HHL Algorithm

Pradeep Kumar - CPSC 619 Project Report
Course Instructor : Dr. Peter Høyer

17th April 2024

1 Abstract

The HHL algorithm, proposed by Harrow, Hassidim, and Lloyd [1] in 2009, addresses the fundamental challenge of efficiently solving linear systems of equations on quantum computers. Unlike classical methods with polynomial time complexity, HHL leverages quantum principles to achieve exponential speedup in solving sparse linear systems. We elucidate the key components of the HHL algorithm, including quantum state preparation, phase estimation, and qubit operations, explaining how they collectively enable the quantum solution of linear systems. Furthermore, we calculate the complexity of the algorithm for bounded error scenarios.

2 Introduction

Traditionally, linear systems are solved using classical algorithms such as Gaussian elimination or iterative methods like Conjugate Gradient, which have a polynomial time complexity. However, as the size and complexity of linear systems grow, even the fastest classical algorithms can struggle to provide solutions in a reasonable time frame, particularly when dealing with very large-scale problems.

3 Background and Preliminaries

3.1 Relevant definitions

The HHL algorithm's primary goal is solving systems of linear equations, which can be described as finding $\vec{x}$ in the equation $A\vec{x} = \vec{b}$, where A is a known matrix and $\vec{b}$ is a known vector. Key concepts from linear algebra relevant to this include:

- **Condition Number:** This is a measure of a matrix's sensitivity to changes in its input. It is defined as the ratio of the largest to the smallest singular value. Matrices with a high condition number are said to be ill-conditioned, which can make numerical solutions difficult or unstable.
- **Sparsity of Matrices:** A matrix is considered sparse if most of its elements are zero. The efficiency of the HHL algorithm significantly improves when dealing with sparse matrices, as it exploits this sparsity.

3.2 Solving a linear system of equations

The HHL algorithm [1] solves a system of linear equations, written in a matrix form as $\mathbf{A}\vec{x} = \vec{b}$

$$\vec{x} = \mathbf{A}^{-1}\vec{b} = \mathbf{A}^{-1}\sum_{i=1}^{N}\beta_i\vec{u}_i \ = \sum_{i=1}^{N}\beta_i\mathbf{A}^{-1}\vec{u}_i = \sum_{i=1}^{N}\frac{\beta_i}{\lambda_i}\vec{u}_i$$

where $\vec{u_i}$ and λ_i are the eigenvectors and eigenvalues of the matrix $\mathbf{A}$ respectively.

4 Classical Algorithms for solving linear system of equations

4.1 Gauss elimination

In Gaussian elimination we perform three row operations

- Swapping two rows
- Multiplying a row with a non-zero number
- Adding a multiple of one row to another row

We start with the top row using the above three operations make the element in the first column equal to one. If the element is zero then use swap operation to get non-zero element in the first-row-first-column. Then if there is a non-zero entry below one, use row operations to make it zero. Use the similar method for the second until we are in the row-echelon form. Then transform the matrix to the diagonal form using row operations.

4.2 Conjugate gradient descent

For the linear system $\mathbf{A}\vec{x} = \vec{b}$, if $\mathbf{A}$ is symmetric, semi-positive matrix and well conditioned then Conjugate gradient descent algorithm can solve the linear system much faster than Gaussian elimination.

5 The HHL Algorithm

5.1 Overview of the HHL Algorithm

The Harrow-Hassidim-Lloyd (HHL) algorithm addresses the problem of solving linear equations described by $\mathbf{A}\vec{x} = \vec{b}$, where A is an $n \times n$ Hermitian matrix. This quantum algorithm provides an exponential speedup compared to the best known classical algorithms under certain conditions. The exponential speedup is contingent on the sparsity of $\mathbf{A}$ and a favorable condition number κ.

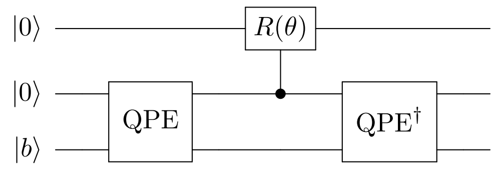
5.2 Input

The input to the HHL algorithm includes an $n \times n$ Hermitian matrix $\mathbf{A}$ and a vector $\vec{b}$, provided in classical form. The vector $\vec{b}$ is encoded as a quantum state, and the matrix $\mathbf{A}$ is represented as a Hamiltonian operator for processing in the quantum algorithm.

5.3 Output

The output of the HHL algorithm is a quantum state $|x\rangle$ that represents the solution to the linear equation $A\vec{x} = \vec{b}$. It can used to estimate $\langle x|\mathbf{M}|x\rangle$, where $\mathbf{M}$ is some operator.

5.4 Procedure



5.4.1 Quantum State Preparation

The initial step in the HHL algorithm involves encoding the vector $\vec{b}$ into a quantum state $|\vec{b}\rangle$. The HHL paper [1] refers to the procedure in Ref. [2] to prepare $|b\rangle$. Alternatively we can use an oracle such that

$$U_b|0\rangle^{\otimes n} = |b\rangle = \sum_{i=1}^{n} b_i|i\rangle$$

where $\{b_i\}$ are the normalized components of $\vec{b}$, and $|i\rangle$ are the standard basis states. The normalization condition requires $\sum_{i=1}^{n} |b_i|^2 = 1$. Therefore :

$$|\psi_{initial}\rangle = |0\rangle^{\otimes n} \otimes |0\rangle^{\otimes m} \otimes |0\rangle$$

$$|\psi_{initial}\rangle \longrightarrow |0\rangle^{\otimes n} \otimes |b\rangle \otimes |0\rangle$$

Where the first $|0\rangle^{\otimes n}$ register is to encode the eigenvalues as shown below.

5.4.2 Quantum Phase Estimation

Quantum Phase Estimation (QPE) is used to estimate the eigenvalues λ_i of the matrix A. First, the unitary operator $e^{iA\tau}$ is prepared from the given Hermitian matrix $\mathbf{A}$, where τ is a suitable chosen time parameter. $\mathbf{A}$ is a s-sparse matrix whose coefficients are efficiently commutable. The system is initialized in the state $|0\rangle^{\otimes n} \otimes |b\rangle \otimes |0\rangle$ and the auxiliary register($|0\rangle^{\otimes n}$) is prepared in an equal superposition of computational basis states, typically using Hadamard gates. The QPE involves evolving this state under the unitary $e^{iA\tau}$ controlled on the auxiliary qubit states. Mathematically, the evolution can be captured as:

$$U = \sum_{j=0}^{T-1} |j\rangle\langle j| \otimes e^{iA\tau j/T}$$

where T is a power of 2, usually $T = 2^t$. After applying U on the prepared state we apply inverse Quantum Fourier Transform, which result in the superpositions of states which are eigenstates of the matrix $\mathbf{A}$.

The idea is that upon application of the operator $e^{iA\tau}$ on the eigenstate $|u_i\rangle$ of $\mathbf{A}$, we will get the state $e^{i\lambda_i\tau}|u_i\rangle$ and using the QPE subroutine we can encode the eigenvalue in the $|0\rangle^{\otimes n}$ register. We can write the state $|b\rangle$ as $\sum_j \beta_j |u_j\rangle$. Post application of the QPE, the state of the system becomes:

$$|\psi_{QPE}\rangle = \sum_j \beta_j |\lambda_j\rangle |u_j\rangle\, |0\rangle$$

where $|u_j\rangle$ are the eigenvectors of A, and the eigenvalues λ_j are encoded in the states of the auxiliary register. Detailed algorithm and circuit for Quantum Phase Estimation is given in the Appendix B.

5.4.3 Controlled Rotations

Following the QPE, controlled rotations are performed on an ancillia qubit based on the eigenvalues encoded in the state [3]. Each rotation angle is inversely proportional to the encoded eigenvalue $R_y(2\arcsin(C/\lambda_j))$ where,

$$R_y(2\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

and C is a normalization constant ensuring that the amplitudes remain valid probabilities. This step effectively encodes the solution into the amplitudes of the quantum state corresponding to the inverse of A. Therefore the state $|\psi_{QPE}\rangle$ transforms as:

$$\mathcal{R}_y\,|\psi_{QPE}\rangle = \sum_j \beta_j |\lambda_j\rangle |u_j\rangle \left(\frac{C}{\lambda_j}\,|1\rangle + \sqrt{1 - \frac{C^2}{\lambda_j^2}}\,|0\rangle \right)$$

The sub-routine and circuit for the controlled rotation is given in Appendix C

5.4.4 Inverse QPE

To extract the solution, the quantum phase estimation procedure must be reversed to uncompute the auxiliary register used in the phase estimation. This process involves applying the inverse of the initial QPE circuit, returning the auxiliary register to its original state and leaving the system in a state to:

$$\sum_j \beta_j |0\rangle^{\otimes n} |u_j\rangle \left(\frac{C}{\lambda_j}\,|1\rangle + \sqrt{1 - \frac{C^2}{\lambda_j^2}}\,|0\rangle \right)$$

5.4.5 Measurement and Post-Processing

The final state $|\vec{x}\rangle$ encodes the solution to the system of linear equations. This state is measured to collapse it to one of the basis states. Conditioned on seeing state $|1\rangle$, we get

$$|0\rangle^{\otimes n} \otimes |x\rangle \otimes |1\rangle = |0\rangle^{\otimes n} \otimes \left(\sum_j \beta_j |u_j\rangle \frac{C}{\lambda_j} \right) \otimes |1\rangle$$

$$|x\rangle = \sum_j \frac{C}{\lambda_j} \beta_j |u_j\rangle$$

The state $|x\rangle$ is proportional to the $\vec{x}$ which is the solution of the linear equation $\mathbf{A}\vec{x} = \vec{b}$ and which can be used to estimate $\langle x| \mathbf{M} |x\rangle$ for some operator $\mathbf{M}$.

6 Error Analysis and Complexity

The overall complexity of HHL is $O(\log(N)s^2\kappa^2/\epsilon)$, where ϵ is the desired precision.

6.1 Computational Complexity of HHL

The computational complexity of the HHL algorithm depends primarily on the sparsity and condition number of the matrix A, as well as the desired precision ϵ of the solution. Key elements influencing the complexity include:

- **Hamiltonian Simulation:** The time complexity for simulating the evolution under A (as a Hamiltonian) is $O(\log(N)s^2\tau)$ [4]
- **Quantum Phase Estimation (QPE):** The precision of eigenvalue estimation is $\mathcal{O}(\frac{1}{\tau})$, therefore the relative error for $\frac{1}{\lambda}$ is $\frac{1}{\lambda\tau}$. If we take this error to be ϵ and also $\lambda \geq 1/\kappa$ then

$$\epsilon = \frac{1}{\lambda\tau} \text{ hence } \epsilon \leq \frac{\kappa}{\tau} \text{ which gives } \tau \leq \frac{\kappa}{\epsilon}$$

- **Controlled Rotations and Amplitude Amplification:** Amplitude amplification [5] might be necessary to increase the probability of measuring the desired state, usually adding a multiplicative factor of $O(\kappa)$ to the complexity (this comes from $\kappa \approx \frac{1}{\sqrt{p}}$, where p is the probability of getting the state $|1\rangle$ upon measuring the ancillia qubit.)

Therefore, the overall complexity of HHL is $O(\log(N)\kappa^2 s^2/\epsilon)$. This makes HHL particularly advantageous for matrices that are sparse and well-conditioned, with large dimensions N where classical algorithms would be infeasible.

6.2 Comparison with Classical Algorithms

- Classical algorithms for solving linear systems, such as Gaussian elimination, have a polynomial time complexity, typically $O(N^3)$
- Methods like Conjugate Gradient have complexities that depend on the sparsity and condition number, generally requiring $O(Ns\kappa)$ operations.

6.3 Conditions for Significant Quantum Advantage

The HHL algorithm provides a significant quantum advantage under specific conditions:

- **Sparsity:** The matrix A must be sparse. This sparsity directly reduces the complexity of Hamiltonian simulation, which is central to the HHL algorithm.
- **Well-conditioned Matrix:** The condition number κ plays a critical role. The smaller κ is, the fewer rotations are required, and the less amplification is needed, directly affecting both the complexity and the probability of successful state preparation and measurement.

Appendices

A Quantum Fourier Transform Algorithm

Algorithm 1 Quantum Fourier Transform (QFT)

```
1:  procedure QFT($|\psi\rangle$)                    ▷ $|\psi\rangle$ is the input quantum state
2:      $n \leftarrow$ number of qubits in $|\psi\rangle$
3:      for $k = 0$ to $n - 1$ do
4:          Apply Hadamard gate to qubit $k$
5:          for $j = k + 1$ to $n$ do
6:              Apply controlled-$R_j$ gate to qubits $k$ and $j$
7:          end for
8:      end for
9:      for $k = 0$ to $\lfloor n/2 \rfloor$ do
10:         Swap qubit $k$ with qubit $n - k$
11:     end for
12: end procedure
```

QFT transforms a basis state as:

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

We can implement single qubit and two qubit gates, which is much faster than the classical algorithm for Fast Fourier transform. Following equations show how we can build the circuit for Quantum Fourier transform. It is borrowed from the reference [6]

To construct this implementation, we express j as an n-bit binary number:

$$j = j_{n-1}j_{n-2}\ldots j_1j_0 = j_{n-1}2^{n-1} + j_{n-2}2^{n-2} + \cdots + j_1 2 + j_0.$$

Then, $\frac{j}{N}$ can be represented using a binary point, which is like a decimal point, but in base 2:

$$\frac{j}{N} = \frac{j_{n-1}2^{n-1} + j_{n-2}2^{n-2} + \cdots + j_1 2 + j_0}{2^n} = 0.j_{n-1}j_{n-2}\ldots j_1j_0.$$

Similarly, we can express k as an n-bit binary number:

$$k = k_{n-1}k_{n-2}\ldots k_1k_0 = k_{n-1}2^{n-1} + k_{n-2}2^{n-2} + \cdots + k_1 2 + k_0.$$

Using these, the exponential in QFT transformation equation above is $e^{2\pi ijk/N} = e^{2\pi i(j/N)k}$:

$$e^{2\pi i(0.j_{n-1}j_{n-2}\ldots j_1j_0)(k_{n-1}2^{n-1}+k_{n-2}2^{n-2}+\cdots+k_1 2+k_0)}.$$

Dropping all the bits left of the binary point, we have:

$$e^{2\pi ijk/N} = e^{2\pi i(0.j_0)k_{n-1}} e^{2\pi i(0.j_1j_0)k_{n-2}} \ldots \times e^{2\pi i(0.j_{n-2}\ldots j_1j_0)k_1} e^{2\pi i(0.j_{n-1}j_{n-2}\ldots j_1j_0)k_0}.$$

Plugging this into the QFT transformation equation, we get:

$$\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N}|k\rangle = \sqrt{\frac{1}{N}} \sum_{k=0}^{N-1} \left(e^{2\pi i(0.j_0)k_{n-1}} e^{2\pi i(0.j_1j_0)k_{n-2}} \ldots e^{2\pi i(0.j_{n-2}\ldots j_1j_0)k_1} e^{2\pi i(0.j_{n-1}j_{n-2}\ldots j_1j_0)k_0} \right) |k\rangle.$$

Since we are summing over all n-bit binary numbers k, each bit $k_{n-1}, k_{n-2}, \ldots, k_0$ sums through 0 and 1, so this becomes:

$$\sqrt{\frac{1}{N}} \sum_{k_{n-1}=0}^{1} \cdots \sum_{k_0=0}^{1} \left(e^{2\pi i(0.j_0)k_{n-1}} \ldots e^{2\pi i(0.j_{n-1}j_{n-2}\ldots j_1j_0)k_0} \right) |k_{n-1}\ldots k_0\rangle.$$

Since $|k_{n-1}\dots k_0\rangle$ is shorthand for $|k_{n-1}\rangle\dots|k_0\rangle$, we can move the terms to get:

$$\sqrt{\frac{1}{N}}\sum_{k_{n-1}=0}^{1} e^{2\pi i(0.j_0)k_{n-1}}|k_{n-1}\rangle\dots\sum_{k_0=0}^{1} e^{2\pi i(0.j_{n-1}j_{n-2}\dots j_1j_0)k_0}|k_0\rangle.$$

Moving the summations, we evaluate them as:

$$\sqrt{\frac{1}{N}}\left(|0\rangle+e^{2\pi i(0.j_0)}|1\rangle\right)\dots\left(|0\rangle+e^{2\pi i(0.j_{n-1}j_{n-2}\dots j_1j_0)}|1\rangle\right).$$

Finally, since $\sqrt{N}=\sqrt{2^n}=(\sqrt{2})^n$, we get the product state:

$$\frac{1}{\sqrt{2}}\left(|0\rangle+e^{2\pi i(0.j_0)}|1\rangle\right)\dots\frac{1}{\sqrt{2}}\left(|0\rangle+e^{2\pi i(0.j_{n-1}j_{n-2}\dots j_1j_0)}|1\rangle\right).$$

Now consider applying Hadamard gate to $|j_{n-1}\rangle$

$$H|j_{n-1}\rangle=\frac{1}{\sqrt{2}}\left(|0\rangle+(-1)^{j_{n-1}}|1\rangle\right)=\frac{1}{\sqrt{2}}\left(|0\rangle+(e^{i\pi j_{n-1}})|1\rangle\right)$$
$$=\frac{1}{\sqrt{2}}\left(|0\rangle+e^{\frac{2\pi i j_{n-1}}{2}}|1\rangle\right)=\frac{1}{\sqrt{2}}\left(|0\rangle+e^{2\pi i(0.j_{n-1})}|1\rangle\right).$$

Next, consider a single-qubit gate that rotates about the z-axis of the Bloch sphere by $\frac{2\pi}{2^r}$ radians, which we call R_r. It acts on basis states by

$$R_r|0\rangle=|0\rangle,\quad R_r|1\rangle=e^{\frac{2\pi i}{2^r}}|1\rangle,$$

and its matrix representation is

$$R_r=\begin{pmatrix}1 & 0\\ 0 & e^{\frac{2\pi i}{2^r}}\end{pmatrix}.$$

After the previous Hadamard matrix, we apply R_2 to qubit $n-1$, controlled by qubit $n-2$. That is, for the state of qubit $n-1$, the amplitude of $|1\rangle$ is multiplied by $e^{\frac{2\pi i}{2^2}}$ if $j_{n-2}=1$, and nothing happens otherwise. That is, the state of the $(n-1)$-th qubit goes from

$$\frac{1}{\sqrt{2}}\left(|0\rangle+e^{2\pi i(0.j_{n-1})}|1\rangle\right)\to\frac{1}{\sqrt{2}}\left(|0\rangle+e^{2\pi i(0.j_{n-1})+\frac{2\pi i}{2^2}j_{n-2}}|1\rangle\right)$$
$$=\frac{1}{\sqrt{2}}\left(|0\rangle+e^{2\pi i(0.j_{n-1}j_{n-2})}|1\rangle\right).$$

Similarly, we can apply R_3 to $n-1$, controlled by qubit $n-3$. Then, the state of qubit $n-1$ would be

$$\frac{1}{\sqrt{2}}\left(|0\rangle+e^{2\pi i(0.j_{n-1}j_{n-2}j_{n-3}}|1\rangle\right).$$

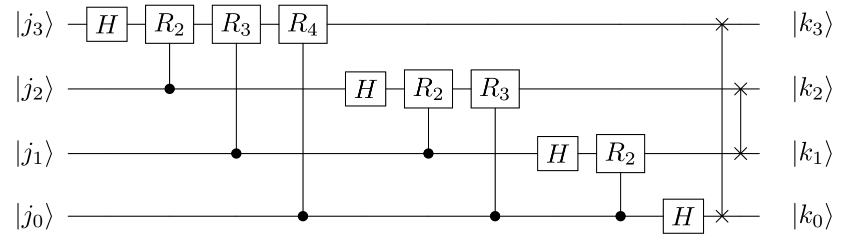
Continuing this through R_n, controlled by qubit 0, the state of qubit $n-1$ is

$$\frac{1}{\sqrt{2}}\left(|0\rangle+e^{2\pi i(0.j_{n-1},j_{n-2}\dots,j_0}|1\rangle\right).$$

Therefore the QFT transformation can be written as :

$$\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}e^{2\pi ijk/N}|k\rangle=H\,|j_0\rangle\otimes HR_2\,|j_1\rangle\otimes\dots\otimes H\sum_{i=2}^{n}R_i\,|j_{n-1}\rangle$$

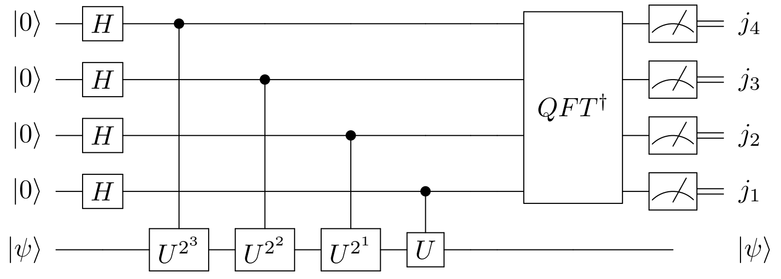
Where R_i is controlled by the state $|j_i\rangle$



B Quantum Phase Estimation Algorithm

Algorithm 2 Quantum Phase Estimation (QPE)

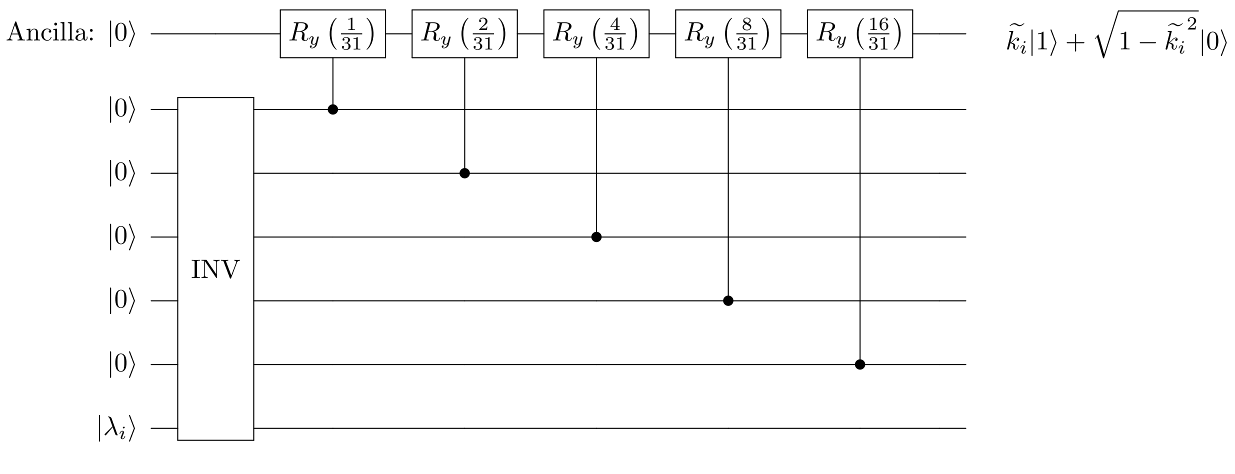
```
1: procedure QuantumPhaseEstimation($U, |\psi\rangle, t$)  ▷ $U$: unitary operator, $|\psi\rangle$: eigenstate, $t$: number of qubits for precision
2:     Prepare the register of $t$ auxiliary qubits in state $|0\rangle^{\otimes t}$.
3:     Apply a Hadamard gate $H$ to each auxiliary qubit to create a superposition.
4:     for $k = 1$ to $t$ do
5:         Apply $U^{2^{k-1}}$ to $|\psi\rangle$ controlled by the $k$-th auxiliary qubit.
6:     end for
7:     Apply the inverse Quantum Fourier Transform ($\text{QFT}^{-1}$) on the auxiliary qubits.
8:     Measure the auxiliary qubits in the computational basis.
9:     The observed outcome $\tilde{\phi}$ is an approximation of the phase $\phi$.
10: end procedure
```



C Controlled Rotation Algorithm

Algorithm 3 Quantum Algorithm for Encoding Inverse Proportional Amplitudes

```
1: Initialize all qubits to $|0\rangle$
2: Prepare the qubit-register $|\lambda_i\rangle$ with the value $\lambda_i$
3: Apply INV operation to transform $|\lambda_i\rangle$ to $|C/\lambda_i\rangle$
4: for $k = 0$ to $4$ do
5:     Apply controlled $R_y\left(\frac{2^k}{2^5-1}\right)$ to the first qubit controlled by the $(k+1)$-th qubit
6: end for
7: The ancillia qubit is now in the state $\tilde{k}_i|1\rangle + \sqrt{1-\tilde{k}_i^2}|0\rangle$
8: end
```



Ancilla: $|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|\lambda_i\rangle$
INV
$R_y\left(\frac{1}{31}\right)$
$R_y\left(\frac{2}{31}\right)$
$R_y\left(\frac{4}{31}\right)$
$R_y\left(\frac{8}{31}\right)$
$R_y\left(\frac{16}{31}\right)$
$\widetilde{k}_i|1\rangle + \sqrt{1-\widetilde{k}_i^{\,2}}|0\rangle$

References

[1] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Phys. Rev. Lett.*, vol. 103, no. 15, Oct. 2009, ISSN: 1079-7114. DOI: 10.1103/physrevlett.103.150502. [Online]. Available: http://dx.doi.org/10.1103/PhysRevLett.103.150502.

[2] L. Grover and T. Rudolph, *Creating superpositions that correspond to efficiently integrable probability distributions*, 2002. arXiv: quant-ph/0208112 [quant-ph].

[3] Y. Cao, A. Papageorgiou, I. Petras, J. Traub, and S. Kais, "Quantum algorithm and circuit design solving the poisson equation," *New Journal of Physics*, vol. 15, no. 1, p. 013 021, Jan. 2013, ISSN: 1367-2630. DOI: 10.1088/1367-2630/15/1/013021. [Online]. Available: http://dx.doi.org/10.1088/1367-2630/15/1/013021.

[4] D. W. Berry, G. Ahokas, R. Cleve, and B. C. Sanders, "Efficient quantum algorithms for simulating sparse Hamiltonians," *Commun. Math. Phys.*, vol. 270, no. 2, pp. 359–371, 2007, ISSN: 0010-3616. DOI: 10.1007/s00220-006-0150-x. [Online]. Available: http://dx.doi.org/10.1007/s00220-006-0150-x.

[5] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, *Quantum amplitude amplification and estimation*, 2002. DOI: 10.1090/conm/305/05215. [Online]. Available: http://dx.doi.org/10.1090/conm/305/05215.

[6] T. G. Wong, *Introduction to Classical and Quantum Computing.* 2023. [Online]. Available: https://www.thomaswong.net/introduction-to-classical-and-quantum-computing-1e4p.pdf.