0}^{n_l-2} \left|\frac{2^{n_l}}{\tilde{\lambda}_j^2}\right| = \frac{n_l-1}{2} \cdot \frac{2^{n_l}}{\tilde{\lambda}_j^2}. \tag{261}$$

Combining Eq. 256 and Eq. 261 gives

$$\left|f\left(\lambda_j\right) - \sum_{l=-2^{n_l-1}+1}^{2^{n_l-1}} \left|\alpha_{l|j}\right|^2 f(l)\right| \leq \frac{n_l+1}{2} \cdot \frac{2^{n_l}}{\tilde{\lambda}_j^2}. \tag{262}$$

□

Proposition 9.4. *Let $f(l)$ as before. Suppose that the eigenvalues register has been restored back to $|0\rangle_{n_l}$ and that the eigenvalues inversion has been successful. Then the post-measurement state is*

$$\sum_{j=0}^{N-1} \sum_{l=0}^{2^{n_l}-1} \left|\alpha_{l|j}\right|^2 f(l) \left|0\right\rangle_{n_l} \left|u_j\right\rangle_{n_b}. \tag{263}$$

Proof. The state after the conditioned rotation has occurred is

$$\sum_{j=0}^{N-1} b_j \left(\sqrt{1-\frac{1}{\tilde{\lambda}_j^2}}\left|0\right\rangle + \frac{1}{\tilde{\lambda}_j}\left|1\right\rangle\right) \sum_{l=0}^{2^{n_l}-1} \alpha_{l|j} f(l) \left|l+\tilde{\lambda}_j\right\rangle_{n_l} \left|u_j\right\rangle_{n_b}. \tag{264}$$

We are going to analyse the effect of the inverse QPE on this state. For this analysis, we will ignore the case where the rotation has failed. Thus we work instead with the state

$$\left|\psi\right\rangle = \sum_{j=0}^{N-1} \frac{b_j}{\tilde{\lambda}_j} \sum_{l=0}^{2^{n_l}-1} \alpha_{l|j} f(l) \left|l+\tilde{\lambda}_j\right\rangle_{n_l} \left|u_j\right\rangle_{n_b}. \tag{265}$$

The reason we can do this is the following. Suppose we have a quantum state

$$\left|y\right\rangle = \{\text{good}\}\left|1\right\rangle\left|\text{good}\right\rangle\left|x\right\rangle_m + \{\text{bad}\}\left|0\right\rangle\left|\text{bad}\right\rangle\left|x\right\rangle_m. \tag{266}$$

Then we apply the inverse QPE on the second register, obtaining

$$\left(I \otimes \mathrm{QPE}^\dagger \otimes I_m\right) |y\rangle = \{\text{good}\} |1\rangle\, \mathrm{QPE}^\dagger |\text{good}\rangle |x\rangle_m + \{\text{bad}\} |0\rangle\, \mathrm{QPE}^\dagger |\text{bad}\rangle |x\rangle_m . \tag{267}$$

Finally, suppose we measure the first qubit in the computational basis and continue working with the post-measurement state only if the outcome is 1. Thus we only consider the post-measurement states of the form

$$\{\text{good}\} |1\rangle\, \mathrm{QPE}^\dagger |\text{good}\rangle |x\rangle_m . \tag{268}$$

Therefore, to calculate analytically Eq. 268 starting from $|y\rangle$, we can ignore the parts with a $|\text{bad}\rangle$ register appended.

Returning to $|\psi\rangle$ from Eq. 265, after applying the QFT on the eigenvalues register it becomes

$$\sum_{j=0}^{N-1} \sum_{l=0}^{2^{n_l}-1} \frac{\alpha_{l|j} f(l)}{2^{n_l/2}} \sum_{y=0}^{2^{n_l}-1} e^{2\pi i y \left(l+\tilde{\lambda}_j\right)/2^{n_l}} |y\rangle |u_j\rangle . \tag{269}$$

The next part of the inverse QPE is powers of e^{-iAt}, giving

$$\sum_{j=0}^{N-1} \sum_{l=0}^{2^{n_l}-1} \frac{\alpha_{l|j} f(l)}{2^{n_l/2}} \sum_{y=0}^{2^{n_l}-1} e^{2\pi i y \left(\frac{l+\tilde{\lambda}_j}{2^{n_l}} - \frac{\lambda_j t}{2\pi}\right)} |y\rangle |u_j\rangle . \tag{270}$$

Finally, applying $H^{\otimes n_l}$ yields

$$\sum_{j=0}^{N-1} \sum_{l=0}^{2^{n_l}-1} \frac{\alpha_{l|j} f(l)}{2^{n_l/2}} \sum_{y=0}^{2^{n_l}-1} \left(e^{2\pi i y \left(\frac{l+\tilde{\lambda}_j}{2^{n_l}} - \frac{\lambda_j t}{2\pi}\right)} \right)^y \sum_{h=0}^{2^{n_l/2}} (-1)^{y\cdot h} |h\rangle |u_j\rangle . \tag{271}$$

The state corresponding to $h = 0$ is then

$$\sum_{j=0}^{N-1} \sum_{l=0}^{2^{n_l}-1} \frac{\alpha_{l|j} f(l)}{2^{n_l/2}} \sum_{y=0}^{2^{n_l}-1} \left(e^{2\pi i y \left(\frac{l+\tilde{\lambda}_j}{2^{n_l}} - \frac{\lambda_j t}{2\pi}\right)} \right)^y |0\rangle |u_j\rangle \tag{272}$$

$$= . \sum_{j=0}^{N-1} \sum_{l=0}^{2^{n_l}-1} \alpha_{l|j} \alpha^*_{l|j} f(l) |0\rangle |u_j\rangle \tag{273}$$

$$= \sum_{j=0}^{N-1} \sum_{l=0}^{2^{n_l}-1} \left|\alpha_{l|j}\right|^2 f(l) |0\rangle |u_j\rangle . \tag{274}$$

□

We can now prove Theorem 9.1.

Proof. Let $|x\rangle$ be the exact solution to the system and let f defined as in Proposition 9.3. From Proposition 9.4,

$$\begin{aligned}
||x\rangle - \mathrm{HHL}\,(A, |b\rangle)| &= \left|\sum_{j=0}^{N-1} b_j \left(\frac{1}{\lambda_j} - \sum_{l=0}^{2^{n_l}-1} \left|\alpha_{l|j}\right|^2 f(l)\right)\right| |u_j\rangle && (275)\\
&= \left|\sum_{j=0}^{N-1} b_j \left(f(\lambda_j) - \sum_{l=0}^{2^{n_l}-1} \left|\alpha_{l|j}\right|^2 f(l)\right)\right| |u_j\rangle && (276)\\
&\leq \sum_{j=0}^{N-1} b_j \left|\left(f(\lambda_j) - \sum_{l=0}^{2^{n_l}-1} \left|\alpha_{l|j}\right|^2 f(l)\right)\right| |u_j\rangle && (277)\\
&= \sum_{j=1}^{N-1} b_j \mathcal{O}\left(\frac{tn_l 2^{n_l}}{\tilde{\lambda}_j^2}\right) \leq \sum_{j=1}^{N-1} b_j \mathcal{O}\left(\frac{tn_l 2^{n_l}}{\tilde{\lambda}_2^2}\right) && (278)\\
&\leq \mathcal{O}\left(\frac{tn_l 2^{n_l}}{\tilde{\lambda}_2^2}\right) \leq \mathcal{O}\left(\frac{tn_l 2^{n_l}}{\tilde{\lambda}_{\min}^2}\right) = \mathcal{O}\left(\frac{n_l}{\tilde{\lambda}_{\min}\lambda_{\min}}\right) && (279)\\
&= \mathcal{O}\left(\frac{n_l 2^{-n_l}}{\lambda_{\max}}\right) = \mathcal{O}\left(\log_2(1/\epsilon)\epsilon\right) =: \epsilon_H. && (280)
\end{aligned}$$

where we have used that $|b\rangle$ is normalised, the definition of t from Eq. 229, that $\tilde{\lambda}_{\min} \leq 2^{n_l}/\kappa$ and taken $n_l = \lfloor \log_2(1/\epsilon) \rfloor$.

Finally, from Lemma 5.3 and Eq. 177,

$$\begin{aligned}
&\left||x\rangle - \mathrm{HHL}\left(\tilde{A}, |\tilde{b}\rangle\right)\right| && (281)\\
&\leq \left||x\rangle - \mathrm{HHL}\left(A, |\tilde{b}\rangle\right)\right| + \left|\mathrm{HHL}\left(A, |\tilde{b}\rangle\right) - \mathrm{HHL}\left(\tilde{A}, |\tilde{b}\rangle\right)\right| && (282)\\
&= \mathcal{O}(\kappa c^2 + \epsilon_A + \epsilon_H). && (283)
\end{aligned}$$

□

In Eq. 280, a bound $\mathcal{O}(\epsilon)$ can be achieved by only inverting values which are larger than $1/\kappa$, as it is shown in Section 2 of [1].

10 Simulations

The algorithm was implemented using the Qiskit software from IBM ([42]). The results are simulations run locally, where the output is a vector array containing the theoretical amplitudes of the final state of the algorithm. In all plots cases, the matrix used was

$$A = \begin{pmatrix} 2 & 1/2 & & 0 \\ 1/2 & \ddots & \ddots & \\ & \ddots & & 1/2 \\ 0 & & 1/2 & 2 \end{pmatrix}, \tag{284}$$

which is well-conditioned. Throughout this section, keeping with the notation from the previous sections, the size of the matrix is 2^{n_b}, its condition number is κ; the size of the eigenvalues register is n_l; c is the small constant used in the Polinomial State Preparation algorithm, with polynomial p; m the initial number of Trotter steps; ϵ_H the error from the HHL algorithm assuming exact initial state and Hamiltonian simulation; ϵ_A the error from Hamiltonian simulation; $|x\rangle$ the exact solution to the system and $|\tilde{x}\rangle$ the solution returned by the HHL algorithm.

Polynomial State Preparation

Proposition 17 states that if $\left|\tilde{b}\right\rangle$ is the state prepared by the algorithm, and $|b\rangle$ the target state then

$$\left\| |b\rangle - \left|\tilde{b}\right\rangle \right\| = \mathcal{O}(c^2). \tag{285}$$

Figure 28a shows the error fixing the number of qubits and decreasing c in preparing the state for the polynomials

$$p_1(x) = x^4 - x^3 + x^2 - 1/2, \tag{286}$$
$$p_2(x) = \pi/2, \tag{287}$$
$$p_3(x) = \frac{3+\sqrt{109}}{80}x^2 - \frac{23+\sqrt{109}}{80}x - 1/8. \tag{288}$$

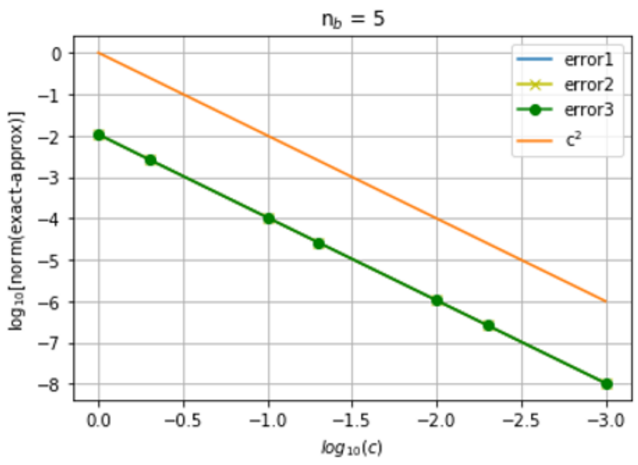
On the other hand, Figure 28b shows the error fixing c and varying the number of qubits in preparing the quantum state with amplitudes given by the polynomial

$$p_3(x) = \frac{3+\sqrt{109}}{80}x^2 - \frac{23+\sqrt{109}}{80}x - 1/8. \tag{289}$$

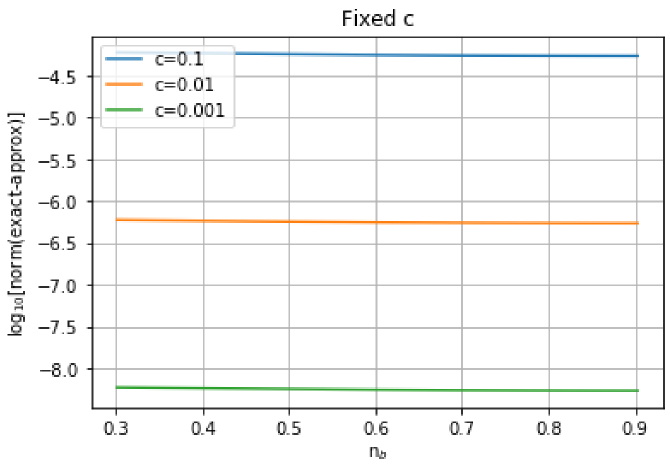
Lemma 5.4 shows that the probability of successfully preparing the state behaves as

$$\mathcal{O}\left(c^2 \|p\|^2_{L^2_{[0,1]}}\right). \tag{290}$$

Figure 29 shows the squared amplitude returned by Qiskit associated to the $|1\rangle$ qubit defined as a success flag for the algorithm.



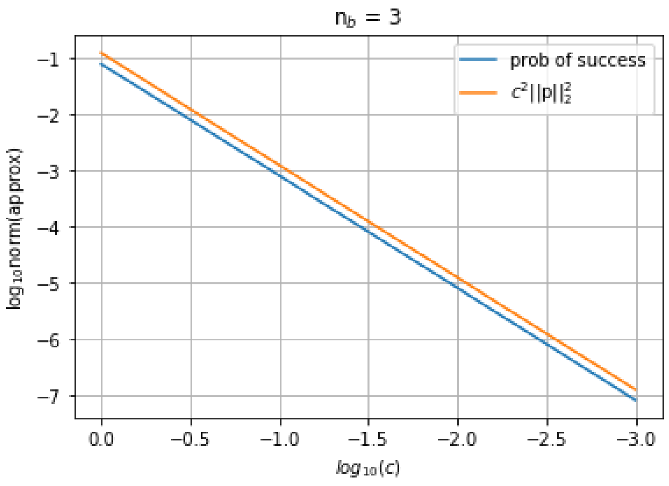
(a) Fixed n_b, varying c and plotted against c^2.



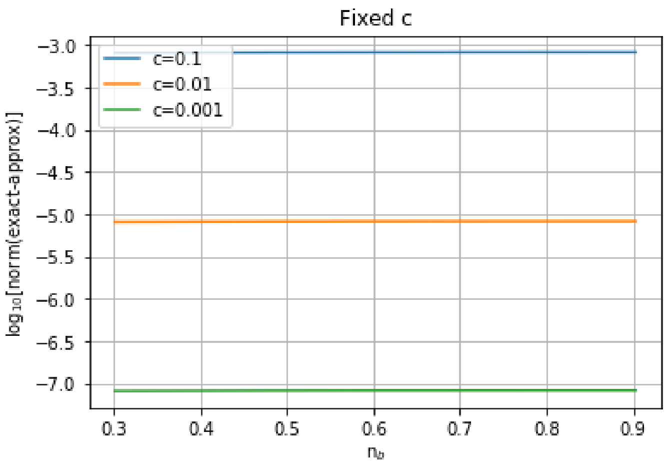
(b) Fixed c, vary the dimension of the state. The error is constant with the dimension.

Figure 28: Error in state preparation varying different parameters.

Lemma 5.3 states that the overall error in the HHL behaves as $\mathcal{O}\left(\kappa c^2 + \epsilon_H\right)$. In Figure 30, the full algorithm was run varying c and compared against the analytically calculated solution.



(a) Fixed n_b, varying c and plotted against $c^2\|p\|^2$.



(b) Fixed c, vary the dimension of the state. The probability is constant with the dimension.

Figure 29: Probability of measuring $|1\rangle$ in the flag-qubit associated with the state preparation.

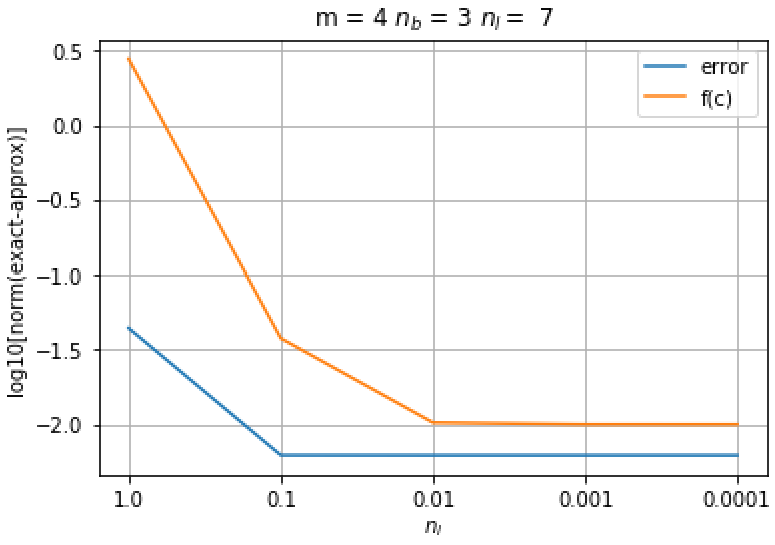


Figure 30: Overall error varying c for $f(c) = \kappa c^2 + \epsilon_H$.

Hamiltonian simulation

The definition of the norm of a matrix, as given in Definition 2.12, is

$$\|A\| = \max_{\||x\rangle\|=1} \|A\,|x\rangle\| \quad \text{for} A \in \mathbb{C}^{m\times n}, |x\rangle \in \mathbb{C}^{n\times 1}. \tag{291}$$

Therefore, to test the approximation obtained for the Hamiltonian simulation, one option is to run the algorithm with several unitary initial states which can be exactly prepared and plot the results.

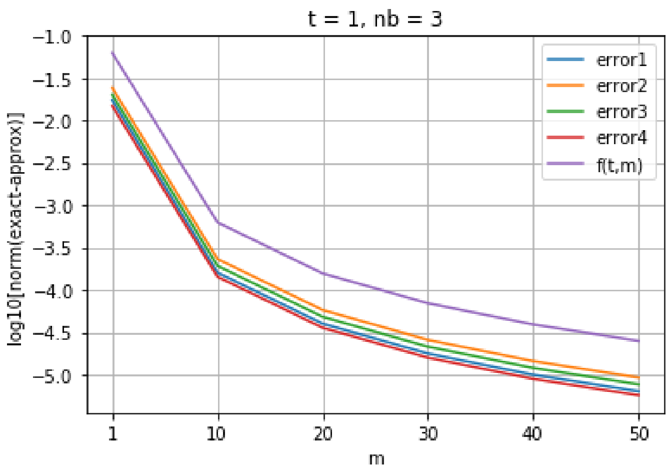
Lemma 6.2 stated that

$$\left\|V - \tilde{V}\right\| = \mathcal{O}\left(\frac{t^3 b^3}{2m^2}\right). \tag{292}$$

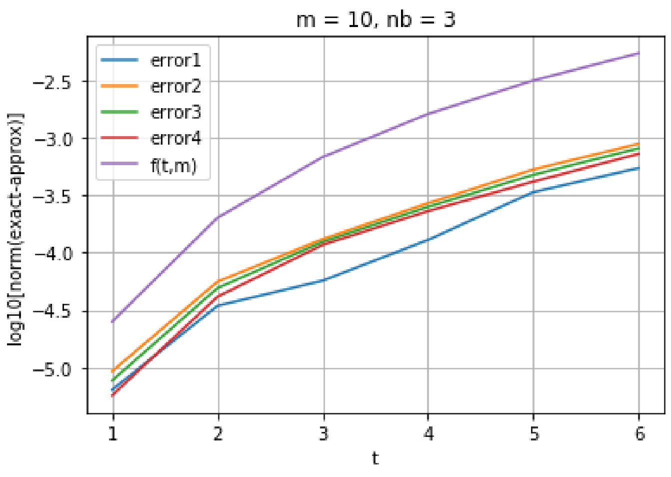
Figure 31 shows the results from calculating

$$\left\|V\,|b\rangle - \tilde{V}\,|b\rangle\right\|, \tag{293}$$

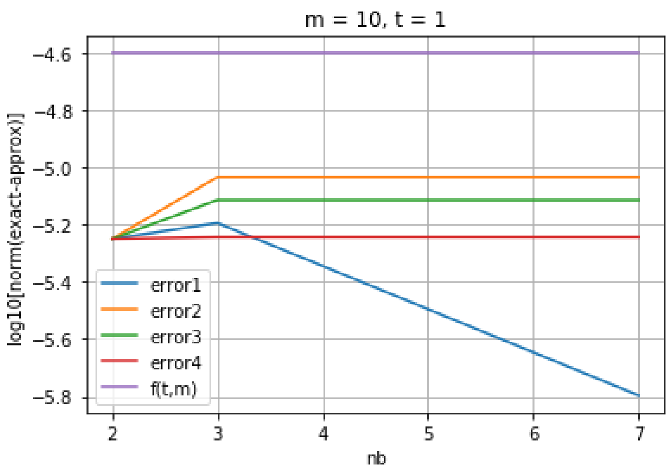
where $\tilde{V}$ is the Hamiltonian simulation from the algorithm and $V = e^{iAt}$, and plotting them against $f(t,m) = \frac{b^3 t^3}{2m^2}$.



(a) Vary m.



(b) Vary t.



(c) Vary n_b.

Figure 31: Error in Hamiltonian simulation varying different parameters. (c) shows that it is independent of the dimension. Here error1 seems to decrease because $|b_1\rangle = H^{\otimes nb}|0\rangle$, and so the vector components decrease with n_b.

Finally, in Lemma 6.3 it was proved that the overall error assuming an exact procedure for preparing the initial state would be in $\mathcal{O}(\epsilon_A + \epsilon_H)$. Thus, by only decreasing m and leaving everything fixed, we expect to see the error decrease at first and then constant dominated by ϵ_A. Figure 32 shows the simulation results.

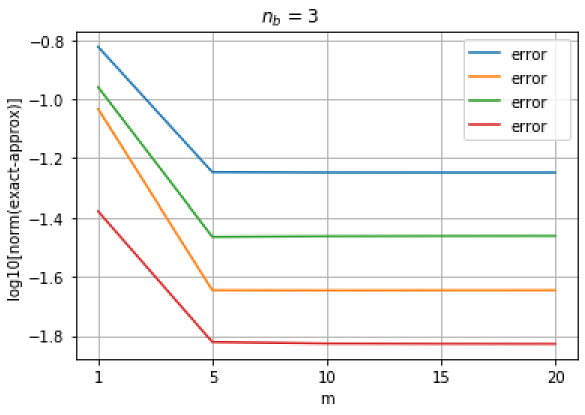


Figure 32: Overall error from the HHL varying m and with an exact initial state.

11 Outlook and Conclusion

In this thesis we have presented a quantum algorithm to solve systems of linear equations. We have achieved an implementation that runs simulations on large systems and non-trivial initial states. As an addition, we have extended the the idea from [45] to load data to a quantum computer via polynomials. The idea, which applied originally to 2-qubits, was extended for an arbitrary number of qubits and an arbitrary degree of the polynomial. Using the scheme to decompose Hamiltonians into a sum of 1-sparse matrices from [17], we have provided a means to simulate general symmetric tri-diagonal matrices. All the circuits for this stage of the algorithm are original. Finally, in this thesis we introduce an algorithm to compute specific functions of the solution without loosing the exponential runtime improvement.

Several paths could be explored to improve the construction of the algorithm. One option would be to investigate an optimal Hamiltonian simulation. Any improved implementation for this stage of the algorithm could be easily integrated and tested with the existing code, as explained in the Appendix.

During the course of this thesis it was considered to include the Richardson Extrapolation, a small description of the method and the findings are included in the Appendix. This is a direction which would be worth probing and which may allow a reduction in the overall complexity of the algorithm.

Further work on the inversion of eigenvalues would be to include the error of the approximating polynomial in the final analysis. In the HHL paper they only perform the inversion for larger values, however, in the given implementation we do not make such difference. Therefore, the overall performance could be improved by doing so and avoiding inverting values that are too close to zero.

Acknowledgements

First and foremost, I would like to thank Stefan Wöerner for his constant directions throughout these months, and for the always fruitful discussions. Secondly, I am grateful to Prof. Hiptmair for his regular calls and input of ideas. I would also like to acknowledge the Quantum Research group from IBM Zürich for their friendliness and for creating a comfortable environment to work, as well as for sharing their research on the weekly meetings. Finally, I would like to thank my family and friends for their support and encouragement throughout these years.

12 Appendix

Richardson Extrapolation

One way to speed up the computation is applying the Richardson extrapolation [43, Chapter-2.2.]. Suppose an algorithm $A(h)$ approximating A^* with error h^n, i.e. $A(h) = A^* + Ch^n + \mathcal{O}\left(h^{n+1}\right)$. And define the function

$$R(h,T) := \frac{T^n A(h/t) - A(h)}{T^n - 1}. \tag{294}$$

Then the Richardson extrapolation of $A(h)$ is

$$\begin{aligned} R(h,T) &:= \frac{T^n\left(A^* + Ch^n + \mathcal{O}\left(h^{n+1}\right)\right) - \left(A^* + Ch^n + \mathcal{O}\left(h^{n+1}\right)\right)}{T^n - 1} \\ &= A^* + \mathcal{O}\left(h^{n+1}\right) \end{aligned}. \tag{295}$$

used as a recurrence relation for even higher-order error estimations of A^*.

It can be used in conjunction with the HHL to reduce the parameter m and therefore the number of gates and time of the computation. Here we have $h = t/m$, and so for a one step Richardson extrapolation we could take $T = 2$ (that is to minimise the increase in m). Then we run the algorithm once with m and once with $2m$ and compute $R(t/m, 2)$ from the results obtained in the measurements.

This method was tested to calculate the norm of the solution and gave promising results, however no analysis of the total number of gates nor the error was carried out and that is why it has not been included in the main of the thesis.

Implementation

In this section we show how to use the code. To run the algorithm one needs to specify the dimension of the system, the coefficients of the matrix and the polynomial for the right hand side.

The code consists of the main HHL algorithm and circuit factories. The latter are subclasses of circuit factory, which has available methods for controlled, inverse and powers and is provided by Qiskit.

Suppose we want to prepare a state given by the polynomial $p(x) = a_0 + a_1x + \ldots + a_dx^d$. The PolynomialState circuit factory takes as input for the constructor method the array $[a_0, a_1, \ldots, a_d]$. Then the method ?build? is for constructing the circuit. The inputs are, in this order, the quantum circuit, the register where we want the state, a flag qubit and a list of ancillas. The following is the simplest version for building a state.

```
# State preparation. The first step is to put the register in a
    superposition of bases states.
for qu in qrb:
qc.h(qu)
# qrr[0] will be a flag. A measurement of 1 of this qubit means the
    procedure has been successful.
PolynomialState(self.px).build(qc, qrb, qrr[0], q_ancillas=qra)
```

The Polynomial State Preparation algorithm needs two important methods: get_controls method computes the possible k-tuples of qubits, $1 \leq k \leq \min(deg, n_b)$ and the get_thetas method computes the coefficient of each monomial, which will later be the angle of the rotation.

```
def get_controls(n, d):
   t = [0] * (n - 1) + [1]
   cdict = {tuple(t): 0}
   clist = list(product([0,1], repeat=n))
   index = 0
   while index < len(clist):
      tsum = 0
      i = clist[index]
      for j in i:
         tsum = tsum + j
      if tsum > d:
         clist.remove(i)
      else:
         index = index+1
      clist.remove(tuple([0]*n))
   # For now set all angles to 0
   for i in clist:
      cdict[i] = 0
   return cdict
def get_thetas(cdict, p, n):
   # For p1 to pd, we have pj*(q0+2q1+...+2^nqn)^j. Thus we calculate
       the coefficients
   for j in range(1,len(p)):
      # List of multinomial coefficients
      mlist = multinomial_coefficients(n, j)
      # Add angles
      for m in mlist:
         temp_t = []
         powers = 1
         # Get controls
         for k in range(0, len(m)):
         if m[k] > 0:
            temp_t.append(1)
            powers *= 2**(k*m[k])
         else:
            temp_t.append(0)
         temp_t = tuple(temp_t)
         # Add angle
         cdict[temp_t] += p[j]*mlist[m]*powers
   return cdict
```

The TridiagonalSimulator Circuit Factory approximates e^{iAt} using the Trotter formula from Eq. 150. It takes as constructor parameters the coefficients of the matrix, the number of qubits where it is acting and the number of Trotter steps.

Currently, the factories for simulating each element from the decomposition are encoded. However these could be taken as constructor parameters too if one wished to simulate different types of matrices or had more efficient implementations.

Below are the construction and build methods for the implementation of e^{iAt}.

```
class TridiagonalSimulator(CircuitFactory):
   """
   Simulates exp(iAt), where A is a tridiagonal symmetric matrix of
   dimension 2^n x 2^n. When no time is provided the simulation runs
   for t=1.
   """
   def __init__(self, number_of_qubits, m, a, b):
      super().__init__(number_of_qubits)
      self.m = m
      self.a = a
      self.b = b
      # Construct a1 and a23 factories
      self._a1_factory = A1Factory(number_of_qubits)
      self._a23_factory = A23Factory(number_of_qubits)
```

```
   def build(self, qc, q, q_ancillas=None, params=1):
      # Since A1 commutes, one application with time t*2^{j} to the last
      qubit is enough
      self._a1_factory.build(qc, q[0], params=self.a*params)
      # exp(iA2t/2m)
      qc.u3(self.b*params/self.m, -np.pi/2, np.pi/2, q[0])
         for _ in range(0, self.m):
            self._a23_factory.build(qc, q, q_ancillas=q_ancillas,
            params=self.b*params/self.m)
      # exp(-iA2t/2m)
      qc.u3(-self.b*params/self.m, -np.pi/2, np.pi/2, q[0])
```

The following is the build method for e^{iH_1t}.

```
   def build(self, qc, q, q_ancillas=None, params=1):
      qc.x(q) qc.u1(params, q)
      qc.x(q) qc.u1(params, q)
```

The following is the build method for $e^{iH_2t}e^{iH_3t}$ inside the bracket from Eq. 150.

```
def build(self, qc, q, q_ancillas=None, params=1):
# Gates for A2
qc.u3(-2*params, -np.pi/2, np.pi/2, q[0])
# Gates for A3
for i in range(0, self.num_target_qubits-1):
q_controls = []
qc.cx(q[i], q[i+1])
q_controls.append(q[i+1])
# Now we want controlled by 0
qc.x(q[i])
for j in range(i, 0, -1):
```

```
qc.cx(q[i], q[j-1])
q_controls.append(q[j-1])
qc.x(q[i])
# Multicontrolled x rotation
if(len(q_controls)>1):
qc = cn_gate(q_controls, qc, q_ancillas, -np.pi/2, np.pi/2,
-2*params, q[i])
else:
qc.cu3(-2*params, -np.pi/2, np.pi/2, q_controls[0], q[i])
# Uncompute
qc.x(q[i])
for j in range(0, i):
58qc.cx(q[i], q[j])
qc.x(q[i])
qc.cx(q[i], q[i+1])
```

Finally, below is the code to implement the observables from Section 9.

```
# This observable gives the squared average of the entries on condition
    seing |0>
def observable_average(self, qc, qr):
   for q in qr:
      qc.h(q)
   return qc

# kth iteration of the observable <x|A|x>
def observable_func(self, qc, qr, k):
   for i in range(0,k-1):
      qc.cx(qr[k-1], q[i])
   qc.h(qr[k-1])
   return qc
```

References

[1] A. W. Harrow, A. Hassidim, S. Lloyd. *Quantum algorithm for solving linear systems of equations.*
`arXiv:0811.3171 [quant-ph]`(2009)

[2] C. D. Meyer. *Matrix Analysis and Applied Linear Algebra.*
`Society for Industrial and Applied Mathematics.`(2000)

[3] M. A. Nilsen, I. L. Chuang. *Quantum Computation and Quantum Information.*
`Cambridge University Press, Cambridge.`(2010)

[4] A. Quarteroni, R. Sacco, F. Saleri. *Numerical Mathematics.*
`Springer.`(2010)

[5] C. H. Bennett. *Logical reversibility of computation.*
`IBM J. Res. Develop.,17(1973), pp. 525-532.`

[6] G. Nannicini. *An Introduction to Quantum Computing, Without the Physics.*
`arXiv:1708.03684 [cs.DM]`(2017)

[7] Lov Grover, Terry Rudolph. *Creating superpositions that correspond to efficiently integrable probability distributions.*
`arXiv:quant-ph/0208112`

[8] A. M. Childs, R. Kothari, R. D. Somma. *Quantum algorithm for systems of linear equations with exponentially improved dependence on precision.*
`arXiv:1511.02306 [quant-ph]`(2017).

[9] L. Wossnig, Z. Zhao, A. Prakash. *A quantum linear system algorithm for dense matrices.*
`arXiv:1704.06174 [quant-ph]`(2017).

[10] B. D. Clader, B. C. Jacobs, C. R. Sprouse. *Preconditioned quantum linear system algorithm.*
`arXiv:1301.2340 [quant-ph]`(2013).

[11] Y. Lee, J. Joo, S. Lee. *Hybrid quantum linear equation algorithm and its experimental test on IBM Quantum Experience.*
`arXiv:1807.10651 [quant-ph]`(2018).

[12] A. N. Soklakov, R. Schack. *Efficient state preparation for a register of quantum bits.*
`arXiv:quant-ph/0408045`(2006).

[13] P. Kaye, M. Mosca. *Quantum Networks for Generating Arbitrary Quantum States.*
`arXiv:quant-ph/0407102`(2004).

[14] V. V. Shende, S. S. Bullock, I. L. Markov. *Synthesis of Quantum Logic Circuits.*
`arXiv:quant-ph/0406176`(2006).

[15] M. Mottonen, J. J. Vartiainen, V. Bergholm, M. M. Salomaa. *Transformation of quantum states using uniformly controlled rotations.*
`arXiv:quant-ph/0407010`(2008).

[16] M. Plesch, C. Brukner. *Quantum-state preparation with universal gate decompositions.*
`arXiv:1003.5760 [quant-ph]`(2011).

[17] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, D. A. Spielman. *Exponential algorithmic speedup by quantum walk.*
`arXiv:quant-ph/0209131`(2002).

[18] G. Ahokas. *Improved algorithms for approximate quantum Fourier transforms and sparse Hamiltonian simulations.*
`M.Sc. Thesis, University of Calgary, 2004.`

[19] D. W. Berry, G. Ahokas, R. Cleve, B. C. Sanders. *Efficient quantum algorithms for simulating sparse Hamiltonians.*
`arXiv:quant-ph/0508139`(2006).

[20] A. M. Childs, N. Wiebe. *Hamiltonian Simulation Using Linear Combinations of Unitary Operations.*
`arXiv:1202.5822 [quant-ph]`(2012).

[21] A. M. Childs. *On the relationship between continuous- and discrete-time quantum walk.*
`arXiv:0810.0312 [quant-ph]`(2009).

[22] A. M. Childs, R. Kothari. *Simulating sparse Hamiltonians with star decompositions.*
`arXiv:1003.3683 [quant-ph]`(2010).

[23] D. W. Berry, A. M. Childs. *Black-box Hamiltonian simulation and unitary implementation.*
`arXiv:0910.4157 [quant-ph]`(2011).

[24] S. Endo, Q. Zhao, Y. Li, S. Benjamin, X. Yuan. *Mitigating algorithmic errors in Hamiltonian simulation.*
`arXiv:1808.03623 [quant-ph]`(2018).

[25] N. Wiebe, M. Roetteler. *Quantum arithmetic and numerical analysis using Repeat-Until-Success circuits.*
`arXiv:1406.2040 [quant-ph]`(2014).

[26] V. Vedral, A. Barenco, A. Ekert. *Quantum Networks for Elementary Arithmetic Operations.*
`arXiv:quant-ph/9511018`(1995).

[27] K. Mitarai, M. Kitagawa, K. Fujii. *Quantum Analog-Digital Conversion.*
`arXiv:1805.11250 [quant-ph]`(2018).

[28] F. Schmueser, D. Janzing. *Quantum analog-to-digital and digital-to-analog conversion.*
`Phys. Rev. A 72, 042324`(2005).

[29] A. Patel, A. Priyadarsini. *Optimisation of Quantum Hamiltonian Evolution: From Two Projection Operators to Local Hamiltonians.*
`arXiv:1503.01755 [quant-ph]`(2017).

[30] A. Patel, A. Priyadarsini. *Efficient Quantum Algorithms for State Measurement and Linear Algebra Applications.*
`arXiv:1710.01984 [quant-ph]`(2018).

[31] M. Grote, T. Huckle. *Black-box Hamiltonian simulation and unitary implementation.*
`Parallel Preconditioning with Sparse Approximate Inverses.`
`SIAM J. Sci. Comput. 18, 838.`(1997).

[32] W. Rudin. *Real and Complex Analysis.*
`McGraw-Hill International Editions.`(1987).

[33] D. Dervovic, M. Herbster, P. Mountney, S. Severini, N. Usher, L. Wossnig *Quantum linear systems algorithms: a primer.*
`arXiv:1802.08227 [quant-ph].`(2018).

[34] W. Rudin. *Principles of Mathematical Analysis.*
`3d ed., McGraw-Hill Book Company, New York.`(1976).

[35] R. Diestel. *Graph Theory.*
`Graduate Texts in Mathematics, Springer.`(2017).

[36] A. Montanaro, S. Pallister. *Quantum algorithms and the finite element method.*
`arXiv:1512.05903 [quant-ph].`(2016).

[37] T. Häner, M. Roetteler, K. M. Svore. *Optimizing Quantum Circuits for Arithmetic.*
`arXiv:1805.12445 [quant-ph].`(2018).

[38] B. Amento, R. Steinwandt, M. Roetteler. *Efficient quantum circuits for binary elliptic curve arithmetic: reducing T-gate complexity.*
`arXiv:1209.6348 [quant-ph].`(2012).

[39] P. Kaye and C. Zalka. *Optimized quantum implementation of elliptic curve arithmetic over binary fields.*
`arXiv:quant-ph/0407095v1.`(2004).

[40] D. Maslov, J. Mathew, D. Cheung, D. K. Pradhan. *On the Design and Optimization of a Quantum Polynomial-Time Attack on Elliptic Curve Cryptography.*
`arXiv:0710.1093 [quant-ph].`(2009).

[41] T. Häner, M. Soeken, M. Roetteler, K. M. Svore. *Quantum circuits for floating-point arithmetic.*
`arXiv:1807.02023 [quant-ph].`(2018).

[42] Qiskit Aqua v0.3.0 (2018).
`https://qiskit.org/aqua.`

[43] C. Brezinski, M. R. Zaglia. *Extrapolation Methods. Theory and Practice.* `North-Holland.`(1991).

[44] D. C. Lay. *Linear Algebra and Its Applications.* `Pearson.`(2012).

[45] S. Woerner, D. J. Egger. *Quantum Risk Analysis.* `arXiv:1806.06893 [quant-ph].`(2018).

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

QUANTUM ALGORITHM FOR SOLVING TRI-DIAGONAL LINEAR SYSTEMS OF EQUATIONS

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Carrera Vazquez	Almudena

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Rüschlikon, 27 November 2018	[illegible]

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.