Report on implementing HHL algorithm

Piotr Lenczewski

February 6, 2025

1 Introduction

In this report, I will explain the implementation and results of the quantum circuit based on a numerical example from paper: Quantum Circuit Design for Solving Linear Systems of Equations. It is a 7-qubit circuit for solving a linear system Ax = b with A of dimension 4 × 4. For the numerical example:

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

$$b = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

The eigenvalues of matrix A are $\lambda_i = 2^{i-1}$ and the corresponding eigenvector $|u_i\rangle = \frac{1}{2}\sum_{j=1}^{4}(-1)^{\delta_{ij}}|j\rangle_C$ where $|j\rangle_C$ represents the state of register C which encode the number j in binary form, δ_{ij} is the Kronecker delta, and the index i runs from 1 to 4. The searched for solution of this problem is:

$$x = \frac{1}{32}\begin{bmatrix} -1 \\ 7 \\ 11 \\ 13 \end{bmatrix}$$

The implementation of the algorithm differs from circuit outlined in the paper. The details on how it differs and why the changes were made are described in the subsection Differences with original paper.

2 Implementation

The circuit was implemented in Python using the qiskit library. It uses 7 qbits distributed in 3 registers:

- Register B (2 qbits), which stores the state that represents vector b and will store the state representing x at the end of the algorithm,
- Register C (4 qbits), which is used to store the eigenvalues after Quantum Phase Estimation,
- Ancilla (1 qbit), on which a controlled rotation dependent on eigenvalues in register C will be applied.

2.1 State preparation

Vector b is encoded in the B register as $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$. To achieve that, Hadamard gates are used on qbits of the B register.

2.2 Quantum Phase Estimation

In this part, QPE is used as an eigenvalue estimation algorithm. The algorithm is made up of 3 parts: C Register preparation, controlled rotation, and inverse quantum Fourier transform.

2.2.1 C register preparation

Hadamard gates are applied to create a superposition of qbits in the C register. The resulting state is $|b\rangle \frac{1}{2^{n/2}}(|0\rangle + |1\rangle)^{\otimes n}|0\rangle = |b\rangle \frac{1}{4}(|0\rangle + |1\rangle)^{\otimes 4}|0\rangle_a$.

2.2.2 Controlled rotation

Here, controlled gates $e^{2\pi i\phi 2^j}$ are applied to the B register, where j-th qbit in C register is the control qbit and ϕ is the phase. This operation is equivalent to multiplying $e^{2\pi i\phi 2^j}$ in front of $|1\rangle$ in j-th qbit in C register. After gates are applied the state is $|b\rangle \otimes \frac{1}{4}\big(|0\rangle + e^{16\pi i\phi}|1\rangle\big) \otimes \big(|0\rangle + e^{8\pi i\phi}|1\rangle\big) \otimes \big(|0\rangle + e^{4\pi i\phi}|1\rangle\big) \otimes \big(|0\rangle + e^{2\pi i\phi}|1\rangle\big) \otimes |0\rangle_a = |b\rangle \frac{1}{4}\sum_{k=0}^{7} e^{2\pi i\phi k}|k\rangle|0\rangle_a$.

2.2.3 Inverse quantum Fourier transform

The phase value is now encoded in the state, that is similar to the result of quantum Fourier transform: $y_l = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}\exp\left(\frac{2\pi ijl}{N}\right)x_j, \quad \text{for } l = 0, \ldots, N-1$

Thus $\sqrt{N} = 4 \rightarrow N = 16$, y_l is the state representing the phase and $\phi = \lambda/N$ where λ is the eigenvalue, j is equivalent of k after controlled rotation and x_j is equivalent of the state of register C after controlled rotation $|k\rangle$.

To retrieve the phase we can use the inverse of QFT. The result state is then $|b\rangle|\phi\rangle|0\rangle$. However, we currently have the phase not the eigenvalue. We can rewrite state $|b\rangle$ as superposition of eigenvectors u_j: $\sum_{j=0}^{2^{n_b}-1} b_j|u_j\rangle$ where n_b is the number of qbits in B register. We also know that $\phi = \frac{\lambda}{N}$ thus the state after IQFT can be written as $\sum_{j=0}^{2^{n_b}-1} b_j|u_j\rangle\left|\frac{\lambda_j}{N}\right\rangle|0\rangle_a$ or in case of numerical example $\sum_{j=0}^{3} b_j|u_j\rangle\left|\frac{\lambda_j}{16}\right\rangle|0\rangle_a$.

Now we can rewrite the controlled gates $e^{2\pi i\phi 2^j} \rightarrow e^{\frac{2\pi i\lambda 2^j}{16}}$. Because $Au_j = \lambda_j u_j$ we can write the final gate as $e^{\frac{2\pi iA2^j}{16}}$.

2.2.4 Implementation of $e^{\frac{2\pi iA2^j}{16}}$ gate

The implementation was created based on the schema from the original paper. The schema assumes MSB ordering of bits so in the code the bits are switched to have the LSB order. The gates used were accessible in qiskit in the same form, except the R_x and R_{zz} gates. The R_x gate in the paper is the inverse of the one available in qiskit library. R_{zz} had to be implemented manually using the X and U gates to be of shape:

$$R_{ZZ}(\theta) = \begin{pmatrix} \exp(i\theta) & 0 \\ 0 & \exp(i\theta) \end{pmatrix}$$

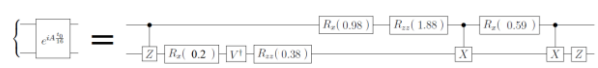

Figure 1: Schema of gate implementation from original paper

2.3 Ancilla bit rotation

In this part we want to encode the reciprocal of the eigenvalue in the $|1\rangle$ state of ancilla qbit using a transformation $|0\rangle_a \rightarrow \sqrt{1 - \frac{1}{\lambda_j^2}}|0\rangle_a + \frac{1}{\lambda_j}|1\rangle_a$. Only results where ancilla is 1 will be used and the rest will be discarded. We can achieve that transformation using R_y gates controlled by the C register qbits.

$$R_Y(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

This gate has the following effect on the $|0\rangle$ state ancilla qbit:

$$R_Y(\theta)|0\rangle_a = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \cos\left(\frac{\theta}{2}\right)|0\rangle_a + \sin\left(\frac{\theta}{2}\right)|1\rangle_a$$

Thus to match the desired transformation we want $\theta = 2\arcsin\left(\frac{1}{\lambda_j}\right)$. In reality we need the rotation to encode the value $\frac{m}{\lambda_j}$ which is proportional to $\frac{1}{\lambda_j}$, m being any multiplayer. This allows us to approximate $\arcsin\left(\frac{m}{\lambda_j}\right) \approx \frac{m}{\lambda_j}$ using Small-Angle Approximation. In the original paper they approximate it in the following way: $\theta = \frac{(2^{3-i}\pi)}{2^{r-1}}$ where i is the index of qbit in the C register (effectively inverting the eigenvalues) and r is an integer number. The bigger the r, the arcsin is approximated with higher precision, thus resulting in better results. However, this has a drawback, as the bigger r the smaller probability of ancilla qbit being 1. The resulting state after ancilla bit rotation, measuring the ancilla and discarding the states where ancilla is 0 is $\frac{1}{\sqrt{\sum_{j=0}^{2^{n_b}-1}\left|\frac{b_j}{\frac{\lambda_j}{16}}\right|^2}}\sum_{j=0}^{2^{n_b}-1} b_j|u_j\rangle|\frac{\lambda_j}{16}\rangle\frac{m}{\lambda_j}|1\rangle_a$.

2.4 Uncomputation

In uncomputation inverse of QPE is applied. The resulting state after uncomputation is $\frac{1}{\sqrt{\sum_{j=0}^{2^{n_b}-1}\left|\frac{b_j m}{\frac{\lambda_j}{16}}\right|^2}}\sum_{j=0}^{2^{n_b}-1}\frac{b_j m}{\lambda_j}|u_j\rangle|0\rangle^{\otimes n}|1\rangle_a$. Next the following properties are used:

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

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

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

The final state is $\frac{1}{\sqrt{\sum_{j=0}^{2^{n_b}-1}\left|\frac{b_j}{\lambda_j}\right|^2}}|\tilde{x}\rangle_b|0\rangle_c^{\otimes n}|1\rangle_a$, where $\tilde{x} = 16x$, due to the division by 16 in the QPE controlled rotation. For the numerical example, the final state is $\frac{1}{\sqrt{85}}|\tilde{x}\rangle_b|0\rangle_c^{\otimes n}|1\rangle_a$

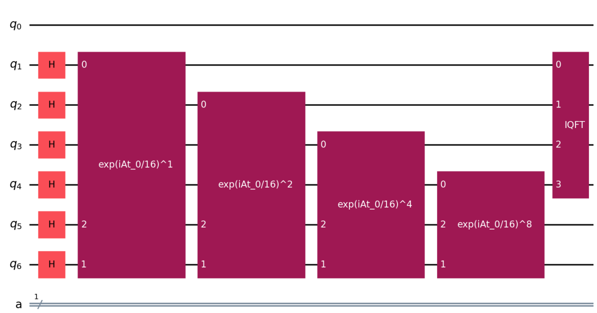

Figure 2: Final circuit 1/3

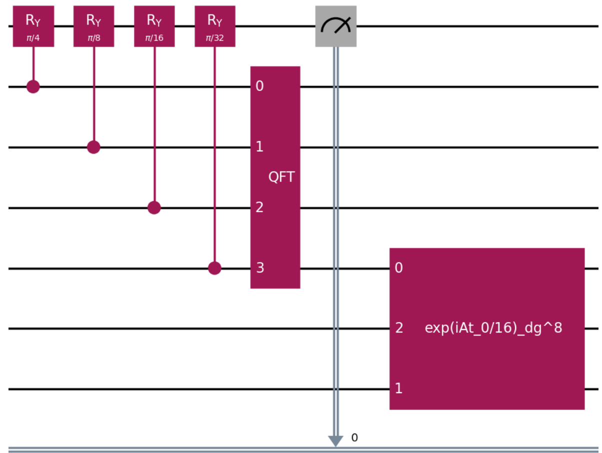

Figure 3: Final circuit 2/3

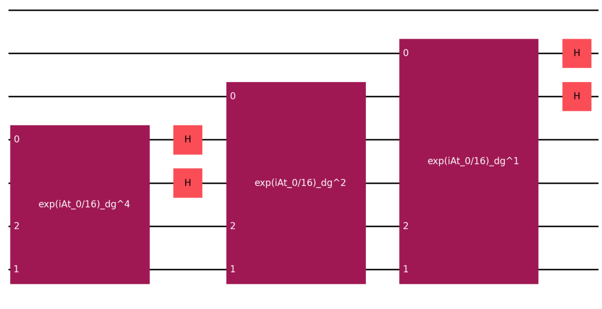

Figure 4: Final circuit 3/3

2.5 Differences with original paper

In the original paper, instead of $e^{\frac{2\pi i A 2^j}{16}}$, a $e^{\frac{-2\pi i A 2^j}{16}}$ gate is used. Thus, the result of the QPE is $1 - \frac{\lambda_j}{16}$ instead of $\frac{\lambda_j}{16}$. Also, a swap gate is used between the second and fourth qbit in the C register to find the reciprocals. After QPE and swap, the following states are produced for the eigenvalues:

- $\lambda_j = 8 \rightarrow 0010$
- $\lambda_j = 4 \rightarrow 0110$
- $\lambda_j = 2 \rightarrow 1110$
- $\lambda_j = 1 \rightarrow 1111$

Those seem to not correspond with the proposed ancilla bit rotation. I have tried to implement it in many different ways without seeing the correct results. I also do not see a theoretical way of explaining why such approach would result in a correct vector x. Thus, I excluded those elements from the implementation.

3 Results

The simulations of the circuit have been run on qiskit's AerSimulator. For r = 6 the state of the system after measuring ancilla as 1 is:

```
state of 0000001: (-0.045401064151482866+6.42812875480612e-16j)
state of 0100001: (0.3775885361238895-1.7660811885366237e-15j)
state of 1000001: (0.5962128017466629-2.296357449690517e-15j)
state of 1100001: (0.706655131056545-2.5379903229791127e-15j)
```

To find the exact vector x, we can simply multiply the real values by $\sqrt{85}$ and divide by 16. Then we have:

- 00: $-0.02616 \approx \frac{-1}{32}$
- 01: $0.21757 \approx \frac{7}{32}$
- 10: $0.34355 \approx \frac{11}{32}$
- 11: $0.40719 \approx \frac{13}{32}$

With smaller r the values are further away from the correct result and with bigger r the opposite.

Here is how the parameter r affects the chances of measuring ancilla as 1 (for single run 8192 shots each):

- r = 1: ['1': 2094, '0': 6098]
- r = 3: ['1': 3361, '0': 4831]
- r = 5: ['1': 1437, '0': 6755]
- r = 7: ['1': 97, '0': 8095]
- r = 10: ['1': 3, '0': 8189]

The results seem to match the results observed in the original paper and the expectations from the circuit implementation.