HHL Algorithm: Solving Linear Systems Goes Quantum

Can Erol
PHYS 345

May 2025

1 Introduction

Solving linear systems of equations is a fundamental computational problem that manifests itself in virtually every field of science and engineering [1]. More formally, given a $N \times N$ matrix A and a vector $\vec{b} \in \mathbb{C}^N$, we often find ourselves seeking a solution $\vec{x} \in \mathbb{C}^N$ to the system:

$$A\vec{x} = \vec{b}.$$

Hence, this problem, which we colloquially call $A\vec{x} = \vec{b}$, is central for applications ranging from numerical simulation and optimization to machine learning and data analysis. Yet, this task of solving $A\vec{x} = \vec{b}$ becomes computationally demanding as the dimension of N grows. Generally, using Gaussian elimination or other methods require a runtime on the order of $O(N^3)$ for an $N \times N$ system, which, intuitively, becomes infeasible for very large N [2]. Thus, even if A is sparse[1] and specialized algorithms are utilized, classical solvers still scale at least linearly in N [3]. In short, all known classical approaches require polynomial time in N, directly implying that solving extremely large systems remain an incredibly challenging task on classical computers.

This is where the search for fundamentally new approaches starts. The reader is perhaps familiar with the fact that quantum computing offers the intriguing possibility of an *exponential speed-up* for certain problems by leveraging the power of quantum states. Thus, in 2009, Harrow, Hassidim, and Lloyd introduced a quantum algorithm for solving linear systems, with the promise of a dramatic improvement in runtime scaling [1]. Now commonly known as the HHL algorithm, this quantum routine specifically targets the case where A is Hermitian[2] and s-sparse[3] [1]. While the algorithm only produces a quantum state proportional to $|x\rangle$[4] instead of the actual solution $\vec{x}$, this still proves to be useful as we are often more interested with a function of $\vec{x}$ rather than the individual amplitudes of $\vec{b}$ [4]. Rather remarkably, the great fundamental claim of the HHL algorithm is that its runtime grows only *polylogarithmically* with the dimension of the system, N, under favorable conditions. Particularly, the HHL algorithm has a runtime on the order of:

$$O\left(\kappa^2 s^2 \log N/\epsilon\right)$$

up to polylogarithmic factors, where κ is the condition number of A[5] and ϵ is the desired solution error tolerance [1]. Thus, comparing this to the polynomial scaling of classical algorithms, we clearly see that HHL constitutes an exponential speed-up in terms of its dependence on the dimension of the system. Yet, there is of course a caveat: the performance hinges on κ and s being relatively small, which is another way of saying that this algorithm works best on well-conditioned, sparse matrices [1]. Yet, despite such nuances, the HHL algorithm is a landmark result as it was one of the first examples to demonstrate a potential exponential quantum speedup for a problem of broad practical importance, and it is also invaluable as it sparked more research into quantum algorithms for linear algebra and machine learning.

With this paper, we aim to establish a pedagogical primer to the HHL algorithm, assuming the reader has some familiarity with quantum computing, but not so much with complex quantum algorithms. Under this purpose, we therefore explain the workings of the HHL algorithm in detail and include a simple worked example to tangibly illustrate how the algorithm operates.

[1] For the unfamiliar reader, a sparse matrix can be thought of as a matrix containing many zeros.
[2] or can be embedded in a Hermitian matrix
[3] meaning each row or column of A has at most s nonzero entries
[4] the quantum encoding of the solution vector
[5] the ratio of its largest to smallest eigenvalue in magnitude

2 Scientific Motivation

As we have already outlined in the previous section, solving linear equations is a foundational task across disciplines, as many real-world computations boil down to finding $A\vec{x} = \vec{b}$ for some large matrix A. As an example, in quantum chemistry, determining the molecular energies or reaction dynamics often entails solving large eigenvalue or linear equations derived from the Schrödinger equation, and as one would expect, the required computational resources scale exponentially with molecular size, rapidly overwhelming classical computers for anything beyond the smallest molecules [5]. As another example, in computational fluid dynamics and other numerical simulations in physics, discretizing partial differential equations[6] yields enormous sparse linear systems that must be solved at each iteration [6]. Likewise, machine learning methods, specifically kernel-based techniques like Gaussian process regression and support vector machines, heavily rely on solving linear equations that involve large kernel matrices to obtain model parameters or to make predictions [7]. Hence, these are just a few examples across various fields of study that illustrate the large demand for efficient linear system solvers.

Classical algorithms, however, start struggling as the problem size N grows, as mentioned in the previous section. Thus, to simply summarize without boring the reader[7], when N is large, storing and solving linear systems on a classical computer can become intractable due to steep memory and time requirements, especially if A is ill-conditioned [1]. Therefore, the HHL algorithm is of great importance to us, as it was developed to overcome these classical bottlenecks by exploiting the parallelism of quantum states. In principle, a quantum computer can encode an N-dimensional vector $\vec{b}$ as a normalized quantum state $|b\rangle$ with only $\log_2 N$ qubits, and similarly, manipulate matrices as unitary operators [1]. Hence, Harrow, Hassidim, and Lloyd showed that given a sparse $N \times N$ matrix A and a well-behaved spectrum, a quantum circuit can produce a quantum state proportional to the solution $|x\rangle$ in polylogarithmic time, in contrast to the classical algorithms that take polynomial time for the same task. Although the theoretical speed-up promise of the HHL algorithm comes with two caveats, namely the condition on κ and sparsity, it still spurred interest in applying HHL to the massive linear systems and made the algorithm a centerpiece in arguments for the long-term impact of quantum computing. Thus, with these insights, we set the stage for the next section, where we delve into the mechanics of the HHL algorithm and unpack how each step works.

3 HHL Algorithm

Having described and motivated problem at hand, and detailed the promises of the HHL algorithm, in this section, we now thoroughly explain the HHL algorithm with the core aim of providing an accessible yet rigorous understanding to the reader.

3.1 Introduction of the Algorithm

Let us first see how we formulate the *classical* problem $A\vec{x} = \vec{b}$ in a quantum setting. We recall that A is a $N \times N$ Hermitian matrix and $\vec{x}$ and $\vec{b}$ are N-dimensional vectors. For simplicity in the explanation, we assume that $N = 2^{n_b}$, where n_b is the number of qubits we need to encode $\vec{b}$ in the quantum circuit, although this is not a strict or general occurrence. Hence, if A and $\vec{b}$ are known and $\vec{x}$ is the unknown, we know that:

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

Thus, it is intuitive that the quantum formulation / solution of the problem should resemble this equation. Hence, for this purpose, we encode the N components of the vector $\vec{b}$ as the amplitudes of $\log_2 N = n_b$ qubits, and we call these qubits the b-register. Thus, we encode $\vec{b}$ in the b-register such that:

$$|b\rangle = \sum_{i=0}^{2^{n_b}-1} \beta_i |i\rangle, \quad \beta_i = \frac{b_i}{||\vec{b}||}.$$

Noting that this provides an exponential compression compared to classical storage, we also require n additional qubits initialized to $|0\rangle^{\otimes n}$, and call this the c-register, or the clock register. This register will be responsible with

[6] via finite element or finite volume methods

[7] For our purposes, it is enough for our reader to understand that the HHL algorithm provides an exponential speed-up over the classical algorithms; readers that are interested more in how the HHL algorithm compares to classical algorithms are encouraged to review the related sections in the original paper [1].

storing information about the eigenvalues of matrix A following a subroutine called quantum phase estimation, and thus, the reader can relate this register to A until we provide further detail.

Finally, we require one more qubit, called the ancilla qubit, initialized to $|0\rangle$, which can be thought of, in a high-level way, as the qubit responsible for effectually applying A^{-1} towards the end of the algorithm.

For the HHL algorithm, we encode the N components of the vector $\vec{b}$ as the amplitudes of n_b qubits, and we call these qubits the b-register.

At this point in our introduction to the algorithm, we are still building towards the quantum formulation of the algorithm. Under this purpose, let us now remark that the matrix A, which is Hermitian, can be written in the way below through eigendecomposition:

$$A = \sum_{i=0}^{2^{n_b}-1} \lambda_i |u_i\rangle \langle u_i|$$

where u_i are its eigenvectors and λ_i are its eigenvalues. Thus, its inverse is also simply:

$$A^{-1} = \sum_{i=0}^{2^{n_b}-1} \lambda_i^{-1} |u_i\rangle \langle u_i| .$$

Moreover, we can also express $|b\rangle$ in the eigenbasis of A as:

$$|b\rangle = \sum_{j=0}^{2^{n_b}-1} b_j |u_j\rangle .$$

Following this, we see that the original equation $A\vec{x} = \vec{b}$ can be encoded as [8]:

$$|x\rangle = A^{-1} |b\rangle = \sum_{i=0}^{2^{n_b}-1} \lambda_i^{-1} b_i |u_i\rangle .$$

Simply put, the goal of the HHL algorithm is to find a solution state in this form, stored in the b-register [8]. After this is achieved, different measurements can be performed to extract information about $|x\rangle$, and consequently $\vec{x}$. It also goes without saying that all coefficients are normalized in all quantum states.

3.2 Schematic Overview of the HHL Algorithm

At its core, the HHL algorithm has 4 main components: state preparation, quantum phase estimation (QPE), controlled rotation and measurement of the ancilla bit, and inverse quantum phase estimation (QPE†), illustrated in Figure 1. Hence, following the successful implementation of these steps, we obtain a quantum state $|x\rangle$ in the b-register, holding information about the vector $\vec{x}$.

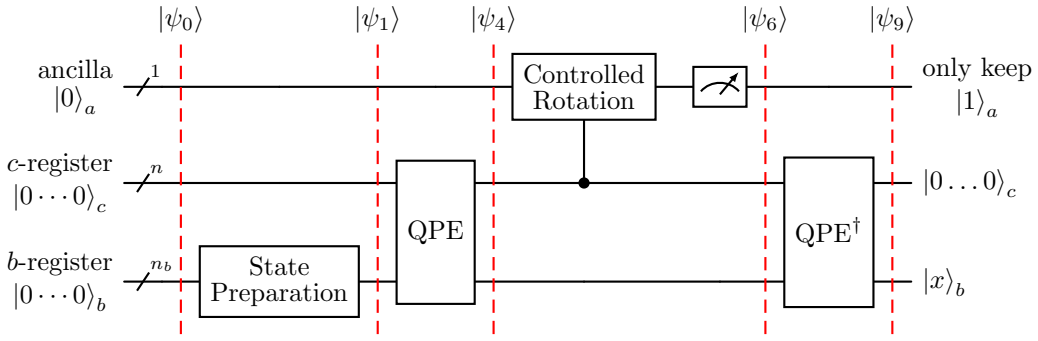

Figure 1: Circuit diagram for the implementation of HHL algorithm. States $|\psi_i\rangle$ represent the system state at the indicated steps, and are derived and discussed in detail in the following subsections.

3.3 State Preparation

The HHL algorithm employs three quantum registers: the b-register of n_b qubits to encode the input vector $\vec{b}$, the c-register of n-qubits for phase estimation, and a single ancilla qubit for conditional rotations [1]. Thus, the total system size is $n_b + n + 1$ qubits, and their initial state is:

$$|\psi_0\rangle = |0\dots0\rangle_b |0\dots0\rangle_c |0\rangle_a = |0\rangle^{\otimes n_b} |0\rangle^{\otimes n} |0\rangle .$$

After the initialization step, we also need to rotate the qubits in the b-register such that they have amplitudes corresponding to the coefficients of $\vec{b}$ [8]. Hence, to load the classical vector $\vec{b}$ into quantum amplitude form, we define the normalized state:

$$|b\rangle = \sum_{i=0}^{2^{n_b}-1} \beta_i |i\rangle , \quad \beta_i = \frac{b_i}{||\vec{b}||}$$

and implement a unitary U_b such that:

$$U_b |0\rangle^{\otimes n_b} = |b\rangle$$

Existing state preparation approaches for arbitrary state generalization generally requires $O(N)$ runtime, yet research shows that the state $|b\rangle$ can be prepared with efficient encoding schemes that operate in poly$(\log N)$ time, thus preserving its exponential speedup [9]. Given the scope of this paper, we will not concern ourselves with this and assume that the state $|b\rangle$ can be efficiently prepared with a unitary U_b.[8] Hence, after the completion of the amplitude encoding step, the full system state becomes:

$$|\psi_1\rangle = |b\rangle_b |0\dots0\rangle_c |0\rangle_a .$$

3.4 Quantum Phase Estimation (QPE)

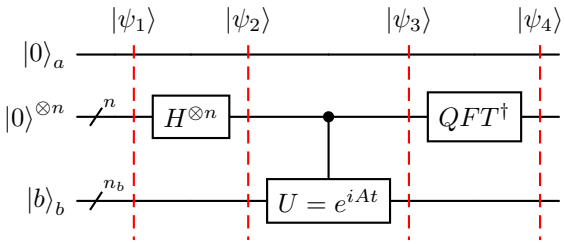

Figure 2: Quantum phase estimation circuit.

With the state preparation step completed, the next step in the HHL algorithm is to apply the subroutine called *quantum phase estimation*.[9] To put it simply, quantum phase estimation is an eigenvalue estimation procedure [8]. To be more explicit, the exact aim of the quantum phase estimation procedure is to estimate the phase ϕ in the eigenvalue equation:

$$U |u\rangle = e^{2\pi i\phi} |u\rangle$$

for a unitary operator U.[10] In the context of the HHL algorithm, phase estimation allows us to extract the eigenvalues of the Hamiltonian-simulated unitary $U = e^{iAt}$ by encoding ϕ (which is proportional to the eigenvalue of λ of A as $2\pi i\phi = i\lambda t$) into the c-register (often called the clock register), and this encoding is the basis for the controlled rotations that come after the phase estimation step [8].

Quantum phase estimation is a three step procedure that involves superposition of the c-register qubits through Hadamard gates, controlled-U operations on the b-register, and inverse quantum Fourier transform on the c-register.

[8] Although we do not dwell too much on this to not distract the main focus of the paper, the state preparation subroutine is critical as any approximation error in $|b\rangle$ directly impacts the precision of the subsequent phase estimation subroutine [10].

[9] Quantum phase estimation is a complex procedure on its own. Thus, if the reader is inexperienced with this subroutine, we highly encourage them to view the detailed explanation in the appendix.

[10] Note that as U is a unitary operator, its eigenvalues will all have a norm of 1, and that is precisely why we can express any eigenvalue of U in the exponential form, where $\phi \in [0, 1)$

In the first step of the phase estimation subroutine, we apply Hadamard gates $H^{\otimes n}$ to the n c-register qubits to obtain a superposition, such that:

$$|\psi_2\rangle = I^{\otimes n_b} \otimes H^{\otimes n} \otimes I \; |\psi_1\rangle = |b\rangle_b \otimes \frac{1}{\sqrt{2^n}} \left(|0\rangle + |1\rangle\right)_c^{\otimes n} \otimes |0\rangle_a .$$

In the second step of the procedure, controlled-U operations are applied to the state $|b\rangle$ with the clock register's qubits serving as the control qubits. Thus, it is perhaps intuitive to see that the number of the qubits in the c-register determines the number of the controlled-U operations that are to be performed. In this setup, each gate is in the form U^{2^j}, where j denotes the index of the respective c-register qubit, and the most significant qubit in the clock register, $|c_{n-1}\rangle$ controls the gate $U^{2^{n-1}}$ on the b-register, while the least significant qubit, $|c_0\rangle$, controls the gate $U^{2^0} = U$ on the b-register.

To better understand the intricacies how this process, we will begin by assuming that $|b\rangle$ is an eigenvector of $U = e^{iAt}$ with eigenvalue $e^{2\pi i\phi}$. Hence, as a standard result from linear algebra, we have that:

$$U \, |b\rangle = e^{2\pi i\phi} \, |b\rangle .$$

Now, note that when the control qubit is $|0\rangle$, $|b\rangle$ will remain unaffected. Yet, if the control qubit is $|1\rangle$, U will operate on $|b\rangle$. Thus, this is equivalent to multiplying the jth clock qubit's amplitude for $|1\rangle$ with $e^{2\pi\phi 2^j}$, as we can assign the prefactor to the controlling qubit [8]. Therefore, after the controlled-U operations, the state of the system becomes:

$$\begin{aligned} |\psi_3\rangle &= |b\rangle \otimes \left(\frac{1}{2^{n/2}} \left(|0\rangle + e^{2\pi i\phi 2^{n-1}} |1\rangle \right) \otimes \left(|0\rangle + e^{2\pi i\phi 2^{n-2}} |1\rangle \right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i\phi 2^0} |1\rangle \right) \right) \otimes |0\rangle_a \\ &= |b\rangle \; \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i\phi k} \, |k\rangle \; |0\rangle_a \end{aligned}$$

where k is the integer representation of the binary string.

Following this step, we move on to the last step of the phase estimation procedure, which is the application of the inverse quantum Fourier transform on the c-register:[11]

$$\begin{aligned} |\psi_4\rangle &= |b\rangle \; \mathrm{IQFT} \left(\frac{1}{2^{\frac{n}{2}}} \sum_{k=0}^{2^n-1} e^{2\pi i\phi k} \, |k\rangle \right) |0\rangle_a \\ &= |b\rangle \; \frac{1}{2^{\frac{n}{2}}} \sum_{k=0}^{2^n-1} e^{2\pi i\phi k} \left(\mathrm{IQFT} \, |k\rangle \right) |0\rangle_a \\ &= |b\rangle \; \frac{1}{2^n} \sum_{k=0}^{2^n-1} e^{2\pi i\phi k} \left(\sum_{y=0}^{2^n-1} e^{-2\pi i y k/N} \, |y\rangle \right) |0\rangle_a \\ &= \frac{1}{2^n} \, |b\rangle \sum_{y=0}^{2^n-1} \sum_{k=0}^{2^n-1} e^{2\pi i k(\phi - y/N)} \, |y\rangle \; |0\rangle_a . \end{aligned}$$

While this expression looks very complicated at the first glance, interference ensures that only $|y\rangle$ satisfying the condition $\phi - y/N = 0$ will have a finite amplitude of $\sum_{k=0}^{2^n-1} e^0 = 2^n$ [8]. Luckily, in any case other than this, the amplitude will be 0 due to destructive interference. Hence, we can ignore the states with zero amplitudes and rearrange $|\psi_4\rangle$ as:

$$\begin{aligned} |\psi_4\rangle &= \frac{1}{2^n} \, |b\rangle \sum_{k=0}^{2^n-1} e^{2\pi i k \cdot 0} \, |N\phi\rangle \; |0\rangle_a \\ &= |b\rangle \; |N\phi\rangle \; |0\rangle_a . \end{aligned}$$

[11] Quantum Fourier transform is also perhaps a new topic for the reader. Hence, the reader is once again highly encouraged to check the detailed explanation in the appendix.

We deem it necessary here to manually highlight that this is of extreme importance for the algorithm, as the phase estimation procedure has now allowed us to use the c-register to represent the phase information (ϕ) of U.[12] As U is related to A through $U = e^{iAt}$ and as U is diagonal in A's eigenbasis $\{u_i\}$, if $|b\rangle = |u_j\rangle$:

$$U\,|b\rangle = e^{i\lambda_j t}\,|u_j\rangle\,.$$

Thus, by the relation $i\lambda_j t = 2\pi i\phi$, we obtain $\phi = \frac{\lambda_j t}{2\pi}$, and we notice that we now have the eigenvalues of A encoded in the c-register:

$$|\psi_4\rangle = |b\rangle\ |N\phi\rangle\ |0\rangle_a = |u_j\rangle\ |N\lambda_j t/2\pi\rangle\ |0\rangle_a\,.$$

Yet, this is for the case where $|b\rangle$ is an eigenvector of U. Yet, by linearity, we can easily generalize this, and hence, in general, as $|b\rangle$ is found in a superposition of eigenvectors of A such that:

$$|b\rangle = \sum_{j=0}^{2^{n_b}-1} b_j\,|u_j\rangle$$

we can correct the state vector of the system to represent the most general case, such that:

$$|\psi_4\rangle = \sum_{j=0}^{2^{n_b}-1} b_j\,|u_j\rangle\ |N\lambda_j t/2\pi\rangle\ |0\rangle_a\,.$$

In most of the cases, the eigenvalues of λ_j are not integers [8]. Hence, we will take the liberty to choose t such that $\tilde{\lambda}_j = N\lambda_j t/(2\pi)$ yield integers. Here, we have introduced new notation, and $\tilde{\lambda}_j$ simply represents a scaled version of the original eigenvalue. With this, we complete the phase estimation algorithm, and thus arrive at the final arrangement of the state:

$$|\psi_4\rangle = \sum_{j=0}^{2^{n_b}-1} b_j\,|u_j\rangle\ |N\lambda_j t/2\pi\rangle\ |0\rangle_a = \sum_{j=0}^{2^{n_b}-1} b_j\,|u_j\rangle\ |\tilde{\lambda}_j\rangle\ |0\rangle_a\,.$$

3.5 Controlled Rotation and Measurement of the Ancilla Qubit

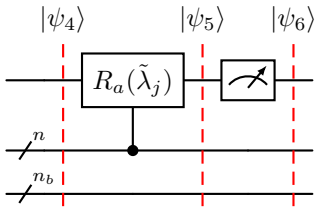

Figure 3: Controlled rotation and measurement of the ancilla qubit.

After the quantum phase estimation procedure, we move on to another critical stage for the algorithm: exploiting the eigenvalue information now encoded in the clock register through a single-qubit, eigenvalue-controlled rotation on the ancilla qubit $|0\rangle_a$. Simply put, this stage involves the rotation of the ancilla qubit based on the encoded eigenvalues in the c-register [1]. Thus, for each state $|\tilde{\lambda}_j\rangle$, we implement a controlled gate:

$$R_a(\tilde{\lambda}_j) = \exp\left[-\frac{i}{2}\theta_j Y_a\right], \quad \theta_j = 2\arcsin\left(\frac{C}{\tilde{\lambda}_j}\right),$$

where Y_a is the Pauli-Y operator acting on the ancilla and $C > 0$ is a constant that is chosen such that $|C/\tilde{\lambda}_j| \leq 1$ for every j [4]. Hence, this gate multiplies the $|1\rangle_a$ component of each term by the factor $C/\tilde{\lambda}_j$, while it leaves the $|0\rangle_a$ component unchanged, yielding the system state:

$$|\psi_5\rangle = \sum_{j=0}^{2^{n_b}-1} b_j\,|u_j\rangle\ |\tilde{\lambda}_j\rangle\left(\sqrt{1-\frac{C^2}{\tilde{\lambda}_j^2}}\,|0\rangle_a + \frac{C}{\tilde{\lambda}_j}\,|1\rangle_a\right).$$

[12] The accuracy of this information depends on the number of qubits used in the b-register.

This is another milestone in the algorithm, and we will now explain why this is such a useful *trick*. The crux of it lies in the fact that we measure the ancilla qubit after this rotation. Specifically, when we measure the ancilla qubit following this rotation, its state will either collapse to $|0\rangle_a$ or $|1\rangle_a$. In the case that it collapses to $|0\rangle_a$, we discard this result and repeat the computation until the state collapses to $|1\rangle_a$ [1]. Thus, the final state that we are interested in will be:

$$|\psi_6\rangle = \frac{1}{\sqrt{\sum_{j=0}^{2^{n_b}-1} |\frac{b_j C}{\tilde{\lambda}_j}|^2}} \sum_{j=0}^{2^{n_b}-1} b_j \, |u_j\rangle \, |\tilde{\lambda}_j\rangle \, \frac{C}{\tilde{\lambda}_j} \, |1\rangle_a$$

where the prefactor is added due to the normalization condition following the measurement. As $|\frac{C}{\tilde{\lambda}_j}|^2$ is the probability of obtaining the state $|1\rangle_a$ when we measure the ancilla qubit, we notice that we should choose C to be as large as possible [8]. Moreover, we can now see why the ancilla rotation and measurement proves to be so useful when we compare this to the equation:

$$|x\rangle = \sum_{i=0}^{2^{n_b}-1} \lambda_i^{-1} b_i \, |u_i\rangle \, .$$

Clearly, $|\psi_6\rangle$ resembles this answer that we are looking for. However, we can only obtain the correct result if we measure the b-register in the eigenbasis of A instead of the computational basis[8]. Yet, in this given state, b-register is entangled with the clock qubits $|\tilde{\lambda}_j\rangle$ (which was a result of the phase estimation procedure), and thus we cannot factorize $|\psi_6\rangle$ into a tensor product of the b-register and the c-register [8]. Therefore, we cannot convert the b-register into the computational basis with the desired amplitudes, and this implies that we need to *uncompute* the system state such that it yields the desired results in the computational basis, and in which the two registers will be unentangled again.

Here, before moving forward with the uncomputation step, we should perhaps also mention that although the measurement of the ancilla qubit is usually performed after uncomputation, for simplicity in the derivation, we perform it before the uncomputation step. This is a perfectly legal move, as the ancilla bit is not involved in any operations after the controlled rotation [8].

3.6 Uncomputation (Inverse QPE)

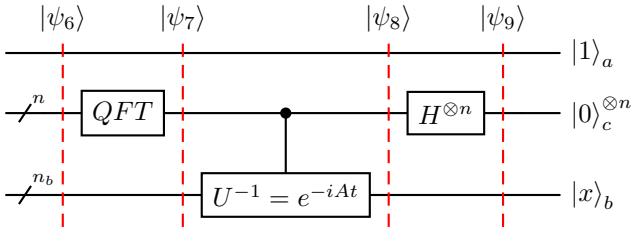

Figure 4: Inverse quantum phase estimation circuit.

Following the operations on the ancilla qubit, we now perform the last step of the HHL algorithm, namely uncomputation, and our goal with this step is to simply unentangle the two registers. Hence, in this stage of the algorithm, we run an inverse quantum phase estimation procedure so that we map every $|\tilde{\lambda}_j\rangle$ back to $|0\rangle^{\otimes n}$. The logic here is very much intuitive; we quite literally reverse everything we did on the state in the quantum estimation procedure.

Firstly, we apply the (this time forward) quantum Fourier transformation to the c-register such that:

$$\begin{aligned} |\psi_7\rangle &= \frac{1}{\sqrt{\sum_{j=0}^{2^{n_b}-1} \left|\frac{b_j C}{\tilde{\lambda}_j}\right|^2}} \sum_{j=0}^{2^{n_b}-1} \frac{b_j C}{\tilde{\lambda}_j} \, |u_j\rangle \, \text{QFT} \, |\tilde{\lambda}_j\rangle \, |1\rangle_a \\ &= \frac{1}{\sqrt{\sum_{j=0}^{2^{n_b}-1} \left|\frac{b_j C}{\tilde{\lambda}_j}\right|^2}} \sum_{j=0}^{2^{n_b}-1} \frac{b_j C}{\tilde{\lambda}_j} \, |u_j\rangle \left(\frac{1}{2^{\frac{n}{2}}} \sum_{y=0}^{2^{n-1}} e^{2\pi i y \tilde{\lambda}_j / N} \, |y\rangle \right) |1\rangle_a \end{aligned}$$

Following this, we then move on with reversing the controlled-U operations that we applied previously. To achieve this, we apply inverse controlled-U operations on the b-register, controlled by the clock qubits, with $U^{-1} = e^{-iAt}$. To put it in an intuitive way, as the forward block multiplied each eigencomponent $|u_j\rangle$ by a phase factor $e^{i\lambda_j ty}$, the backward block must multiply the same component by the complex conjugate phase $e^{-i\lambda_j ty}$ [11]. Thus, the way we implement this is to let every clock qubit control a power of U^{-1} such that when those clock qubits are in the state $|y\rangle$, the net controlled operation is U^{-y}. Therefore, with this step, the system state becomes:

$$|\psi_8\rangle = \frac{1}{2^{\frac{n}{2}}\sqrt{\sum_{j=0}^{2^{n_b}-1}\left|\frac{b_j C}{\tilde{\lambda}_j}\right|^2}} \sum_{j=0}^{2^{n_b}-1} \frac{b_j C}{\tilde{\lambda}_j} |u_j\rangle \left(\sum_{y=0}^{2^n-1} e^{-i\lambda_j ty} e^{2\pi i y\tilde{\lambda}_j/N} |y\rangle \right) |1\rangle_a$$

As we set $\tilde{\lambda}_j = N\lambda_j t/(2\pi)$ previously, the two exponential terms annihilate each other, and we are left with:

$$\begin{aligned} |\psi_8\rangle &= \frac{1}{2^{\frac{n}{2}}\sqrt{\sum_{j=0}^{2^{n_b}-1}\left|\frac{b_j C}{\tilde{\lambda}_j}\right|^2}} \sum_{j=0}^{2^{n_b}-1} \frac{b_j C}{\tilde{\lambda}_j} |u_j\rangle \sum_{y=0}^{2^n-1} |y\rangle\ |1\rangle_a \\ &= \frac{C}{2^{\frac{n}{2}}\sqrt{\sum_{j=0}^{2^{n_b}-1}\left|\frac{b_j C}{\tilde{\lambda}_j}\right|^2}} |x\rangle_b \sum_{y=0}^{2^n-1} |y\rangle\ |1\rangle_a . \end{aligned}$$

Now that the c- and b-registers are finally unentangled and we have $|x\rangle$ stored in the b-register, we apply the Hadamard gates on the c-register as our final step, and obtain:

$$\begin{aligned} |\psi_9\rangle &= \frac{1}{\sqrt{\sum_{j=0}^{2^{n_b}-1}\left|\frac{b_j C}{\lambda_i}\right|^2}} \sum_{j=0}^{2^{n_b}-1} \frac{b_j C}{\lambda_j} |u_j\rangle\ |0\rangle^{\otimes n}\ |1\rangle_a \\ &= \frac{C}{\sqrt{\sum_{j=0}^{2^{n_b}-1}\left|\frac{b_j C}{\lambda_i}\right|^2}} |x\rangle_b\ |0\rangle_c^{\otimes n}\ |1\rangle_a \end{aligned}$$

which is the desired solution that we were looking for [8]. Moreover, if C is real, and by using the fact that:

$$\sum_{j=0}^{2^{n_b}-1} \left|\frac{b_j}{\lambda_i}\right|^2 = 1$$

we obtain:

$$\boxed{|\psi_9\rangle = |x\rangle_b\ |0\rangle_c^{\otimes n}\ |1\rangle_a}$$

Hence, the solution $|x\rangle$ is stored in the b-register successfully, and this concludes the HHL algorithm.

4 HHL Algorithm in Action

4.1 Problem Instance and Classical Solution

Now that we have completed the derivation of the HHL algorithm, we want to translate this into a concrete demonstration on a simple system, with the goal of building intuition in the reader. For this goal, we will consider the linear system $A\vec{x} = \vec{b}$, where:

$$A = \begin{pmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{pmatrix}, \quad \vec{b} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Hence, we are seeking the solution $\vec{x} = A^{-1}\vec{b}$. Classically, it is easily verified that:

$$A^{-1} = \frac{4}{3}\begin{pmatrix} 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 \end{pmatrix} \quad \text{and} \quad A^{-1}\vec{b} = \vec{x} = \begin{pmatrix} 4/3 \\ -2/3 \end{pmatrix}$$

and therefore, our goal with the HHL algorithm is to obtain a normalized state $|x\rangle = \alpha_1|0\rangle + \alpha_2|1\rangle$ such that $\alpha_1 \propto 4/3$ and $\alpha_2 \propto -2/3$. Thus, with this example system, we will walk through every step of the algorithm, aiming to provide the reader a better understanding of the intricacies of this rather complicated routine.

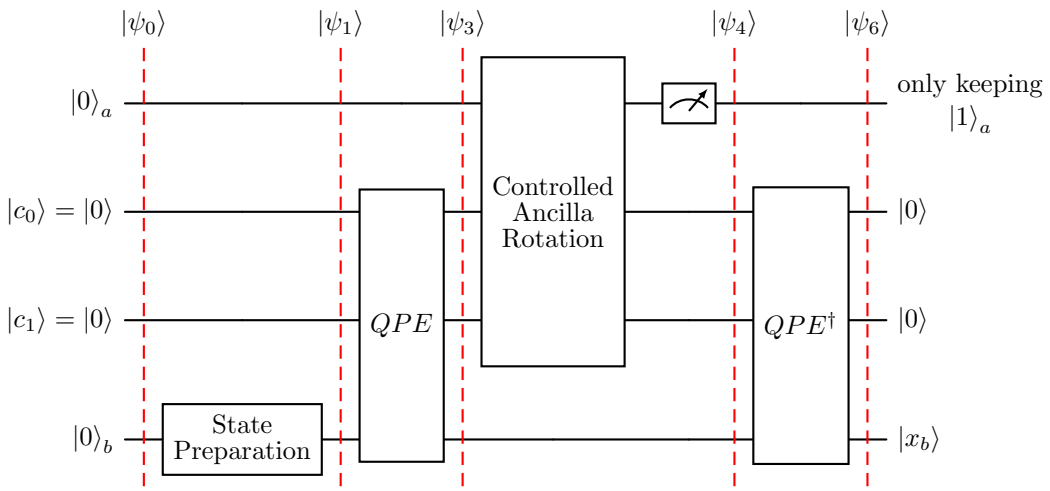

Figure 5: 4-qubit HHL algorithm circuit to implement our example, which includes one qubit in the b-register, 2 qubits in the c-register, and one ancilla qubit, all initialized to $|0\rangle$.

4.2 Preliminaries: Eigendecomposition of A & $\vec{b}$ in the Eigenbasis

Let us first analyze A in its eigenbasis. A quick calculation shows that the eigenvalues of A are $\lambda_1 = \frac{3}{2}$ and $\lambda_2 = \frac{1}{2}$. Additionally, we also see that the corresponding normalized eigenvectors in the computational basis are:

$$|u_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{and} \quad |u_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

We also see that as A is diagonalizable, by eigendecomposition:

$$A = \lambda_1 |u_1\rangle \langle u_1| + \lambda_2 |u_2\rangle \langle u_2| \quad \text{and} \quad A^{-1} = \lambda_1^{-1} |u_1\rangle \langle u_1| + \lambda_2^{-1} |u_2\rangle \langle u_2| .$$

We now express $\vec{b}$ in this eigenbasis. Let us notice that $\vec{b}$ exactly corresponds to $|0\rangle$. Thus, this means that we can express $\vec{b}$ in the eigenbasis of A as:

$$|b\rangle = \frac{1}{\sqrt{2}} |u_1\rangle + \frac{1}{\sqrt{2}} |u_2\rangle .$$

This also aligns perfectly with the general result; if $|b\rangle = \sum_j b_j |u_j\rangle$, then the normalized solution is:

$$|x\rangle = A^{-1} |b\rangle = \sum_{i=0}^{N-1} \lambda_i^{-1} b_i |u_i\rangle .$$

We see that for our case, $b_1 = b_2 = \frac{1}{\sqrt{2}}$, so we can infer that:

$$|x\rangle \propto \frac{b_1}{\lambda_1} |u_1\rangle + \frac{b_2}{\lambda_2} |u_2\rangle = \frac{1}{3\sqrt{2}} |u_1\rangle + \sqrt{2} |u_2\rangle .$$

When we convert back to the computational basis, this yields:

$$|x\rangle \propto \frac{4}{3} |0\rangle - \frac{2}{3} |1\rangle$$

which correctly aligns with the classical solution $\vec{x} = (4/3, -2/3)^T$. Hence, our goal with the HHL is simply to prepare this state $|x\rangle$ in the quantum register such that we encode the solution vector's components as the amplitudes of $|0\rangle$ and $|1\rangle$. In the following sections, we thus detail every step of the algorithm and see how this is achieved.

4.3 State Preparation

As we have outlined in the explanation of the HHL algorithm, we require three registers, an n_b qubit b-register for the vector encoding, an n qubit c-register (clock register) for phase estimation, and a single ancilla qubit. In our example, as $\vec{b}$ is 2-dimensional, $n_b = 1$ (since $N_b = 2^{n_b}$) and $n = 2$ as we will use 2 qubits of precision for the phase estimation procedure. Hence, the initial state of the system is:

$$|\psi_0\rangle = |0\dots0\rangle_b \otimes |0\dots0\rangle_c \otimes |0\rangle_a = |0\rangle_b |00\rangle_c |0\rangle_a .$$

Now, we must load $\vec{b}$ into the b-register, but in our example, this step is trivial as $\vec{b}$ already corresponds to the basis state $|0\rangle_b$, such that the b-register is already $|b\rangle = |0\rangle_b$. Yet, it is important to note once again that in general, state preparation would involve a unitary U_b. As an example, if $\vec{b} = (\beta_0, \beta_1)^T$, we could have applied a single qubit rotation such that $R_y(\theta)|0\rangle = \beta_0|0\rangle + \beta_1|1\rangle$ to load $\vec{b}$ into the b-register. Thus, after the trivial step of loading $\vec{b}$, our system state is:

$$|\psi_1\rangle = |b\rangle \otimes |00\rangle_c |0\rangle_a = |0\rangle_b \otimes |00\rangle_c |0\rangle_a .$$

4.4 Quantum Phase Estimation

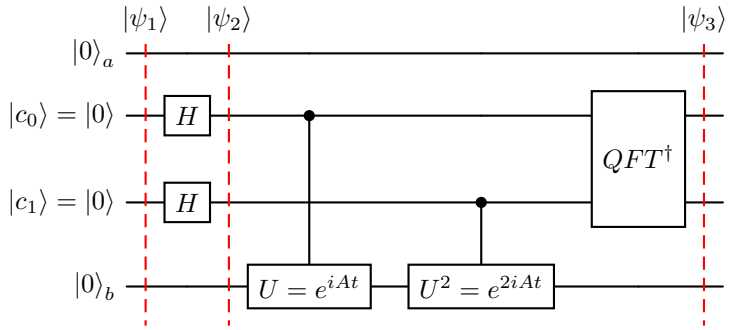

Figure 6: Quantum phase estimation procedure for the 4-qubit example. The c-register qubits are labeled with c_1, c_0.

Following the state preparation step, the next stage in the HHL algorithm concerns quantum phase estimation. Here, to put it very simply, our goal is to encode the eigenvalues λ_i into the c-register so that later gates can access λ_i^{-1}. Hence, as the first step of the phase estimation subroutine, the two clock qubits c_1, c_0 are put into an equal superposition state by Hadamard gates, which, for $n = 2$ yields:

$$\begin{aligned}
|\psi_2\rangle &= |0\rangle_b \otimes \left(H^{\otimes 2}|00\rangle_c\right) \otimes |0\rangle_a \\
&= |0\rangle_b \otimes \left(\frac{1}{2}\sum_{k=0}^{3}|k\rangle_c\right) \otimes |0\rangle_a \\
&= |0\rangle_b \otimes \frac{1}{2}\left(|00\rangle_c + |01\rangle_c + |10\rangle_c + |11\rangle_c\right) \otimes |0\rangle_a .
\end{aligned}$$

where k is the integer representation of the binary string c_1c_0.

After this, the next step in the phase estimation procedure is to perform the controlled-U operations, with the goal of entangling the c-register with the b-register according to the eigenvalues. Specifically, we treat the Hermitian matrix A as a Hamiltonian to define a unitary $U = e^{iAt}$, which will then act on the b-register. As explained in the previous section, each clock qubit c_j controls the application of U^{2^j} on the b register, and in our case, we thus have two controlled-U operations on the b-register: the most significant clock qubit c_1 controls the application of $U^{2^1} = U^2 = e^{2iAt}$ and the least significant clock qubit c_0 controls the application of $U^{2^0} = U = e^{iAt}$.

Here t is formally called the time evolution parameter t [8]. Yet, as $A = I + \frac{1}{2}X$ (where X is our Pauli-X operator) in our example, we can think of the unitary e^{iAt} as an x-axis rotation on the b-register qubit by an the angle t. Thus, recalling the relationship $2\pi i\phi = i\lambda t$, we set $t = \pi$ so that $\phi_j = \lambda_j t/(2\pi) = \{\frac{3}{4}, \frac{1}{4}\}$. That is, we choose t in such a way that ϕ_j correspond to simple rational fractions that we can exactly represent with our 2-qubit c-register.

Following this, we now proceed to applying the controlled unitaries U and U^2 on the b-register qubit. For this, we first apply $U = e^{iA\pi}$, controlled by c_0, and then apply $U^2 = e^{iA(2\pi)}$. To understand this better, keeping in mind that we can express $|0\rangle_b$ as a linear combination of the common eigenstates $|u_j\rangle$ of A and $U = e^{iAt}$, we first observe the effect of the unitaries and the inverse quantum Fourier transform on $|u_j\rangle$ and then derive the two procedures' effect on the actual system state by linearity. Hence, for explanatory purposes, assuming $|b\rangle_b = |u_j\rangle$, we see that under these operations $|u_j\rangle$ picks up a phase $e^{i\lambda_j t}$, written onto the clock register, and the system state after these two operations become:

$$|u_j\rangle_b \otimes \left(\frac{1}{\sqrt{2^2}}\left(|0\rangle + e^{2(2\pi i\phi)}|1\rangle\right) \otimes \frac{1}{\sqrt{2^2}}\left(|0\rangle + e^{2\pi i\phi}|1\rangle\right)\right) \otimes |0\rangle_a = |u_j\rangle \otimes \left(\frac{1}{2}\sum_{k=0}^{3} e^{2\pi i\phi_j k}|k\rangle_c\right) \otimes |0\rangle_a .$$

With the same assumption $|b\rangle_b = |u_j\rangle$ active, we see that applying an inverse quantum Fourier transform on the c-register would then yield:

$$|u_j\rangle_b \otimes |\phi_j 2^n\rangle_c \otimes |0\rangle_a$$

which means that the eigenvalues of A are now encoded in the c-register as $\phi_j = \lambda_j t/(2\pi)$. Now that we have seen the effect of the two controlled unitaries and the inverse quantum Fourier transform on the eigenbases $|u_j\rangle$ of A, and given that $|b\rangle = b_1|u_1\rangle + b_2|u_2\rangle$ by linearity, we can conclude that after the controlled-U operations and inverse quantum Fourier transformation, we obtain the system state:

$$|\psi_3\rangle = b_1|u_1\rangle_b|4\phi_1\rangle_c|0\rangle_a + b_2|u_2\rangle_b|4\phi_2\rangle_c|0\rangle_a = b_1|u_1\rangle|11\rangle_c|0\rangle_a + b_2|01\rangle_c|0\rangle_a$$

where the c-register and b-register are now tangled and $b_1 = b_2 = \frac{1}{\sqrt{2}}$ as before. Most importantly, we highlight that we now have the scaled eigenvalue information encoded into the c-register as $|\tilde{\lambda}_1\rangle = |11\rangle_c$ and $|\tilde{\lambda}_2\rangle = |01\rangle_c$. Thus, this marks the completion of the phase estimation procedure, but now the caveat is that the c-register is entangled with the b-register.

4.5 Eigenvalue Inversion via Ancilla Rotation

Now that we have our $\tilde{\lambda}_j$ scaled eigenvalues encoded in the c-register with quantum phase estimation, we continue with the next stage of the algorithm: inverting these eigenvalues. Hence, we simply achieve this by performing a rotation on the ancilla qubit based on the encoded eigenvalues in the c-register. Hence, as we have explained previously, for each state $|\tilde{\lambda}_j\rangle$, we implement a controlled gate:

$$R_a(\tilde{\lambda}_j) = \exp\left[-\frac{i}{2}\theta_j Y_a\right], \quad \theta_j = \arcsin\left(\frac{C}{\tilde{\lambda}_j}\right).$$

In practice, this corresponds to a rotation $R_y(2\theta_j)$ such that $\theta_j = \arcsin(C/\tilde{\lambda}_j)$ for a normalization constant C of our choice. For convenience, we pick $C = \min(\tilde{\lambda}_j) = 1$ so that $\sin\theta_j$ never exceeds 1. Thus, with $\tilde{\lambda_1} = 3$ and $\tilde{\lambda}_2 = 1$ we obtain:

$$2\theta_1 = 2\arcsin\frac{1}{3} \quad \text{and} \quad 2\theta_2 = \pi.$$

Essentially, this means that when the c-register is in the state $|11\rangle_c$ that corresponds to $|\tilde{\lambda}_1\rangle$, we apply $R_y(2\theta_1)$ on the ancilla and when the c-register is in the state $|01\rangle_c$ that corresponds to $|\tilde{\lambda}_2\rangle$, we apply $R_y(\pi)$ on the ancilla. Hence, to implement this, we use two controlled rotations on the ancilla, and with following these rotations, the system state thus becomes:

$$\begin{aligned}
|\psi_4\rangle &= \sum_{j=0}^{1} b_j|u_j\rangle|\tilde{\lambda}_j\rangle(\cos\theta_j|0\rangle_a + \sin\theta_j|1\rangle_a) \\
&= b_1|u_1\rangle_b|11\rangle_c\left(\sqrt{1-\frac{C^2}{\tilde{\lambda}_1^2}}|0\rangle_a + \frac{C}{\tilde{\lambda}_1}|1\rangle_a\right) + b_2|u_2\rangle_b|01\rangle_c\left(\sqrt{1-\frac{C^2}{\tilde{\lambda}_2^2}}|0\rangle_a + \frac{C}{\tilde{\lambda}_2}|1\rangle_a\right) \\
&= \frac{1}{\sqrt{2}}|u_1\rangle|11\rangle_c\left(\frac{2\sqrt{2}}{3}|0\rangle + \frac{1}{3}|1\rangle\right)_a + \frac{1}{\sqrt{2}}|u_2\rangle|01\rangle_c(0\cdot|0\rangle + 1\cdot|1\rangle)_a .
\end{aligned}$$

Here, we urge the reader to notice how the amplitudes of $|1\rangle_a$ terms are now proportional to $\frac{b_j}{\lambda_j}$ respectively, as this is what motivates us to measure the ancilla following the rotation and only retaining information when it collapses to $|1\rangle_a$.

Moreover, at this stage, following the controlled rotation, if we were to measure the ancilla qubit, there is a certain probability that it will be found in $|1\rangle_a$, and with these amplitudes, we can now calculate this probability. Thus, only in these successful cases the b-register will collapse to the desired solution state, as discussed in the explanation of the algorithm. This is equivalent to saying that if the ancilla collapses to the state $|0\rangle_a$, we discard the outcome and try again until we obtain the $|1\rangle_a$ state collapse. This is called post-selection, and is extremely important for the successful completion of the algorithm [8]. With a quick calculation, we see that in our example, the success probability is $P(|1\rangle_a) = \left(\frac{1}{\sqrt{2}}\right)^2 \left(\frac{1}{3}\right)^2 + \left(\frac{1}{\sqrt{2}}\right)^2 (1)^2 = 5/9 \approx 55.6\%$, which means that out of a very large number of trials that we conduct, in 55.6% of them, we expect the ancilla qubit to collapse to the state $|1\rangle_a$.

With these established, assuming that the measurement of the ancilla bit yields $|1\rangle_a$, we move on with the final step of the algorithm.

4.6 Inverse QPE (Uncomputation)

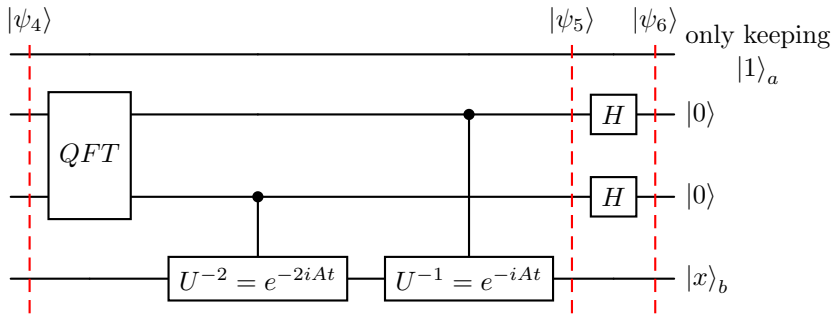

Figure 7: Inverse quantum phase estimation procedure for the 4-qubit example.

Following the successful application of the previous steps, we now apply the final stage of the HHL algorithm: applying inverse quantum Fourier transform on the c-register. Simply put, the purpose of this is to disentangle and reset the c-register, so that the b-register and the ancilla are left in a standalone state. Thus, what we do here is essentially the inverse of the earlier phase estimation procedure. To achieve this, we first apply a forward quantum Fourier transform on the c-register, then undo the controlled-U operations, and finally apply Hadamard gates on the clock qubits. While the mathematical expressions that we derived step by step in the explanation of the algorithm seem very complex, the general idea is thus pretty intuitive.

Essentially, the three steps all serve to remap every $|11\rangle_c, |01\rangle_c$ to $|00\rangle_c$ and therefore to unentangle the c- and b-registers Hence, while we are not going to get lost in mathematical complexities of these steps, after applying forward quantum Fourier transform and inverse controlled unitaries on the c-register, we obtain the state:

$$
\begin{aligned}
|\psi_5\rangle &= \frac{1}{2^{\frac{n}{2}}\sqrt{\sum_{j=0}^{2^{n_b}-1}\left|\frac{b_j C}{\tilde{\lambda}_j}\right|^2}} \sum_{j=0}^{2^{n_b}-1} \frac{b_j C}{\tilde{\lambda}_j} |u_j\rangle \sum_{y=0}^{2^n-1} |y\rangle\ |1\rangle_a \\
&= \frac{1}{2^{n/2}} |x\rangle_b \sum_{y=0}^{2^n-1} |y\rangle\ |1\rangle_a \\
&= \frac{1}{2\sqrt{|\frac{1}{\sqrt{2}}\frac{1}{3}|^2 + |\frac{1}{\sqrt{2}}\frac{1}{1}|^2}} \left(\frac{b_1 C}{\tilde{\lambda}_1} |u_1\rangle + \frac{b_2 C}{\tilde{\lambda}_2} |u_2\rangle\right)_b (|00\rangle_c + |01\rangle_c + |10\rangle_c + |11\rangle_c)\ |1\rangle_a \\
&= \frac{1}{\sqrt{\frac{5}{9}}} \left(\frac{1}{3\sqrt{2}} |u_1\rangle + \frac{1}{\sqrt{2}} |u_2\rangle\right)_b \frac{1}{2} (|00\rangle_c + |01\rangle_c + |10\rangle_c + |11\rangle_c)\ |1\rangle_a .
\end{aligned}
$$

With this step, we note that the c- and b-registers are finally unentangled, and we have $|x\rangle$ stored in the b-register. As our final step, following the application of Hadamard gates on our c-register, we obtain:

$$\begin{aligned}|\psi_6\rangle &= \frac{1}{\sqrt{5/9}} |x\rangle_b |00\rangle_c |1\rangle_a \\ &= \frac{3}{\sqrt{5}} \left(\frac{1}{3\sqrt{2}} |u_1\rangle + \frac{1}{\sqrt{2}} |u_2\rangle \right)_b |00\rangle_c |1\rangle_a\end{aligned}$$

where our normalization constant is $3/\sqrt{5}$, and we thus complete the algorithm.

4.7 Interpretation and Verification

Now that we have completed the HHL algorithm, as a final step, we aim to verify our results, first analytically, and then by employing a Qiskit simulation of our circuit.

Let us first start by noting that as:

$$|u_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{and} \quad |u_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle),$$

we can further simply the final state $|\psi_6\rangle$ in the computational basis. Hence, substituting the expressions of $|u_1\rangle$ and $|u_2\rangle$ in the computational basis yields:

$$\begin{aligned}|\psi_6\rangle &= \frac{3}{\sqrt{5}} \left(\frac{1}{3\sqrt{2}} \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) + \frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \right)_b |00\rangle_c |1\rangle_a \\ &= \frac{3}{\sqrt{5}} \left(\left(\frac{1}{6} + \frac{1}{2} \right) |0\rangle + \left(\frac{1}{6} - \frac{1}{2} \right) |1\rangle \right)_b |00\rangle_c |1\rangle_a \\ &= \frac{3}{\sqrt{5}} \left(\frac{4}{6} |0\rangle + -\frac{2}{6} |1\rangle \right)_b |00\rangle_c |1\rangle_a \\ &= \left(\frac{4}{\sqrt{20}} |0\rangle - \frac{2}{\sqrt{20}} |1\rangle \right) |00\rangle_c |1\rangle_a .\end{aligned}$$

Hence, we see that as we expected, we have $\alpha_1 \propto \frac{4}{3}$ and $\alpha_2 \propto -\frac{2}{3}$ such that:

$$|x\rangle \propto \frac{4}{3} |0\rangle - \frac{2}{3} |1\rangle$$

and this implies that we have found amplitudes that are proportional to the actual amplitudes of the vector $\vec{x}$, and the probability of obtaining $|0\rangle$ and $|1\rangle$ when the b-register is measured is $4:1$, as one would expect. We now simulate our circuit with Qiskit for purposes of further verification, and present the results below.

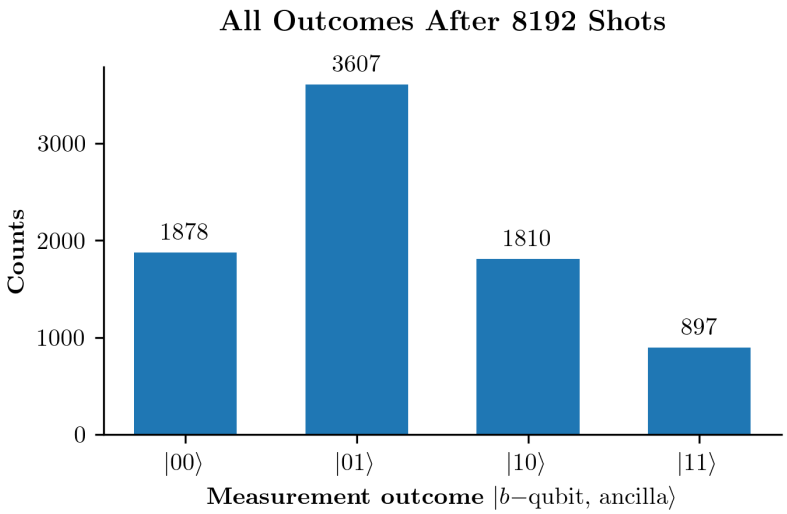

Figure 8: Qiskit simulation results of the circuit with 8192 shots. Only the b-qubit and the ancilla qubit are measured.

Through the measurement results that we present in the histogram above, we first verify the success rate of our circuit, that is, the probability of obtaining $|1\rangle_a$ when measuring the ancilla qubit. Given that the most significant qubit is the b-qubit and the least significant qubit is the ancilla qubit in the histogram's measurement outcome notation, we see that in $3607 + 897 = 4504$ instances out of 8192 shots, we measure the ancilla as $|1\rangle_a$. Thus, this translates to a success probability of $0.5498 \approx 0.55 = 55\%$, and thus, this is close enough to the success probability of 55.6% that we analytically found.

Moreover, and perhaps most importantly, we now want to verify that the probability of obtaining $|0\rangle$ and $|1\rangle$ in the b-qubit is $4:1$, respectively. Hence, out of the successful instances, which are $|01\rangle$ and $|11\rangle$ as represented on the histogram, we see that 3607 of them collapse to $|0\rangle_b$ and 897 of them collapse to $|1_b\rangle$. Thus, their ratio is

$$\frac{3607}{897} \approx 4.0212$$

verifying the analytical prediction that the algorithm has led us to make and empirically demonstrating how we obtain information on the vector $\vec{x}$ from the quantum state $|x\rangle$.

5 Outlook

As we come to an end, we would also like to provide an outlook on the HHL algorithm, primarily to emphasize its importance in the field and to help the reader probe into future developments. Looking forward, the least we can say is that the HHL algorithm has already spurred a rich ecosystem of quantum linear system methods, and we believe that this is likely to evolve significantly as both theory and hardware advance.

As we have just derived ourselves, the original HHL algorithm has made a milestone in the field of quantum linear solvers, proving that quantum computers can, in principle, invert sparse Hermitian matrices in polylogarithmic time, an exponential speed-up over classical methods. Yet, with rapid developments in the field, its key ideas like eigenvalue controlled rotations and post-selection now underpin a broader, even more efficient toolkit. As an example, it has caught our attention that quantum singular value transformation (QSVT) and block-encoding techniques have emerged as a unifying paradigm for the future of quantum linear algebra, building on the ideas of the original algorithm while subsuming it as a special case [12]. On the other hand, we have also seen that recent methods based on discrete adiabatic evolution and quantum walks achieve a linear dependence on the condition number κ, a great improvement compared to HHL's quadratic scaling [13]. Hence, together, such results suggest us that future fault-tolerant computers will perhaps run QSVT-based or adiabatic solvers rather than the original HHL circuit, primarily reserving HHL as a pedagogical and historical prototype.

Pivoting to the hardware front, we see that realizing HHL's promised speedups still remain challenging, yet we are starting to see some concrete progress. Unfortunately, the complete HHL circuit requires deep phase estimation and many ancilla qubits, which is beyond the capabilities of the current Noisy Intermediate Scale Quantum (NISQ) computers [14]. As a result, hybrid and variational approaches have been developed to make quantum linear algebra more NISQ-friendly. Yet, such modified algorithms typically trade away the full asymptotic speedup, and thus cheat a little on the promises of the original algorithm [14]. Thus, looking further ahead, we can say that truly fault-tolerant quantum computers will be needed to harness the true power of the HHL algorithm. However, it also seems like by the time large scale fault tolerant hardware becomes available, the field will favor QSVT-based or adiabatic solvers rather than the original HHL protocol, as these methods perform better in various metrics [12].

It is no wonder that the HHL algorithm has been a pioneer in an in-progress paradigm shift in scientific computing across various fields. Thus, we believe that HHL's legacy lies less in the specific circuit we implement than in the conceptual template it provided. New and better theoretical methods that we outlined all instantiate that template, and as hardware scales from NISQ to early fault tolerant devices, we can expect the techniques that were inspired by the HHL algorithm to integrate seamlessly into larger quantum workflows across disciplines. Delivering this speedup will demand simultaneous progress in all aspects of quantum computing, but the trajectory seems clear to us: quantum linear systems algorithms are moving from foundational theory toward practical, application-driven quantum software stacks.

6 Conclusion

In this paper, we aimed to create an accessible yet rigorous primer for the HHL algorithm, with the high hopes of simplifying a complex quantum algorithm for the reader. For this purpose, we have introduced the problem,

motivated it, derived the algorithm step by step, validated it with a 4-qubit example, and provided an outlook on the future of the field. Together, these elements show concretely how quantum resources can compress an N-dimensional system to polylogarithmic depth, highlighting the practical promise (and current limits) of quantum linear algebra techniques.

Appendix

A Quantum Fourier Transform

A very powerful way of solving a problem in mathematical sciences is to transform it into some other problem for which a solution is already known [15]. Hence, one such transformation is the discrete Fourier transformation, which takes as input an N-dimensional vector of complex numbers, $\vec{x} = (x_0, x_1, \ldots, x_{N_1})^T$, and outputs the transformed data, an N-dimensional vector of complex numbers, $\vec{y} = (y_0, y_1, \ldots, y_{N-1})^T$, defined by:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i jk/N}.$$

Thus, the quantum Fourier transform is the quantum analog of the discrete Fourier transform, where the main idea is to perform the exact same transformation but in a quantum state. The quantum Fourier transform on an orthonormal basis $\{|0\rangle, \ldots |N-1\rangle\}$ is hence defined to be a linear transformation with the following action on the basis states [15]:

$$|j\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i jk/N} |k\rangle .$$

Equivalently, the action on an arbitrary state of our choice may thus be written as:

$$\sum_{j=0}^{N-1} x_j |j\rangle \longrightarrow \sum_{k=0}^{N-1} y_k |k\rangle ,$$

where the amplitudes y_k are exactly the discrete Fourier transform of the amplitudes x_j. While it is not immediately obvious from this definition, we must note that the quantum Fourier transformation is a unitary transformation.

B Quantum Phase Estimation

In the previous section of the appendix, we have extensively described the QFT, mainly as the Fourier transform lies at the heart of another general procedure known as phase estimation, which in turn is central for many other quantum algorithms [15]. To understand the quantum phase estimation algorithm, let us consider a unitary operator U, and suppose that U has an eigenvector v with the corresponding eigenvalue $e^{2\pi i \phi_v}$, where we do not know the value of ϕ_v. Hence, the goal of the quantum phase estimation algorithm is to simply estimate ϕ_v.

In order to perform the quantum phase estimation procedure, we need two registers. Our first register has t qubits initialized to state $|0\rangle$, and we have freedom over how many qubits our first register should include. Thus, how we choose t depends on two concerns: the accuracy we wish to have in our estimate of ϕ, and with what probability we wish the phase estimation to be successful [15]. Moreover, our second register begins in the state $|v\rangle$, and contains as many qubits as needed to store $|v\rangle$.

The first stage of the quantum phase estimation procedure consists of two stages. First, we apply a Hadamard gate to each qubit in the first register, and this is followed by the application of controlled-U operations on the second register, with U raised to successive powers of two [15]. Let us now detail the first stage of the phase estimation procedure. At first, we have the state:

$$|\psi_0\rangle = |0\rangle^{\otimes t} |v\rangle .$$

After applying a t-bit Hadamard gate operation $H^{\otimes t}$ on the first register, also called as the counting register, we obtain:

$$|\psi_1\rangle = \frac{1}{\sqrt{2^t}} (|0\rangle + |1\rangle)^{\otimes t} |v\rangle .$$

Following this step, we introduce the controlled-U operation that applies our unitary operator U to the second register only if its corresponding control bit is $|1\rangle$. As U is a unitary operator with eigenvector $|v\rangle$ such that $U|v\rangle = e^{2\pi i \phi_v} |v\rangle$, this implies that:

$$U^{2^j} |v\rangle = U^{2^j - 1} U |v\rangle = U^{2^j - 1} e^{2\pi i \phi_v} |v\rangle = \cdots = e^{2\pi i 2^j \phi_v} |v\rangle .$$

Hence, applying all the t controlled-U^{2^j} operations with $0 \leq j \leq t-1$, and using the fact that $|0\rangle \otimes |v\rangle + |1\rangle \otimes e^{2\pi i\phi}|v\rangle = \left(|0\rangle + e^{2\pi i\phi}|1\rangle\right) \otimes |v\rangle$, we obtain:

$$\begin{aligned}|\psi_2\rangle &= \frac{1}{\sqrt{2^t}}\left(|0\rangle + e^{2\pi i 2^{t-1}\phi_v}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i 2^{t-2}\phi_v}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i 2^0 \phi_v}|1\rangle\right) \otimes |v\rangle \\ &= \frac{1}{2^{\frac{t}{2}}} \sum_{k=0}^{2^t-1} e^{2\pi i \phi_v k}|k\rangle \otimes |v\rangle\end{aligned}$$

where k denotes the integer representation of t-bit binary numbers [15]. This completes the first stage of the quantum phase estimation procedure.

The second stage of the phase estimation procedure involves applying the inverse quantum Fourier transform on the first register and then doing measurement [15]. To motivate this, let us point out that the expression for $|\psi_2\rangle$ is exactly the result of applying a quantum Fourier transform on the first register. To better see this, recall that the Fourier transform takes an n-qubit state $|x\rangle$ as input and produces an output as:

$$\text{QFT}|x\rangle = \frac{1}{2^{\frac{n}{2}}}\left(|0\rangle + e^{\frac{2\pi i}{2}x}|1\rangle\right) \otimes \left(|0\rangle + e^{\frac{2\pi i}{2^2}x}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{\frac{2\pi i}{2^{n-1}}x}|1\rangle\right) \otimes \left(|0\rangle + e^{\frac{2\pi i}{2^n}x}|1\rangle\right).$$

Hence, observe that replacing x by $2^t\phi_v$ in the expression above yields exactly the expression we derived for ψ_2. Thus, to recover $|x\rangle = |2^t\phi_v\rangle$, we apply an inverse Fourier transform on the first register, which then yields:

$$|\psi_3\rangle = \frac{1}{2^{\frac{t}{2}}} \sum_{k=0}^{2^t-1} e^{2\pi i\phi k}|k\rangle \otimes |v\rangle \xrightarrow{\text{QFT}_t^{-1}} \frac{1}{2^n} \sum_{x=0}^{2^t-1}\sum_{k=0}^{2^t-1} e^{\frac{2\pi i k}{2^t}(x - 2^t\phi_v)}|x\rangle \otimes |v\rangle$$

It is not hard to see that the above expression achieves peaks near $x = 2^t\phi_v$. Thus, for the case when $2^t\phi_v$ is an integer, measuring in the computational basis gives us an approximate phase in the first register, with high probability:

$$|\phi_4\rangle = |2^t\phi_v\rangle \otimes |v\rangle,$$

and in the case when it is not an integer, it can still be shown that it peaks near $x = 2^t\phi_v$ with probability better than $4/\pi^2 \approx 40\%$ [15].

C Qiskit Implementation

```python
import numpy as np
from numpy import pi
from scipy.linalg import expm, solve
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister, transpile
from qiskit.quantum_info import Statevector
from qiskit.circuit.library import RYGate, UnitaryGate
from qiskit.circuit import Measure
from qiskit_aer import Aer
import matplotlib.pyplot as plt

# We first define the problem data appropriately.
A = np.array([[1, 0.5],
              [0.5, 1 ]], dtype=float)
b_vec = np.array([1, 0], dtype=float)
evals = np.linalg.eigvalsh(A)
t  = pi
U  = expm(1j * A *  t)
U2 = expm(1j * A * 2*t)

# We also define a helper to wrap any 1-qubit matrix to a controlled unitary gate.
def ctrl_unitary(Umat, label='U'):
    return UnitaryGate(Umat, label=label).control()

```

```
# We proceed to build the circuit to simulate our example.
n_b, n_clk = 1, 2
breg  = QuantumRegister(n_b,   'b')
clk   = QuantumRegister(n_clk, 'clk')
anc   = QuantumRegister(1,     'anc')
creg  = ClassicalRegister(2,   'c')   # We note that here c[0]=ancilla and c[1]=b-qubit.
qc = QuantumCircuit(anc, clk, breg, creg)

# As our state is already prepared, we directly start with quantum phase estimation
    subroutine.
qc.h(clk)
qc.append(ctrl_unitary(U), [clk[0], breg[0]])
qc.append(ctrl_unitary(U2), [clk[1], breg[0]])
qc.swap(clk[0], clk[1])   # This step aligns MSB/LSB.

# Next, we perform the inverse quantum Fourier transform.
qc.h(clk[0])
qc.cp(pi/2, clk[0], clk[1])
qc.h(clk[1])

# Following this, we apply the controlled rotation on the ancilla
C = 0.5   # For the sake of convenience, we choose C = evals_min as the ratio does not
    change by equal scaling.
theta_min = 2*np.arcsin(C / evals[0])
theta_max = 2*np.arcsin(C / evals[1])

qc.append(RYGate(theta_min).control(2, ctrl_state='01'), [clk[0], clk[1], anc[0]])
qc.append(RYGate(theta_max).control(2, ctrl_state='11'), [clk[0], clk[1], anc[0]])

# We now measure the ancilla.
qc.measure(anc, creg[0])

# Following this, we apply inverse phase estimation on the c-register.
qc.swap(clk[0], clk[1])
qc.h(clk[0])
qc.cp(-pi/2, clk[0], clk[1])
qc.h(clk[1])
qc.append(ctrl_unitary(U2.conj().T), [clk[1], breg[0]])
qc.append(ctrl_unitary(U.conj().T), [clk[0], breg[0]])
qc.h(clk)

# We perform the final measurement, measuring the b-qubit.
qc.measure(breg, creg[1])

# Finally, we execute the example.
backend = Aer.get_backend('aer_simulator')
shots   = 8192
counts  = backend.run(transpile(qc, backend), shots=shots).result().get_counts()

# We filter and only keep the shots where ancilla collapsed to |1>
success = {bs: ct for bs, ct in counts.items() if bs[-1] == '1'}
soln_counts = {'0': 0, '1': 0}
for bs, ct in success.items():
    soln_counts[bs[-2]] += ct
```

References

[1] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009. doi: 10.1103/PhysRevLett.103.150502. URL https://link.aps.org/doi/10.1103/PhysRevLett.103.150502.

[2] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 4th edition, 2013.

[3] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2nd edition, 2003.

[4] Danial Dervovic, Mark Herbster, Peter Mountney, Simone Severini, Naïri Usher, and Leonard Wossnig. Quantum linear systems algorithms: a primer, 2018. URL https://arxiv.org/abs/1802.08227.

[5] B. P. Lanyon, J. D. Whitfield, G. G. Gillett, M. E. Goggin, M. P. Almeida, I. Kassal, J. D. Biamonte, M. Mohseni, B. J. Powell, M. Barbieri, A. Aspuru-Guzik, and A. G. White. Towards quantum chemistry on a quantum computer. *Nature Chemistry*, 2(2):106–111, 2010. doi: 10.1038/nchem.483. URL https://doi.org/10.1038/nchem.483.

[6] Ashley Montanaro. Quantum algorithms: an overview. *npj Quantum Information*, 2(1):15023, 2016. doi: 10.1038/npjqi.2015.23. URL https://doi.org/10.1038/npjqi.2015.23.

[7] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Phys. Rev. Lett.*, 113:130503, Sep 2014. doi: 10.1103/PhysRevLett.113.130503. URL https://link.aps.org/doi/10.1103/PhysRevLett.113.130503.

[8] Anika Zaman, Hector Jose Morrell, and Hiu Yung Wong. A step-by-step hhl algorithm walkthrough to enhance understanding of critical quantum computing concepts. *IEEE Access*, 11:77117–77131, 2023.

[9] Guang Ping He. Solving the encoding bottleneck: of the hhl algorithm, by the hhl algorithm, 2025. URL https://arxiv.org/abs/2502.13534.

[10] Jessie M. Henderson, John Kath, John K. Golden, Allon G. Percus, and Daniel O'Malley. Addressing quantum's "fine print"with efficient state preparation and information extraction for quantum algorithms and geologic fracture networks. *Scientific Reports*, 14(1):3592, 2024. doi: 10.1038/s41598-024-52759-0. URL https://doi.org/10.1038/s41598-024-52759-0.

[11] Raphael Seidel, Nikolay Tcholtchev, Sebastian Bock, and Manfred Hauswirth. *Uncomputation in the Qrisp High-Level Quantum Programming Framework*, page 150–165. Springer Nature Switzerland, 2023. ISBN 9783031381003. doi: 10.1007/978-3-031-38100-3_11. URL http://dx.doi.org/10.1007/978-3-031-38100-3_11.

[12] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC '19, page 193–204. ACM, June 2019. doi: 10.1145/3313276.3316366. URL http://dx.doi.org/10.1145/3313276.3316366.

[13] Pedro C. S. Costa, Dong An, Yuval R. Sanders, Yuan Su, Ryan Babbush, and Dominic W. Berry. Optimal scaling quantum linear systems solver via discrete adiabatic theorem, 2021. URL https://arxiv.org/abs/2111.08152.

[14] Romina Yalovetzky, Pierre Minssen, Dylan Herman, and Marco Pistoia. Solving linear systems on quantum hardware with hybrid hhl++. *Scientific Reports*, 14(1):20610, 2024. doi: 10.1038/s41598-024-69077-0. URL https://doi.org/10.1038/s41598-024-69077-0.

[15] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.