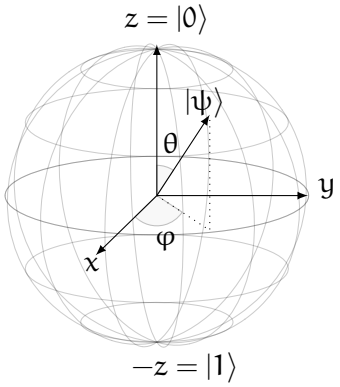
Report
Simulation Sciences Seminar

How to solve a linear system of equations using a quantum computer

Philipp Schleich
Matricel number: 391779

July 11, 2019



Supervisor: Prof. Dr. Benjamin Stamm

MATH CCES Center for Computational Engineering Science Mathematics Division | RWTH AACHEN UNIVERSITY

Abstract

This seminar report gives a short summary on the capababilty of quantum computers to solve linear systems of equations. An introduction in quantum computing will be given before discussing the building blocks of the HHL algorithm proposed by Harrow, Hassadim and Lloyd in 2009. At last, it will be discussed on a 4x4 system, and a outlook on advanced topics like the error analysis on it will be provided.

Contents

1 Introduction

Solving linear systems of equations is task that arises in all areas of science, engineering or economics. It is one of the tasks, that are known for a long time, well understood, but still tedious to solve, especially for large systems. Usually, the goal is to increasingly larger as this in some manner gives more insight in the underlying problem. When solving partial differential equations, this means better accuracy, when dealing with machine learning applications this allows to use more data to improve the model. But solving these problems typically has cubic complexity (Gauss-Elimination), in the best cases it is linear (Conjugate Gradient for symmetric positive definite matrices). Thus, if one wants to go to even larger amounts of data, larger systems, one might want to look into a different direction.

Quantum computing is currently one of the fastest changing fields with a lot of potential, that might disrupt the landscape of computing as of today. It promises a lot of fancy, yet almost magic features such as teleportation of information. On the other hand, one of the biggest advantages of quantum computers is the capability to store and operate on exponentially more data than the units of information it consists of. Therefore, it might be a promising candidate to approach these high-dimensional problems. Apart from that, it has the potential to be a game-changer in artificial intelligence or computational quantum physics/chemistry.

In the following, a short introduction into quantum mechanics aswell as quantum computing will be given. Then, the linear system that we want to solve is introduced, and the conceptual idea of the algorithm by Harrow, Hassadim and Lloyd [HHL09] is stated. Afterwards, the subroutines necessary to implement the algorithm on a quantum computer are explained and assembled to the HHL algorithm. Finally, an application to a very specific linear system is discussed. The report is closed by a few additional remarks on further topics and a conclusion.

2 Basic concepts of quantum mechanics and quantum computing

In order to be able to study algorithms for quantum computers, one needs to be familiar with certain concepts in quantum mechanics and their extension on quantum computing.

2.1 Quantum mechanics

At first, some technicalities of quantum systems will be introduced before discussing three of the postulates of quantum mechanics. The latter will then allow to make the transition to quantum computing.

Quantum mechanics describes our world in a framework that holds also on the nanoscale. Meaning, quantum mechanics allows us to study the behaviour of building blocks of matter, such as molecules, atoms, nuclei and elementary particles.

In a quantum system, the particles obey the Schrödinger equation, which was postulated in 1925 by Erwin Schrödinger:

$$i\hbar\frac{\partial}{\partial t}\psi = \check{H}\psi \tag{1}$$

where $i^2 = -1$ stands for the imaginary unit, $\hbar$ denotes the so called Planck's constant, $\check{H}$ represents the Hamiltonian operator (which is a respresentation for the system's total energy) and ψ is the wavefunction. The latter is a function of space and time $\psi(x_1, \ldots, x_i, , \ldots, x_N, ; t)$ and describes the state of a N-particle system.

As the Schrödinger equation (1) is linear, it allows several interesting properties, of which the most important for the problem dealt with here is superposition. Thus, if ψ_1, ψ_2 solve (1), so does $\alpha\psi_1 + \beta\psi_2$, $\alpha, \beta \in \mathbb{C}$.

Further, we shall introduce bracket or Dirac notation, where $|\psi\rangle$ denotes our wavefunction and $\langle\psi|$ a linear functional, and we define $\langle\psi|\phi\rangle$ as the inner product between the two states.

$$\langle\psi|\phi\rangle = \int_{\mathbb{R}^{3N}} \psi^*\phi \, dx_1 \ldots dx_N \tag{2}$$

The so called Born-rule states, that $\langle\psi|\psi\rangle = \int_{\mathbb{R}^{3N}} |\psi(\mathbf{x};t)|^2 d\mathbf{x}_1 \dots d\mathbf{x}_N$ describes the probability of finding our N-particle system in state $|\psi\rangle$ given $\mathbf{x}$ and t. This requires that $\langle\psi|\psi\rangle = 1$ and thus, ψ has to be square-integrable, meaning $\psi \in L^2(\mathbb{R}^{3N};\mathbb{C})$. As the Hamiltonian operator $\check{H}$ in (1) typically contains a second derivative, one usually requires $\psi \in H^1(\mathbb{R}^{3N};\mathbb{C})$ s.th. the derivative is defined at least in the weak sense. This is also called the form domain of $\check{H}$. To be more precise, as one is looking for normalized functions with $\langle\psi|\psi\rangle = 1$, these are equivalence classes with the equivalence relation $\psi \sim \alpha\psi$, for some $\alpha \in \mathbb{C}\backslash\{0\}$.

This can all be translated into a discrete setting. Then, $|\psi\rangle$ do not anymore represent functions but vectors, which now live in a finite-dimensional vector space, that is still a Hilbert space (more about that in first postulate). The adjoint of $|\psi\rangle$ now is the adjoint vector, $\langle\psi| = \psi^\dagger = (\psi^*)^\mathsf{T}$. The inner product $\langle\psi|\phi\rangle$ thus reads $\psi^\dagger\phi = \sum_i \psi_i^*\phi_i$. [Kib18; NC10]

Now, three postulates of quantum mechanics[1] (following [NC10]) will be introduced, as we will rely on them later when precising what a physical representation of a quantum computer is exptected to fulfil.

1. Any isolated physical system has an associated complex, complete linear space equipped with an inner product (i.e. a Hilbert space). This space is called *state space*. A system is thus described by its *state vector* (or *state function* in the continuous case). This vector is a unit vector in the associated space (cf. Born-rule). Also, superposition of states as mentioned above stems from this postulate (linearity of the associated space).

2. Unitary transformations describe the evolution of closed quantum systems. That means, $|\psi(t)\rangle = U|\psi(t')\rangle$, where U depends only on t, t'. In this case, U is also called the time evolution operator or propagator. This postulate also implies that closed quantum systems obey the Schrödinger equation (equation (1)). To see this, remark that $e^{-\frac{i}{\hbar}\check{H}t}$ is unitary for hermitian $\check{H}$, where t' from above is chosen $t' = 0$.

[1] There are more postulates, though these three are the most important for the concept of quantum computing.

3. Measurements in quantum mechanics are described by a set of M_m *measurement operators*, which act on the state space. There is usually a discrete set of possible outcomes, where m refers to an element of this set. If the state is $|\psi\rangle$, then the probability to obtain outcome m in a measurement writes

$$p_m = \langle\psi| M_m^\dagger M_m |\psi\rangle \tag{3}$$

Obviously, the measurement operators have to satisfy the so called completeness relation, $\sum_m M_m^\dagger M_m = \mathbb{I}$, $\mathbb{I}$ being the identity, as the sum of the probabilty over all m outcomes has to sum up to one. This postulate also implies the expectation value of some hermitian operator E (usually called observables, i.e. measurable properties of a quantum system such as location, momentum, kinetic energy, total energy), as $M_m^\dagger M_m$ is hermitian. Say, we have a quantum system that can take the states $|0\rangle$ and $|1\rangle$ in the observable E. Postulate 1 allows therefore $|\psi\rangle = a|0\rangle + b|1\rangle$, where $|a|^2 + |b|^2 = 1$ ($a, b \in \mathbb{C}$). This leads to an expectation value of E as $\langle E\rangle = \langle\psi| E |\psi\rangle$. The probabilities for the individual measurement outcomes then can be expressed as $p(0) = |a|^2$ and $p(1) = |b|^2$. It is important here to state, that, when measuring this quantum system, the outcome is either 0 or 1 in a probabilistic manner. That being said, throughout measurement, a quantum system collapses out of the superposition in the measured state. Thus, the measurement operators are not unitary, as the impact of measurement cannot be reversed.

From here on, every physical constant (such as $\hbar$) is set to 1.

2.2 Quantum computing

We shall now extend what we know about quantum mechanics to the field of quantum information or quantum computing. A computer is a machine, that processes and stores information. Talking about a quantum computer, this machine represents a certain state, manipulates its time evolution and in the end, gives the user the possibility to take a look at the outcome (or in scientific terms, measure it), while underlying the laws of quantum mechanics. These three general properties one requires a computer to fulfill can be directly translated to a quantum version of such a machine by means of the three postulates introduced before.

Nielsen and Chuang ([NC10]) name in consequence some key properties a quantum computer should fulfill:

- Robust representation of quantum information (first postulate)
- Cabability to perform universal family of unitary transformations (second postulate)
- Preparation of a fiducial initial state
- Measurement of the output

The importance of the first, the third and the last point are quite obvious. In order to work with a quantum machine, it is important, that the information it encapsulates does not underly any unwanted influence from the surroundings[2], that the input state is known and that we can in some form work with the result of a computation. The second point may not be that obvious, might not be necessary for simple physical realization, but is crucial when it comes to extended use: One wants a computer to be able to do *any* possible calculation, and represented by a sequence of unitary operators. The goal is to have a computer being able to perform a finite number of unitary transformations (which in physical realization correspond so some building block), which then has the consequence that one only needs a finite number of building blocks that is able to perform any possible calculation (that can be represented by unitary transformations). These individual building blocks are called quantum gates, which each represent a certain unitary transformation.

[2] Readers interested in further topics of quantum information may want to look into quantum decoherence

There exist *universal* families of quantum gates[3], that allow, to write an arbitrary unitary operation U up to an accuracy ϵ in terms of a sequence of a finite number L of members of an universal family F. That being said, a set of quantum gates F is universal, if there exists a finite $L \in \mathbb{N}$, such that for any unitary operator U $\|U - U_L^F U_{L-1}^F \dots U_1^F\| \leq \epsilon$ with some $\epsilon > 0$. The significancy of this is revealed by the fact, that $L = \mathcal{O}(\log^2(\epsilon^{-1}))$, and therefore exponential accuracy is obtained by using only a polynomial number of gates (Solovay-Kitaev theorem, e.g. [NC10].

Apart from that, it is worth mentioning, that any computation feasible on a classical machine is also realizable on quantum computers. Intuitively, this is possible, as the *Toffoli*-gate, which is universal on classical computers, has a direct quantum equivalent. [NC10]

2.2.1 Qubits

Information on classical computers is stored by means of a so called *bit* as the fundamental unit of information, while a bit system can have a value $x \in \{0, 1\}$, which forms the computational basis. Any integer k can thus be represented in form of a bit string $\sum_{i=0}^{N} k_i 2^i$, whereas a real numbers can be represented up to some decimal precision by adding an integer with some $\frac{j}{2^m} = 0.j_{n-m+1} \dots j_n$. In comparison to that, on quantum computers, the *qubit* is the fundamental unit of information, possible values being $|0\rangle$ and $|1\rangle$. Thus, $\{|0\rangle, |1\rangle\}$ have to form an orthogonal basis in the associated Hilbert space, $\mathbb{C}^2$. Due to the quantum nature of a qubit, it can additionaly take any superposition of these 2 values, with the restriction that it is normalized, meaning $|\psi\rangle = a|0\rangle + b|1\rangle$, while $|a|^2 + |b|^2 = 1$, $a, b \in \mathbb{C}$. Again, to be more precise, $|\psi\rangle$ is from the subspace of $\mathbb{C}^2$ modulo the equivalence relation $(a, b) \sim (\alpha a, \alpha b)$ for some nonzero, complex α. As qubits are finite-dimensional, there also exits a vector form:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, |\psi\rangle = a|0\rangle + b|1\rangle = \begin{pmatrix} a \\ b \end{pmatrix}. \tag{4}$$

[3] We shall not worry about how these look here.

It is also possible, analogeous to the classical bit-string, to represent numbers on a quantum computer, which then of course also requires more than one qubit. Multiple qubits are combined via the tensor product, meaning the joint state of $|\phi\rangle$, $|\psi\rangle$ reads as $|\phi\rangle \otimes |\psi\rangle = |\phi\rangle\,|\psi\rangle$, while the last expression is introduced as short notation. Observing that $|\phi\rangle = a\,|0\rangle + b\,|1\rangle$ and $|\psi\rangle = c\,|0\rangle + d\,|1\rangle$,

$$\begin{aligned} |\phi\rangle \otimes |\psi\rangle &= ac\,|0\rangle \otimes |0\rangle + ad\,|0\rangle \otimes |1\rangle + bc\,|1\rangle \otimes |0\rangle + bd\,|1\rangle \otimes |1\rangle \\ &= ac\,|00\rangle + ad\,|01\rangle + bc\,|10\rangle + bd\,|11\rangle\,. \end{aligned} \tag{5}$$

Here, $\{|0\rangle \otimes |0\rangle\,, |0\rangle \otimes |1\rangle\,, |1\rangle \otimes |0\rangle\,, |1\rangle \otimes |1\rangle\}$ form an orthonormal basis of $\mathbb{C}^4 \sim \mathbb{C}^2 \otimes \mathbb{C}^2$, corresponding to the computational basis of a 2-qubit system. Introducing vector notation, the same expression can be rewritten to

$$|\phi\rangle\,|\psi\rangle = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}. \tag{6}$$

Extending this to even more qubits, it is easy to see that N qubits live in a 2^N-dimensional Hilbert space, $\mathbb{C}^{2^N} \sim \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2$, and thus can represent 2^N units of information. This is exponentially more than a classical computer can process with the same number of bits. But one has to emphasize, that a quantum computer outperforms a conventional one only in processing information. If one is interested in reading out this information, measurement collapses each qubit in one certain state and N qubits finally only give an output of an N-dimensional bitstring, similar to classical computing. Consider an integer $k = \sum_{l=1}^{2^n} 2^{n-l} k_l$. As a bit-string this writes

$$k = (k_1 k_2 \ldots k_n), \text{where} k_l \in \{0, 1\}. \tag{7}$$

The quantum notation is simply

$$|k\rangle = |k_1 k_2 \ldots k_n\rangle\,. \tag{8}$$

As an example, $k = 3 = 2^1 + 2^0 = (11)$, with its quantum complement $|3\rangle = |11\rangle$. Decimal numbers are can be represented by division by some approbiate power of 2:

$$\frac{k}{2^m} = 0.k_{n-m+1} \ldots k_n. \tag{9}$$

At this point, the concept of entanglement will be explained briefly on a system of two qubits. Let $|\psi_A\rangle \in \mathcal{H}_A$ and $|\psi_B\rangle \in \mathcal{H}_B$ from two Hilbert spaces $\mathcal{H}_A, \mathcal{H}_B$. In the case of qubit states, these both are $\mathbb{C}^2$. The system's Hilbert space then becomes $\mathcal{H}_A \otimes \mathcal{H}_B \sim \mathbb{C}^4$, with the basis $\{|0\rangle_A \otimes |0\rangle_B, |0\rangle_A \otimes |1\rangle_B, |1\rangle_A \otimes |0\rangle_B, |1\rangle_A \otimes |1\rangle_B\}$. A state is called factored or separable, if $|\psi\rangle$ can be expressed as $|\psi\rangle = |\psi_A\rangle \otimes |\psi_B\rangle$, meaning is can be decomposed into terms from separate Hilbert spaces, which are combined via the tensor product. If that is not possible, they are called entangled. An example for a factored state is $|0\rangle_A \otimes \frac{1}{\sqrt{2}}(|0\rangle_B + |1\rangle_B)$. A quite famous set of entangled states are the Bell states, of which the first one is $\frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B + |1\rangle_A \otimes |0\rangle_B)$. Multiple-qubit systems represent in most cases entangeled states.

Now, again only a single qubit will be considered, as this will enable us to give a nice intuition for qubits and a smooth transition to manipulating qubits by qugates. A single qubit state $|\psi\rangle$ can be represented by (x, y, z, t) on the unit sphere S^3 in $\mathbb{R}^4$:

$$\mathbb{C}^2 \to S^3 : a|0\rangle + b|1\rangle \mapsto (x, y, z, t), \tag{10}$$

while $a = x + iy$ and $b = z + it$. Furthermore, it is possible to depict (x, y, z, t) as a point of the sphere S^2 in $\mathbb{R}^3$ expressed by $(r = 1, \theta, \varphi)$. Indeed, up to a global phase[4] , $|\psi\rangle$ can be written as

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle, 0 \leq \theta \leq \pi, 0 \leq \varphi \leq 2\pi. \tag{11}$$

Remember from quantum mechanics, that a state can always only be defined up to a global phase, which means we can discard this. The sphere S^2 is usually called *Bloch sphere*. Mathematically, the transition to the Bloch sphere can be identified with the complex projective line or a Hopf-fibration from the three- to the two-sphere. [Kib18]

Now, if one wants to change the state of this qubit by means of unitary transformations, this can be represented as a rotation around an axis in the Bloch sphere. For instance, if $|\psi\rangle = |0\rangle$, this can be transformed into a $|1\rangle$ by a rotation around x by π. This is further discussed in the next section.
[Der+18; EHI00; NC10]

[4] Here, we do not talk about the parameter φ. This is a local phase and necessary to describe the state.

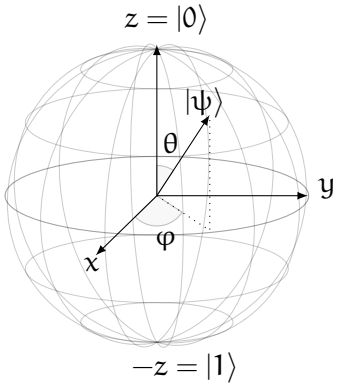


Figure 1: Bloch sphere

2.2.2 Quantum circuits

Being familiar with qubits, it is no possible to take a step further and look at how one can manipulate their states in order to perform computations. In accordance to classical gates, these parts are calles *qugates* on quantum computers. Conventional computers perform arbitrary computations using a finite sequence of logical gates, such as AND, OR, XOR. Qugates represent unitary transformations on a quantum state, and thus enable the user to evolve the quantum system that is represented by the computers *qubits*.

Single-qubit gates To begin, consider a one-qubit system (figure 1). As already mentioned, unitary transforms on a single qubit can be represented by rotations around an axis in the Bloch sphere. This again can be decomposed into single rotations around the three axes x, y, z. It turns out, that the Pauli matrices, which for instance arise when describing the spin of quantum-mechanical particles, exactly correspond to rotations around these axes by $\pi/2$:

$$\sigma_x = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_z = Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{12}$$

One can assure oneself, that this holds, if one applies the Pauli matrices to the expressions from equation (4) and writes the state as in equation (11).

The Pauli gates can aswell be used to implement rotations of an arbitrary angle. To see this, remark that $R_i(\theta) = \exp(-i\theta\sigma_i)$. As an example, using X in the previous expression yields

$$\exp\left(-i\theta\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\right) = \mathbb{I}_2 - i\theta X - \frac{\theta^2}{2}\mathbb{I}_2 - i\frac{\theta^3}{6}X + \cdots = \tag{13}$$
$$= \cos\theta\mathbb{I} - i\sin\theta X = \begin{pmatrix}\cos\theta & -i\sin\theta\\ -i\sin\theta & \cos\theta\end{pmatrix} \equiv R_x(\theta).$$

Moreover, recognize, that the quantum X-gate corresponds to the classical NOT-gate. Simply apply X in matrix-form on a state $|\psi\rangle$, which gives $|0\rangle \xrightarrow{X} |1\rangle$ and $|1\rangle \xrightarrow{X} |0\rangle$.

Another important gate is the *Hadamard*-gate, which represents the so called Hadamard-transform. It can be represented in matrix-form as

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}. \tag{14}$$

Elementary algebra reveals the following relations:

$$H|0\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \text{ and } H|+\rangle = |0\rangle \tag{15}$$
$$H|1\rangle = |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \text{ and } H|-\rangle = |1\rangle. \tag{16}$$

Remark that an additional application of the Hadamard transform $H(H|\psi\rangle) = H^2|\psi\rangle = \mathbb{I}|\psi\rangle$ returns the initial state. The Hadamard transform is closely related to the Fourier transform, whose quantum counterpart will be introduced later.

Apart from the gates introduced above, there is a whole zoo of single-qubit operations. But as they basically all only represent some rotation around an axis in the Bloch sphere, and as for the means understanding how to solve a linear system on a quantum computer, the already mentioned gates are sufficient. [Kib18; NC10]

Controlled unitary operations Often, it is necessary, that certain operations only take place, *if* some condition is fulfilled. Quantum information allows such a construct in the form of so called *controlled* or *conditional* gates.

First, take a look at a classical way to express this:

```
if (expression = true):
        do work
else if (expression = false):
        do nothing
```

This can be translated to quantum conditioned gates, with a subtle difference. Whereas on in conventional computing, the work is either done or not done at all, a controlled unitary gate acts only on the $|1\rangle$-part of a state, as superpositions are possible. As a first example, the CNOT-gate is introduced, which leaves the $|0\rangle$-part as is and applies a NOT-operation on the $|1\rangle$-part. In matrix notation for the previously defined computational basis of a two-qubit system, this reads

$$C_X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{17}$$

which can be recognized as a permutation matrix that swaps the lower entries when acting on a vector. And indeed, when applying (17) on the two-qubit system in (5) and (6), one obtains

$$C_X |\phi\psi\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bd \\ bc \end{pmatrix} = ac\,|00\rangle + ad\,|01\rangle + bd\,|10\rangle + bc\,|11\rangle\,. \tag{18}$$

This means, C_X acts conditioned on $|\phi\rangle$. If 0, corresponding to the a-part, then nothing happens. If 1, which corresponds to b, then the X-gate is applied on $|\psi\rangle$.

As a second example, controlled y-rotation is discussed. Recall that a y-rotation around an arbitrary angle $\theta \in \mathbb{R}$ can be expressed as $R_y(\theta) = \exp(-i\theta\sigma_y)$.

$$R_y(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{19}$$

We denote the conditioned y-rotation around an angle $\tilde{\theta}$ as $U_{\tilde{\theta}}$, which is the d-finite approximation[5] of θ:

$$\begin{aligned} U_{\tilde{\theta}} : |\tilde{\theta}\rangle |0\rangle &\mapsto |\tilde{\theta}\rangle \left(\cos\tilde{\theta}\,|0\rangle + \sin\tilde{\theta}\,|1\rangle\right) \\ |\tilde{\theta}\rangle |1\rangle &\mapsto |\tilde{\theta}\rangle \left(-\sin\tilde{\theta}\,|0\rangle + \cos\tilde{\theta}\,|1\rangle\right) \end{aligned} \tag{20}$$

Thus,

$$U_{\tilde{\theta}} = |\tilde{\theta}\rangle \langle\tilde{\theta}| \otimes exp(-i\tilde{\theta}\sigma_y) \tag{21}$$

This can be shown in a straightforward manner by the matrix notation of R_y in (19) and the vector notation of the basis $|0\rangle$, $|1\rangle$. The left part of (21) acts as the identity on $|\tilde{\theta}\rangle$, the right part rotates the input vector then around y by $\tilde{\theta}$. [Der+18]

Quantum circuits Let us here introduce shortly the notion of quantum circuits. They are graphical representations of a quantum algorithm, read from the left to the right. On the very left, the individual qubits (or, if one wants to summarize a collection of qubits, such as when approximating a real number by a bit string), the quantum registers can be found. From there on to the right, there is a line, called a wire, that describes the evolution of the register during computation. On this wire, one can apply unitary operations at a certain point, by means of quantum gates. The gates from the equations 12 and 14 depicted in a quantum circuit can be found in figure 2.

[5] i.e. using d qubits

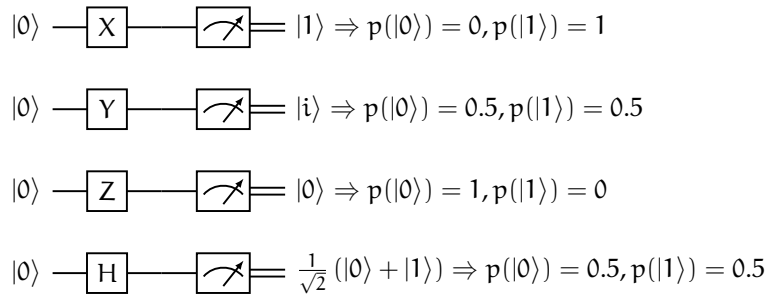
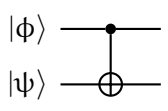


Figure 2: The introduced single qubit gates in a quantum circuit with measurement in the end; probabilites assuming the states are coherent and not perturbed (e.g. through measurement errors)

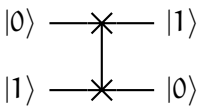
The quantum circuit notation thus combines the demands we put on a quantum computers: Quantum bits or registers are written under each other, and represent the state. The direction to the right along the wire, unitary transformations take care of the time evolution of the states. Finally, the state is measured (and thereby transferred to some classical information, which is illustrated by the double-line wire). A crossed wire represents a wire bundle, which means that the associated state consists of multiple qubits.

The C_X-gate discussed in the previous section is shown in figure 3a. Another useful gate is the SWAP-gate, which is able to exchange the states of two *single* qubits. Its graphical representation is depicted in figure 3b. Its matrix notation in the usual computational basis is represented by

$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{22}$$



(a) Depiction of a CNOT-gate, corresponding to equation (18)



(b) Depiction of a SWAP-gate, corresponding to equation (22)

Figure 3: Selected multiple-qubit gates

[Der+18; EHI00; Kib18]

2.3 Some examples of physical realizations

In this section, a few ideas on how quantum computers can be realized in practice are mentioned very briefly. The goal is just to give the reader a conceptual idea. Detailed explanations, including advantages and disadvantages, can be found e.g. in [NC10].

- Harmonic oscillator quantum computer
  - n qubits via energy levels $|0\rangle, |1\rangle, |2^n\rangle$ of a *single* quantum oscillator
  - Unitary evolution via matching the eigenvalue spectrum of the unitary operators with those given by the system's Hamiltonian
  - Just conceptual, so no initial state preparation or readout considered
- Optical photon quantum computer
  - Location of a single photon between two modes $|01\rangle, |10\rangle$ or polarization
  - Unitary evolution via so called phase shifters (R_z rotations) and beam splitters (R_y rotations)
  - Initial states via creation of single photon states (e.g. lasers)
  - Readout via detection of single photons
- Trapped ion quantum computers
  - Qubits represented by nuclear spin (so called hyperfine states) of an atom or lowest level vibrational modes of trapped atoms
  - Unitary evolution can be realized by application of laser pulses
  - Initial state can be created by cooling down atoms to motional ground state
  - Readout by measurement of the population of the hyperfine states
- ...and more

The optical quantum computer has the advantage, that it does not need cooling to temperatures close to absolute zero as for instance trapped ion quantum computers need. On the other hand, they are much delicate when it comes to decoherence, and need to operate in far shorter time scales.

3 The HHL algorithm

We already know, that everything, a classical computer is capable of, also the quantum machine can do. And further, arithmetic manipulations can be implemented efficiently. Thus, one might think of using fast classical solution schemes, e.g. Krylov methods such as GMRES or Conjugate Gradient, to solve this problem on a quantum computer. And possibly, there might be some speed-up. But this is not the point of using a quantum computer to approach the problem. As we know by now, the quantum nature of these machines (including concepts such as the superposition of states or entanglement) enables the user to approach problems in ways, that are not possible on classical computers. Thus, we will look into an algorithm that exploits the quantum nature to get significant (as it turns out, exponential) speedup over the best classical alternative.

In this section, the general problem statement is introduced, which will be followed by a few elementary building blocks we shall later need to assemble the HHL algorithm. Then, being familiar with the basic idea, we will look into how one could apply this algorithm onto solving a (very) specific, small example, following [Cao+12]. Finally, an overview will be given over the results of error and complexity analysis of the algorithm aswell extensions of it.

3.1 The problem statement

The problem statement of our linear system can be written as

$$Ax = b, \tag{23}$$

where $A \in \mathbb{R}^{N\times N}$, $x, b \in \mathbb{R}^N$, and A hermitian, i.e. $A^\dagger = A$. To solve (23) on a quantum machine, it has to be written in terms of a quantum state:

$$A\left|x\right\rangle = \left|b\right\rangle, \tag{24}$$

where $\left|x\right\rangle = \frac{x}{\|x\|}, \left|b\right\rangle = \frac{b}{\|b\|}$. Additionally, A is required to be sparse. This is not necessary for the algorithm to give results, but it is required for the speedup with respect to the best classical alternative holds. This will be further discussed in section 4.
[Der+18; HHL09]

3.2 The big picture

Now take a look at equation (24) from a different perspective. For A hermitian, one can write $A = \sum_{j=1}^{N} \lambda_j |u_j\rangle \langle u_j|$, where $\lambda_j \in \mathbb{R}$ are eigenvalues, and $|u_j\rangle \in \mathbb{R}^N$ the respective eigenvectors. Alongside, the right-hand side can be expressed as $|b\rangle = \sum_{j=1}^{N} |u_j\rangle \langle u_j|b\rangle = \sum_{j=1}^{N} \beta_j |u_j\rangle$. Now, as the system matrix is avaible in diagonal form, the solution to (24) can be easily computed:

$$|x\rangle = A^{-1} |b\rangle = \sum_{j=1}^{N} \frac{\beta_j}{\lambda_j} |u_j\rangle . \tag{25}$$

Thus, if there were quantum routines, that allow to give the spectral decomposition of the system matrix and apply the inverted eigenvalues on the right-hand side, this would be a first idea for a linear systems algorithm. And indeed, Harrow, Hassadim and Lloyd proposed such an algorithm in [HHL09], and furthermore proved, that it is in some sense optimal and cannot be improved significantly.

3.3 Important subroutines

At this point, the key subroutines of the linear systems algorithms will be presented subsequently, without any relation to each other. This has the aim to give the reader a understanding on how these algorithms work, such that the actual algorithm later can be read somewhat fluently and details can be looked up here.

3.3.1 Hamiltonian Evolution

Hamiltonian evolution or simulation refers to the second postulate proposed in section 2.1. Recall (1)

$$i\partial_t |\psi\rangle = \check{H} |\psi\rangle , \tag{26}$$

then the time evolution of the state $|\psi\rangle$ can be expressed as

$$|\psi(t)\rangle = e^{-i\check{H}t} |\psi\rangle . \tag{27}$$

On one hand, observe, that $e^{-i\check{H}t}$ is unitary[6]. Hence, every unitary operatur U can be written in terms of $e^{i\check{H}}$ for some hermitian $\check{H}$. This fact will be interesting when introducing the phase estimation procedure.

[6]This can easily be showed by the fact, that $\check{H}$ is hermitian and the definition of the matrix exponential.

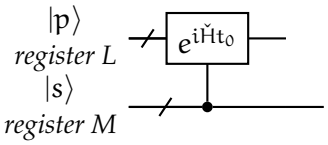


Figure 4: Conditional Hamiltonian simulation

For the purposes of this report, we shall not further worry about what $e^{-i\check{H}t}$ looks like when implemented on a quantum computer. Interested readers are referred to the literature (either a textbook such as [NC10], and also [Der+18] gives a good introduction to this topic). Remember, that by means of a universal family of quantum gates, it is possible, to write an arbitrary unitary operation in terms of a sequence of these fundamental gates. What usually is done now, is to find such an approximation for a unitary operation by a sequence of gates via an optimization algorithm, such as the Group Leader algorithm [DK11] – which in the following will be assumed to be done, whenever a non-elementary transformation such as Hamiltonian evolution is encountered.

Hamiltonian simulation can be implemented efficiently for large N, if the system matrix (here: $\check{H}$) is sparse [Der+18]. This is the only subroutine, where sparsity on the system matrix actually is required in order to get the full speedup of the HHL algorithm.

Before moving on to the next routine, *conditional* Hamiltonian simulation will be described. Consider a quantum circuit as in figure 4. There are two registers L and M with l, m qubits respectively. Starting with a state of $\sum_{\lambda=1}^{2^l-1}\sum_{\mu=1}^{2^m-1}|p_\lambda\rangle|s_\mu\rangle$ ($p_\lambda, s_\mu \in \{0,1\}$), Hamiltonian simulation via $\exp(i\check{H}t_0)$ conditioned on register M yields

$$\sum_{\lambda=1}^{2^l-1}\sum_{\mu=1}^{2^m-1}|p_\lambda\rangle|s_\mu\rangle \xrightarrow{\exp(i\check{H}t_0)} \sum_{\lambda=1}^{2^l-1}\sum_{\mu=1}^{2^m-1} e^{i\check{H}t_0 s_\mu}|p_\lambda\rangle|s_\mu\rangle\,. \tag{28}$$

Observe, that the Hamiltonian simulation is applied if $|s_\mu\rangle = |1\rangle$, if not the exponential is 1 and for any $|s\mu\rangle = |0\rangle$, the state is left as is. As this notation is rather cumbersome, we will omit the indices λ, μ in most cases from now on, and write the sum over s, p. Although this is abuse of notation, we will keep in mind, that in a $\sum_i |i\rangle$, in general, the state $|i\rangle$ for ith summand corresponds to the bitstring in binary notation up to the ith entry. For instance, in equation (33), this relation is visible between the third and fourth expression.

3.3.2 The Quantum Fourier Transform and the phase estimation procedure

QFT The discrete Fourier transfrom writes as

$$\mathrm{DFT} : y_k \to \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} e^{\frac{2\pi i}{N} kn} x_n \tag{29}$$

$$\mathrm{DFT}^\dagger : x_n \to \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} e^{\frac{-2\pi i}{N} kn} y_k. \tag{30}$$

Remark that the Fourier basis states are orthogonal, which means that the Fourier transformation itself is a unitary transformation. One can see this, as the adjoint of the DFT operator is indeed the inverse. This means, that the quantum analogue, the QFT, can be implemented by a sequence of elementary quantum gates.

The Quantum Fourier transformations can be analogeously written as

$$\mathrm{QFT} : |j\rangle \to \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i}{N} jk} |k\rangle \tag{31}$$

$$\mathrm{QFT}^\dagger : |k\rangle \to \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{\frac{-2\pi i}{N} jk} |j\rangle . \tag{32}$$

Now, we take $N = 2^n$ and look at some integer $k = \sum_{l=1}^{2^n} 2^{n-l} k_l$, while we denote this as $k = k_1 k_2 \dots k_n$, fractions of this are written as $\frac{k}{2^m} = 0.k_{n-m+1} \dots k_n$. Further, assume, that $|0\rangle \dots |2^{n-1}\rangle$ is a basis on our quantum computer. Performing the Fourier transform, a straightforward calculation or a look in [NC10] reveals

$$|j\rangle = |j_1 \dots j_n\rangle \xrightarrow{\mathrm{QFT}} \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi i}{N} jk} |k\rangle = \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{\frac{2\pi i}{N} j(\sum_{l=1}^{n} k_l 2^{-l})} |k_1 \dots k_n\rangle \tag{33}$$

$$\begin{aligned} &= \dots \\ &= \frac{1}{2^{n/2}} \left(|0\rangle + e^{2\pi i\, 0.j_n} |1\rangle\right) \left(|0\rangle + e^{2\pi i\, 0.j_{n-1} j_n} |1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i\, 0.j_1 j_2 \dots j_n} |1\rangle\right) \end{aligned}$$

QPE An important application of the QFT is the phase estimation procedure (QPE). The name phase estimation comes from the fact, that the eigenvalues of a unitary operator U can be written as $U|u_j\rangle = \lambda_j |u_j\rangle = e^{2\pi i \varphi_j} |u_j\rangle$. The phase estimation therefore is able to determine $\varphi_j \in [0, 1]$, or more precisely, to get an n-bit approximation $\varphi_j \approx \tilde{\varphi}_j = 0.\varphi_{j,1} \varphi_{j,2} \dots \varphi_{j,n}$.

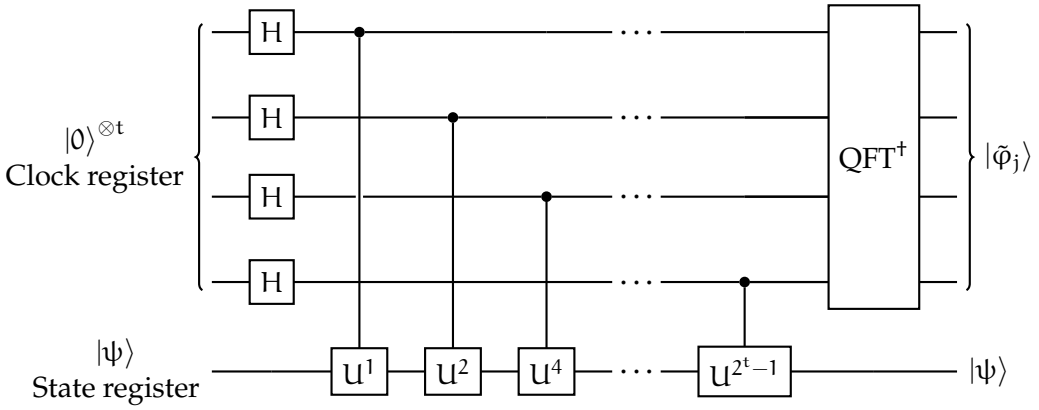


Figure 5: Quantum phase estimation circuit

Two registers, one of size t prepared in the $|0\rangle^{\otimes t}$ state, and one with m qubits, $|\psi\rangle \in \mathbb{C}^{2^m}$, of which we assume for now, that it is an eigenvector of U. The notation $|0\rangle^{\otimes t}$ means, that t qubits of the same state are combined together with the tensor product, $|0\rangle \otimes \cdots \otimes |0\rangle$. The choice of t depends on how accurate one wants to represent the eigenvalues and on the desired probability of success of the procedure. The next step is to apply a Hadamard-transform on the t qubits to create a superposition of states. Formally,

$$|0\rangle^{\otimes t} |\psi\rangle \xrightarrow{H^{\otimes n} \otimes \mathbb{I}} \frac{1}{2^{t/2}} (|0\rangle + |1\rangle)^{\otimes t} |\psi\rangle . \tag{34}$$

Now, the controlled U^{2^k}-operation is applied. Meaning, U^{2^0} conditioned on the first qubit in the first register operates on the second register, U^{2^1} conditioned on the second qubit on the first register, and so on until $U^{2^{t-1}}$.
Remembering that $U|\psi\rangle = e^{2\pi i \varphi} |\psi\rangle$, we end up with the following expression for the first register. This can be identified with the QFT, when comparing with (33).

$$\begin{aligned} &\frac{1}{2^{t/2}} \left(|0\rangle + e^{2\pi i\, 2^{t-1} \varphi} |1\rangle\right) \left(|0\rangle + e^{2\pi i\, 2^{t-2} \varphi} |1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i\, 2^{0} \varphi} |1\rangle\right) \\ =&\frac{1}{2^{t/2}} \left(|0\rangle + e^{2\pi i\, 0.\varphi_t} |1\rangle\right) \left(|0\rangle + e^{2\pi i\, 0.\varphi_{t-1} \varphi_t} |1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i\, 0.\varphi_1 \varphi_2 \ldots \varphi_t} |1\rangle\right) \\ =&\frac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2\pi i \varphi k} |k\rangle \end{aligned} \tag{35}$$

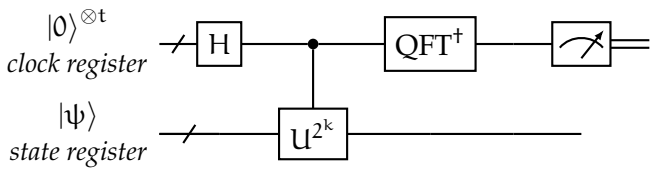


Figure 6: Phase estimation procedure

Applying the inverse QFT, one obtains for the full state:

$$\frac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2\pi i \varphi k} |k\rangle |\psi\rangle \xrightarrow{\text{QFT}^\dagger} \frac{1}{2^t} \sum_{x=0}^{2^t-1} \sum_{k=0}^{2^t-1} e^{-\frac{2\pi i k}{2^t}(x-2^t\varphi)} |x\rangle |\psi\rangle . \tag{36}$$

This is expression peaks near $x = \lfloor 2^t\varphi \rfloor$. The reason therefore is, that sums like $\sum_{k=0}^{M-1} e^{2\pi i k x/M}$ are zeros except x is integer-divisible by M. If x is an integer, the $x = 2^t\varphi$, and $|2^t\varphi\rangle |\psi\rangle$ can be obtained by measurement with high probability. If x is not an integer, this measurement has to be repeated, for which an upper bound of repetitions until success can be derived [NC10]. Recall, that we represent φ only by an t-bit approximation $\varphi \approx \tilde{\varphi} = \sum_{m=1}^{t} \varphi_m 2^{-m}$. We will keep the tilde-notation for binary approximations from now on, and only express the sum explicitly, if the number of qubits used is not clear from the context.

At this point, recall the fact, that the Hamiltonian evolution operator $e^{-i\check{H}t}$ is unitary. Furthermore, $e^{-i\check{H}t} |u_j\rangle = e^{-i\lambda_j t} |u_j\rangle$. Thus, substituting U in the QPE by the evolution operator, gives an estimation of the Hamiltonian $\check{H}$, or in general, any hermitian matrix. The first register from before is in this case usually called the *clock* register, as it is responsible for the time evolution.

A physical intuition can be given via the von Neumann scheme of measurement. [Q] Here, consider the first register as a "pointer" register (basically the clock above), which we continue to call clock register for reasons, that will get clearer in a moment. Further, denote, that an application of the momentum operator $\check{p}$ "pushes" the state forward in time. The idea now is, to "push" the pointer register, such that it contains the eigenvalues. Pointer and system have the initial state $|0\rangle |u_j\rangle$, and undergo the unitary transformation $e^{-it\check{p}\otimes\check{H}}$, which gives the final state $|\lambda_j t\rangle |u_j\rangle$. It turns out, that translating $e^{-it\check{p}\otimes\check{H}} |0\rangle |u_j\rangle$ to a quantum computer gives exactly the QPE-scheme [Q][7].

[7] The IBM Q-Experience website has been rebuilt during the work on this report. Under the supplied link, you still find the derivation via the von Neumann measurement, the new site provides the same

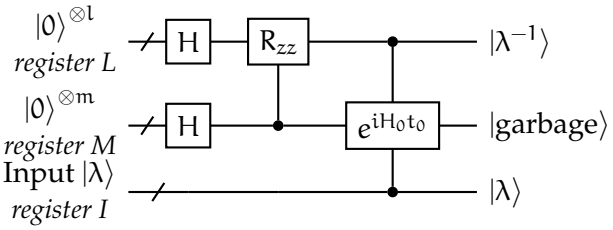


Figure 7: Quantum circuit to invert a number

An interesting fact for the linear systems algorithm is, that one does not need to have access to an eigenvector $|u_j\rangle$. Any quantum state can be decomposed into an arbitrary, orthogonal basis, which here will be chosen to be the eigenbasis of the unitary operator U:

$$|\psi\rangle = \mathbb{I}\,|\psi\rangle = \sum_j |u_j\rangle\,\langle u_j|\psi\rangle = \sum_j \beta_j\,|u_j\rangle\,. \tag{37}$$

To wrap up, the QPE-procedure applied to an arbitrary state $|\psi\rangle$ yields

$$|0\rangle\,|\psi\rangle = \sum_j \beta_j\,|0\rangle\,|u_j\rangle \xrightarrow{\text{QPE}} \sum_{j,k} \alpha_{j,k}\beta_j\,|\tilde{\varphi}_k\rangle\,|u_j\rangle\,, \tag{38}$$

where $\alpha_{j,k}$ is a preconstant resulting from the QPE-procedure. If the QPE is perfect, $\alpha_{j,k} = 1$ and $|\tilde{\varphi}_k\rangle = |\lambda_j\rangle$, which we assume in this report.
[Der+18; NC10]

3.3.3 How to invert a number on a quantum-computer

The QPE gives the eigenvalues of the system matrix, but as we shall see, it is required, that the inverse eigenvalues λ_j^{-1} are available. Usually, in papers that do not deal with the implementation but only with the algorithm, this is omitted, as one can show, that arithmetic manipulations on a quantum machine can be achieved easily [Bac06].

To give an idea, the way to invert the eigenvalues proposed in [Cao+12] will be explained briefly.

Figure 7 shows the quantum circuit of the algorithm. Input is a state $|\lambda\rangle$ with T qubits, containing one or also possibly multiple numbers one wants to invert. Apart from that, register L contains l and register M contains m qubits.

derivation as [NC10]

Here we assume, that $|\lambda\rangle$ is available from a previous computation. The outcome of register M is not further needed and therefore a so called $|\text{garbage}\rangle$-state, that was produced along the way when aiming for some other outcome.

The algorithm starts with a Hadamard-transfrom of the with $|0\rangle$ initialized registers L and M. Then, a conditional phase shift R_{zz} on register L is applied, which results in the state

$$\sum_{s=0}^{2^l-1}\sum_{p=0}^{2^m-1} e^{i\frac{p}{2^m}t_0}\,|s\rangle^L\,|p\rangle^M\,|\lambda\rangle^I\,. \tag{39}$$

Superscripts over the quantum states denote to which register they relate to. The next step is the Hamiltonian evolution $e^{iH_0t_0}$ of M conditioned on L and I. Here, $H_0 = \text{diag}(1,2,\dots 2^{m-1})$, which is clearly hermitian. The double conditioning can be interpreted as follows: The k_lth qubit of L aswell as the jth eigenvalue of I[8] controls a $e^{-ip(\frac{\lambda_j}{2^m}\frac{1}{2^{l-k_l}}t_0)}$ simulation acting on M. The final state thus reads

$$\sum_{j=1}^{n}\sum_{s=0}^{2^l-1}\sum_{p=0}^{2^m-1} e^{i\frac{p}{2^{m+l}}t_0(2^l-\lambda_j s)}\,|s\rangle^L\,|p\rangle^M\,|\lambda_j\rangle^I\,. \tag{40}$$

Note the color coding to recognize the conditioning relations better.

Using $t_0 = 2\pi$, the final state is sharply peaked for states, when $s = \frac{2^l}{\lambda_j}$.

Writing the register L as $|\frac{2^l}{\lambda_j}\rangle$, one can observe, that the algorithm yielded the desired inverse up to a known constant.

[Cao+12]

[8] This again is in some sense abuse of notation, but makes the relations quite clear. If one wants to be more precise, the eigenvalues can be decomposed in binary form, which then also alters the upper limit for the sum over j. But if the eigenvalues were as in3.5, the given expression would be true, and the control would actually act on the jth qubit.

3.4 Putting everything together - the HHL algorithm

Now, it is possible to assemble the steps to the linear systems algorithm, which is depicted in figure 8. One starts off with some initial state $|\text{initial}\rangle$, and assuming that there exists a unitary operation B, that can prepare the input $|b\rangle^I$ efficiently, one obtains the latter state. Efficiently means here, such that it does not harm the complexity of $\mathcal{O}(\log N)$ in the end. This can be for instance achieved by the qRAM suggested in [Der+18]. Any errors in creation of the input state are neglected. The algorithm starts with the phase estimation procedure, for which the clock register should be prepared in the state

$$|\Psi_0\rangle^C = \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} \sin\left(\frac{\pi(\tau+\frac{1}{2})}{T}\right) |\tau\rangle^C , \tag{41}$$

which minimizes a certain cost function [BDM99]. The reason for preparing the clock in a manner that is optimal in some sense, is that the pre-constant resulting of phase estimation as in (38) can be bound from above, which turns out to be very useful in the error analysis of the algorithm. This can be done in polylogarithmic time [GR02]. T is the number of computational steps, that is needed, to compute the Hamiltonian evolution e^{iAt} for $0 \leq t \leq t_0$, when A is sparse. Thus, $\frac{t_0}{T}$ corresponds to the step size of the Hamiltonian simulation.

The next step is the phase estimation procedure, while here, the eigenvalues of the unitary e^{iAt} are the subject of desire. As A is hermitian, e^{iAt} is unitary with $e^{iAt}|u_j\rangle = e^{i\lambda_j t}|u_j\rangle$, while λ_j are also eigenvalues of A. Applying $\sum_{\tau=0}^{T-1} |\tau\rangle\langle\tau|^C \otimes e^{iA\tau t_0/T}$ conditional on the clock register, the registers C and I read

$$|\Psi_0\rangle^C |b\rangle^I \xrightarrow{\sum_{\tau=0}^{T-1}|\tau\rangle\langle\tau|^C\otimes e^{iA\tau t_0/T}} \sqrt{\frac{2}{T}} \sum_{j=1}^{N} \beta_j \left(\sum_{\tau=0}^{T-1} e^{\frac{i\lambda_j \tau t_0}{T}} \sin\left(\frac{\pi(\tau+\frac{1}{2})}{T}\right) \right) |u_j\rangle^I . \tag{42}$$

The inverse QFT yields

$$(42) \xrightarrow{QFT^\dagger} \sum_{j=1}^{N} \beta_j \sum_{k=0}^{T-1} \underbrace{\left(\sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} e^{\frac{i\tau}{T}(\lambda_j t_0 - 2\pi k)} \sin\left(\frac{\pi(\tau+\frac{1}{2})}{T}\right) \right)}_{:=\alpha_{k,j}} |k\rangle^C |u_j\rangle^I \tag{43}$$

For the following, we will abbreviate the bracket expression in (43) with $\alpha_{k,j} \in \mathbb{R}$. One can show, that $\alpha_{k,j} \leq \frac{8}{\pi^2}$, which is important for the error analysis of the algorithm. Further, the upper bound holds whenever $|k - \lambda_j \frac{t_0}{2\pi}| \geq 1$. $\alpha_{k,j}$ is large only if $\lambda_j \approx \frac{2\pi k}{t_0}$. Now, we rename the states in the clock register to this expression, namely $\lambda_j \approx \frac{2\pi k}{t_0}$.

The current state in numerical approximation (tilde-notation) can thus be written as

$$\sum_{j=1}^{N} \beta_j \sum_{k=0}^{T-1} \alpha_{k,j} \left|\tilde{\lambda}_k\right\rangle^C \left|u_j\right\rangle^I, \tag{44}$$

as $\tilde{\lambda}$ depends also on k. At this point, an arbitrary algorithm to invert for the eigenvalues can be applied, such as the one proposed in [Cao+12]. To keep things simpler here, assume, register C now contains the inverse eigenvalues (or, to be more exactly, a state proportional to the inverse eigenvalues). As already mentioned, this is the usual procedure in the literature, as arithmetic on quantum states can be accomplished easily.

Now, an ancilla register is adjoined, which is initialized in the state $|1\rangle^S$. A $R_y(\tilde{\theta})$ rotation conditioned on register C is applied, again assuming that C now contains the inverse eigenvalues. Now conditional y-rotation around $\tilde{\theta} = -\gamma\tilde{\lambda}_k^{-1}, \gamma \in \mathbb{R}$ as in 21 will be applied (recalling the results of section 3.3.3 $\tilde{\theta} = 2^1\tilde{\lambda}_k^{-1}$). This results in

$$\begin{aligned} &\sum_{j=1}^{N} \beta_j \sum_{k=0}^{T-1} \alpha_{k,j} \left|1\right\rangle^S \left|\tilde{\lambda}_k^{-1}\right\rangle^C \left|u_j\right\rangle^I \\ \xrightarrow{R_y(\tilde{\lambda}_k^{-1})} &\sum_{j=1}^{N} \beta_j \sum_{k=0}^{T-1} \alpha_{k,j} \left(\sqrt{1 - \frac{\gamma^2}{\tilde{\lambda}_k^2}} \left|0\right\rangle^S + \frac{\gamma}{\tilde{\lambda}_k} \left|1\right\rangle^S\right) \left|\tilde{\lambda}_k^{-1}\right\rangle^C \left|u_j\right\rangle^I. \end{aligned} \tag{45}$$

If the phase estimation was perfect, $\alpha_{k,j} = 1$ for $\tilde{\lambda}_k = \lambda_j$, which we assume in this report. The state can thus be written as

$$\sum_{j=1}^{N} \beta_j \left(\sqrt{1 - \frac{\gamma^2}{\lambda_j^2}} \left|0\right\rangle^S + \frac{\gamma}{\lambda_j} \left|1\right\rangle^S\right) \left|\lambda_j^{-1}\right\rangle^C \left|u_j\right\rangle^I. \tag{46}$$

Now, the end is almost in sight, although we are not yet finished.

As the current state between the individual qubits and registers is entangled, a measurement in one register is highly compromitted by the others, if their content is not needed anymore[9]. This is the case for the clock register containing the (inverse) eigenvalues at this point. As there is no possibility in quantum computing, to simply delete the information, a procedure called *uncomputation* is used.

[9] When measuring, one has to capture the whole state of all registers combined (via the tensor product). This means, that each state, that is in a superposition, contributes to the system's global probability distribution. Thus, it is highly advantageous to set the states one is not interested in to $|0\rangle$. $|1\rangle$ would also be possible, but the former is practically easier to achieve.

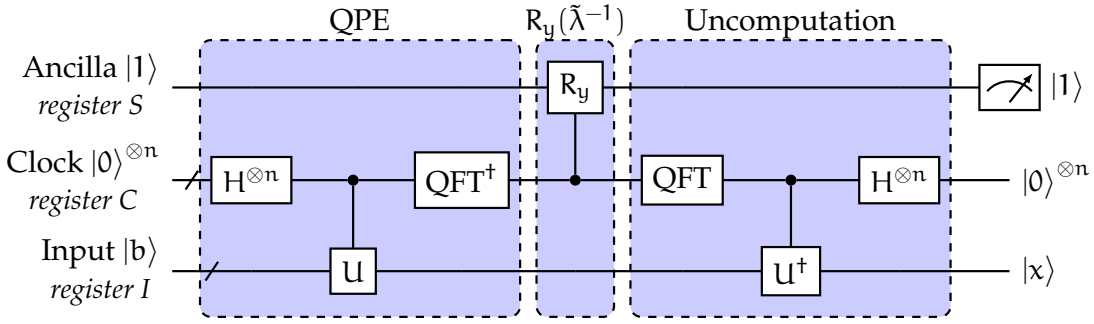


Figure 8: HHL algorithm, subdivided in QPE (quantum phase estimation) to determine the eigenvalues, controlled rotation R_y to "extract" the eigenvalue information and uncomputation to revert the phase estimation

This can be achieved by an application the adjoint of the unitary transformation so far , which corresponds to undoing all computations. Thus, the clock register can be set back to $|0\rangle^C$ this way. Recall the note on entanglement in section 2.2.1, in particular the examples for factored and entangled states. One can see, that after uncomputation, a factored state can be achieved.

The final step is a measurement on the Ancilla register, while postselecting on $|1\rangle$. This can be expressed by a projection onto the $|1\rangle$-subspace of the register S, thus discarding the $|0\rangle$-part:

$$(46) \xrightarrow{|1\rangle\langle 1|^S} \sum_{j=1}^{N} \beta_j \left(\sqrt{1 - \frac{\gamma^2}{\lambda_j^2}} \langle 1|0\rangle + \frac{\gamma}{\lambda_j} \langle 1|1\rangle \right) |1\rangle^S |0\rangle^C |u_j\rangle^I \tag{47}$$

Finally, the solution to $A|x\rangle = |b\rangle$ up to a (known, global) constant factor γ is obtained in re-normalized form in the input register:

$$|x\rangle = \frac{1}{\sqrt{\sum_{j=1}^{N} \gamma^2 \beta_j^2 / \lambda_j^2}} \sum_{j=1}^{N} \gamma \frac{\beta_j}{\lambda_j} |u_j\rangle^I . \tag{48}$$

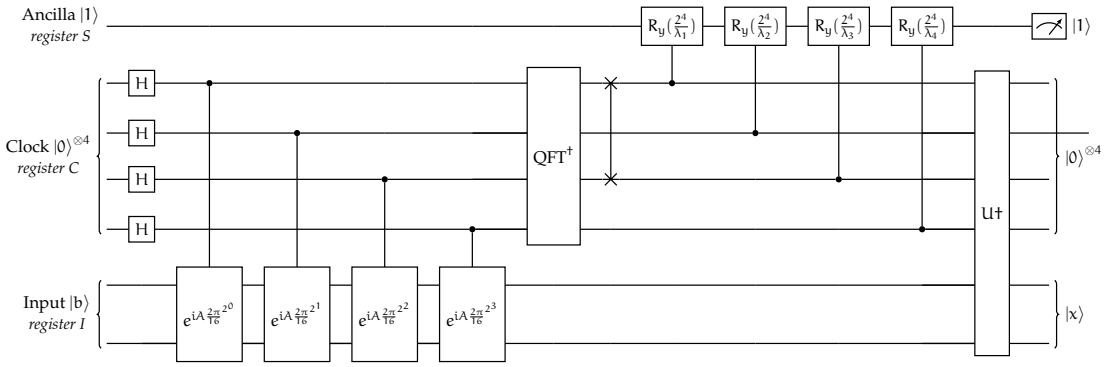


Figure 9: HHL algorithm for a specific 4-dimensional system; $U^\dagger$ includes uncomputation of the QPE aswell as of the swap-gate. Observe, that the conditioned rotation on 4 qubits needs to be decomposed according to the eigenvalues

3.5 Application on a low-dimensional example

The linear systems algorithm shall now be applied to a very specific 4×4-system after [Cao+12]. In the beginning, let us take a look at the quantum circuit.

The system that will be solved for has the form of equation (24), while

$$A = \frac{1}{4}\begin{pmatrix} 15 & 9 & 5 & -3 \\ 9 & 15 & 3 & -5 \\ 5 & 3 & 15 & -9 \\ -3 & -5 & -9 & 15 \end{pmatrix} \tag{49}$$

and the right-hand side

$$|b\rangle = \frac{1}{2}(1,1,1,1)^T. \tag{50}$$

First, one can see, that A is real and symmetric, thus also hermitian. Secondly, it just happens to be, that the eigenvalues of A are $\lambda_j = 2^{j-1}, j = 1,\dots,4$ with the corresponding eigenvectors $|u_j\rangle = \frac{1}{2}\sum_{i=1}^{4}(-1)^{\delta_{ij}}|i\rangle^C$, i.e. $|u_1\rangle = \frac{1}{2}(-|0001\rangle + |0010\rangle + |0100\rangle + |1000\rangle)$. Now, observe that $|b\rangle$ can be rewritten as $|b\rangle = \sum_{j=1}^{4}\beta_j|u_j\rangle$, while $\beta_j = \frac{1}{2}$ for all $j = 1,\dots,4$. This means, the input register can be prepared by starting with $|0\rangle^{\otimes 2}$, and applying a Hadamard-gate at each qubit, which yields $|b\rangle$ as in (50) in the computational basis.

The corresponding quantum circuit that will be used can be found in figure 9. The first step, having prepared $|b\rangle$, is as usual the QPE. The unitary operations U^{2^k} in this case are $e^{iAt_0/16}$, $e^{iAt_0/8}$, $e^{iAt_0/4}$ and $e^{iAt_0/2}$, with $k = 0,1,2,3$ with $t_0 = 2\pi$.

Furthermore, the simulation time T according to the number of qubits is $T = 2^4 = 16$. This gives the state

$$\sum_{j=1}^{4} \underbrace{\frac{1}{2}}_{\beta_j} |\lambda_j\rangle^C |u_j\rangle^I \tag{51}$$

As the eigenvalues are integers, the QPE can be assumed to be exact.

At this point, the inversion procedure from section 3.3.3 could be applied. But considering the specific properties of the system, we can take a shortcut. The goal is to have states of the form $|\frac{2^4}{\lambda_j}\rangle$ (compare to section 3.3.3, last paragraph). Now, write the eigenvalues in binary form as $\lambda_1 = 1 = |0001\rangle, \lambda_2 = 2 = |0010\rangle, \lambda_3 = 4 = |0100\rangle, \lambda_4 = 8 = |1000\rangle$.

Observe, that

$$\begin{aligned}
\lambda_4^{-1} = |1000\rangle^{-1} &\rightarrow \frac{1}{8} \cdot 2^4 = |0010\rangle = |\frac{2^4}{\lambda_4}\rangle \\
\lambda_3^{-1} = |0100\rangle^{-1} &\rightarrow \frac{1}{4} \cdot 2^4 = |0100\rangle = |\frac{2^4}{\lambda_3}\rangle \\
\lambda_2^{-1} = |0010\rangle^{-1} &\rightarrow \frac{1}{2} \cdot 2^4 = |1000\rangle = |\frac{2^4}{\lambda_2}\rangle \\
\lambda_1^{-1} = |0001\rangle^{-1} &\rightarrow \frac{1}{1} \cdot 2^4 = 2^4 |0001\rangle = |\frac{2^4}{\lambda_1}\rangle .
\end{aligned} \tag{52}$$

This means, that to get the inverse eigenvalue in desired form, it is only necessary to swap the first and third qubit of the clock register here. Thus, it is now possible to adjoin the ancilla register S and to rotate conditioned on the inverse eigenvalues. Before the projective measurement of the ancilla, uncomputation is performed in the usual manner (this also includes undoing the swap of the eigenvalues).

Finally, we end up in a state $\frac{1}{\sqrt{85}}(8|u_1\rangle + 4|u_2\rangle + 2|u_3\rangle + |u_4\rangle)^I \otimes |0000\rangle^C + |1\rangle^S$. This can be rewritten in the basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, giving

$$|x\rangle = \frac{1}{\sqrt{340}} (-|00\rangle + 7|01\rangle + 11|10\rangle + 13|11\rangle) \propto \frac{1}{32} (-1, 7, 11, 13)^T = x. \tag{53}$$

Thus, we have the solution of the linear system up to a constant. Note, that only two qubits are necessary here to encode a $\mathbb{R}^4$-vector. This also means, that the solution still is encoded in the quantum state. To read it out, one would have to perform (multiple!) measurements on the input, to end up with a probability distribution corresponding to the final state.

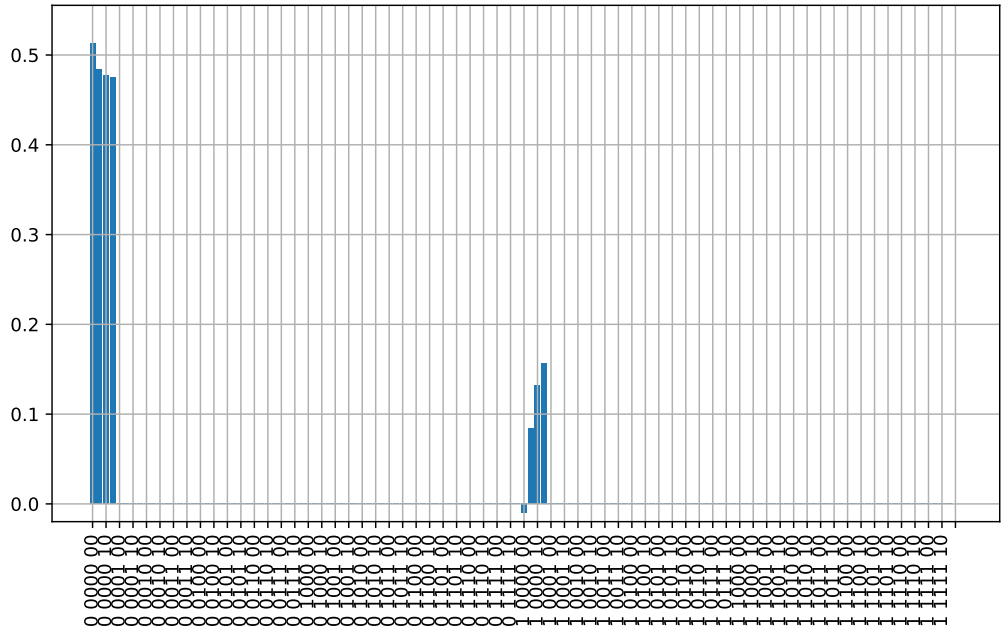


Figure 10: Result of the HHL algorithm applied on the 4×4-system using the tool of [Adr18]; The y-axis shows the amplitude, while the x-Axis represents the individual qubits. On the very left, one can identify the vector $|b\rangle$, and in the middle the solution of the system. Observe, that this result matches with the diagram in [Cao+12]

There can be found several tools online to play around with this implementation. On [10], you can find an implementation on Quirk, which is a tool to simulate quantum circuits that can be built with a graphical interface. Another very nice implementation is given by [Adr18], available on GitHub under [11]. This tool gives up to numerical precision the same result as 53, and a plot for the final qubit state, which is depicted in figure 10. It makes especially sense to try it out, if one is interested in the entire experimental procedure, as it also provides a heuristic optimization algorithm that approximates the unitary transformations.

[10] https://goo.gl/v4JbRw or https://quantumcomputing.stackexchange.com/questions/3743/error-simulation-of-quantum-algorithm-for-linear-systems-of-equations-for-4

[11] https://github.com/nelimee/quantum-hhl-4x4

4 Additional remarks on the HHL algorithm

Dealing with ill-conditioned systems The algorithm relies on an eigendecomposition of A and inverting for its eigenvalues. Now, suppose some of these eigenvalues are close to zero. As they are slightly perturbed by numerical precision, this has the consequence, that the system is ill-conditioned in the subspace spanned by the eigenvectors corresponding to these eigenvalues. To overcome this, [HHL09] proposed filter functions, that allow, to either only invert on the well-conditioned subspace or estimate the ill-conditioning of the system. The idea is to have functions $f(\lambda), g(\lambda)$, that have to fulfil certain continuity properties, and add an extra three-qubit register

$$|h(\lambda)\rangle = \sqrt{1 - f(\lambda)^2 - g(\lambda^2)}\,|\text{nothing}\rangle + f(\lambda)\,|\text{well}\rangle + g(\lambda)\,|\text{ill}\rangle\,, \tag{54}$$

where nothing means no inversion took place, well and ill express well- and inconditionedness. Now, the technique of amplitude amplification is applied, which for instance is also used in Grover's algorithm. This can be understood such that the well-part is amplified and the ill-part is dampened by means of some special unitary operators. In the end, postselecting on $|\text{well}\rangle$ is applied.

This routine though not anymore unitary due to the filter functions. This introduces an error and requires some repetition, until the true result is obtained with certainty. [Der+18; HHL09]

Dealing with non-hermitian systems Suppose now A non-hermitian. Then define $C = \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}$. Solving $Cy = b^*$ with $y = (0, x)^T$ and $b^* = (b, 0)^T$ gives the desired solution, as C now is hermitian. This can be also extended to non-square cases, which are further discussed in [Der+18; HHL09].

Error analysis and complexity Harrow et. al assume, that the errors due to Hamiltonian simulation and preparing the initial state are neglegible, which holds, as they can be implemented efficienctly. The main influence of errors thus arises from the post-selecting when using the filter functions, as there is still some information left in the $|\text{ill}\rangle$-part, that is discarded. They show, that finally, the error with respect to the exact solution is of order $\mathcal{O}(\kappa/t_0)$, where κ represents the condition number and t_0 the simulation time chosen for the Hamiltonian simulation. Taking $t_0 = \mathcal{O}(\kappa/\epsilon)$, this gives an error of ϵ on the final state. With this information, it is now possible, to calculate the number of qubits needed to obtain a solution up to an error ϵ.

The overall complexity in the end from the initial state to the solution (both in form of a quantum state) turns out to be $\mathcal{O}(\kappa^2/\epsilon s^2 \log N)$, where s is the sparsity of the system. In comparison to that, solving a linear system by Gauss elimination takes $\mathcal{O}(N^3)$, the conjugate gradient algorithm for sparse, symmetric positive definite matrices needs $\mathcal{O}(Ns\kappa \log \frac{1}{\epsilon})$.

If interested, the detailed error analysis can be found in [HHL09], and in even more detail described in [Der+18].

5 Conclusions and outlook

After having introduced some basic concepts of quantum mechanics and how these can be used to store and process information (building a computer), the so called HHL-Algorithm aswell as some important subroutines were presented. This algorithm has the potential to be a powerful tool in the future, as soon as quantum computers are accessible to the public or at least a larger range of people. As mentioned before, the HHL algorithm gives the solution (up to a known constant) of a linear system encoded in a quantum state. Assuming efficient preparation of $|b\rangle$ and sparsity of A, this can be done in time $\mathcal{O}(\log N)$, which is exponentially faster than the best known classical algorithm for solving a linear system, the Conjugate Gradient method ($\mathcal{O}(N)$). But looking at it sharply, this speedup can be achieved only if one is not interested in reading out the entries of x explicitly. A possible outcome then might be some global expectation value $\langle x|M|x\rangle$ to some hermitian M. A strong field of application for this algorithm therefore is machine learning on quantum computers. In machine learning, one usually has a lot of data - the advantage of the algorithm here is, that one only requires $\log N$ qubits to encode N bits of data. Furthermore, one is usually not interested in each bit of data, but looks out for global properties. For example [Zha+18] uses the HHL algorithm applied on a deep learning problem. An implementation in python, that can be used to run the HHL algorithm on quantum computers by Rigetti, can be found under [12].

Another use might be to use the algorithm calculating functions of some hermitian operator A[13], as $f(A) = \sum_j f(\lambda_j)|u_j\rangle\langle u_j|$. Now, instead of calculating the inverse of the eigenvalue, one applies some function f on it. The rest of the algorithm can stay [HHL09].

Another point to adress is the (once understood) simplicity of the algorithm, to invert the matrix using the eigendecomposition. Here, the algorithm is not approximative, but would give arithmetically the exact solution considering having exact eigenvalues. This is the bottleneck though, as the phase-estimation procedure indeed is of iterative, approximating nature, and is the main point contributing to the error analysis in [HHL09]. Another peculiarity of the QPE is the Hamiltonian simulation, which has to run for a certain time. As current quantum computers are quite suscepible for noise from the surroundings, this comparably long time needed for the simulation can be a fatal due to quantum decoherence.

[12] https://gitlab.com/apozas/bayesian-dl-quantum/blob/master/HHL_Rigetti.py
[13] or the hermitian extension of an arbitrary operator

In other applications, when eigenvalues are needed, such as quantum chemistry, another approach is used: Instead of simulating the entire Hamiltonian, it is decomposed in small blocks using second quantization [Per+14]. Unfortunately, for the purposes of solving linear systems, this method is not applicable, as it returns the eigenvalues in terms of expectation values of the individual parts of the decomposed Hamiltonian.

The HHL algorithm has also already been applied on Finite-Elements [MP16], and research has been done on improving it by preconditioning [CJS13]. It turns out, that when using the preconditioner, a polynomial speedup can achieved by using the quantum algorithm. Thus, the algorithm is not only valuable for fields as machine learning, but also classical disciplines of scientific computing.

The HHL algorithm provides a universal tool to the field of quantum computing. At this point (2019), more than ten years after being published, the main problem that it is not yet in wider use, are availability and accesibility of quantum computers with a significant number of qubits. So we can curiously look forward to the future, where a new era of computing will await us.

References

[Adr18] CERFACS Adrien Suau. *Implementation the HHL for a 4x4 system*. 2018. URL: https://github.com/nelimee/quantum-hhl-4x4.

[Bac06] Dave Bacon. *Quantum Computing Reversible Classical Circuits and the Deutsch-Jozsa Algorithm*. Tech. rep. Department of Computer Science and Engineering, Universtiy of Washington, 2006. URL: https://courses.cs.washington.edu/courses/cse599d/06wi/lecturenotes6.pdf.

[BDM99] V. Bužek, R. Derka, and S. Massar. "Optimal quantum clocks". In: *Physical Review Letters* 82.10 (1999), pp. 2207–2210. arXiv: 9808042v2 [arXiv:quant-ph].

[Cao+12] Yudong Cao et al. "Quantum circuit design for solving linear systems of equations". In: *Molecular Physics* 110.15-16 (2012), pp. 1675–1680. arXiv: arXiv:1110.2232v2.

[CJS13] B. D. Clader, B. C. Jacobs, and C. R. Sprouse. "Preconditioned quantum linear system algorithm". In: *Physical Review Letters* 110.25 (2013), pp. 1–5. arXiv: arXiv:1301.2340v4.

[Der+18] Danial Dervovic et al. "Quantum linear systems algorithms: a primer". In: (2018). arXiv: 1802.08227. URL: http://arxiv.org/abs/1802.08227.

[DK11] Anmer Daskin and Sabre Kais. "Decomposition of unitary matrices for finding quantum circuits: Application to molecular Hamiltonians". In: *Journal of Chemical Physics* 134.14 (2011). arXiv: arXiv:1009.5625v3.

[EHI00] Artur Ekert, Patrick Hayden, and Hitoshi Inamori. "Basic concepts in quantum computation". In: 3 (2000), pp. 1–37. arXiv: 0011013 [quant-ph]. URL: http://arxiv.org/abs/quant-ph/0011013.

[GR02] Lov Grover and Terry Rudolph. "Creating superpositions that correspond to efficiently integrable probability distributions". In: (2002), pp. 1–2. arXiv: 0208112 [quant-ph]. URL: http://arxiv.org/abs/quant-ph/0208112.

[HHL09] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum algorithm for linear systems of equations". In: *Physical Review Letters* 103.15 (2009), pp. 1–15. arXiv: arXiv:0811.3171v3.

[Kib18] Maurice Robert Kibler. "Quantum information and quantum computing: an overview and some mathematical aspects". In: 1 (2018), pp. 1–40. arXiv: 1811.08499. URL: http://arxiv.org/abs/1811.08499.

[MP16] Ashley Montanaro and Sam Pallister. "Quantum algorithms and the finite element method". In: *Physical Review A* 93.3 (2016), pp. 1–16. arXiv: arXiv:1512.05903v2.

[NC10] M. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 10th. New York, NY, USA: Cambridge University Press, 2010, pp. 500–527, 582–607. arXiv: arXiv:1011.1669v3.

[Per+14] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor". In: *Nature Communications* 5.2 (2014), pp. 1–10. arXiv: arXiv:1304.3061v1.

[Q] IBM Q. *IBM Q Experience Full User Guide*. URL: https://quantumexperience.ng.bluemix.net/proxy/tutorial/full-user-guide/introduction.html (visited on 06/21/2019).

[Zha+18] Zhikuan Zhao et al. "Bayesian Deep Learning on a Quantum Computer". In: (2018), pp. 1–11. arXiv: 1806.11463. URL: http://arxiv.org/abs/1806.11463.