PHYS440 - Project: HHL Algorithm

John Hurst

June 2024

Contents

I Introduction

The Harrow-Hassidim-Lloyd (HHL) algorithm for solving linear systems[4] is the first of the quantum linear system algorithms (QLSAs), and one of the first algorithms to promise a quantum speedup for a variety of real-world problems. The walk-through in [12] shows the mathematics for each step and provides background on the relevant quantum computing concepts. The paper gives a numerical example to illustrate the steps, and provides MATLAB and Qiskit code for the example.

For my PHYS440 project I studied the walkthrough to understand the HHL algorithm in some detail, and supplemented this by reading several other introductory sources such as [2] and [5].

I ported the MATLAB sample code to two different Mathematica implementations. The first implementation is a direct port of the matrix formulation of the circuit from MATLAB. The second implementation uses the Wolfram Quantum Framework to implement the example using the QuantumCircuitOperator feature.

The paper's example uses a 2x2 matrix, and only two clock qubits, which does not sufficiently illustrate some features of the algorithm. I created my own example with a 4x4 matrix and three clock qubits, to study these features in more detail.

I verified my example in Mathematica, and in a Qiskit program run on the IBM Quantum simulator. I then created a more general Qiskit program to explore the behaviour and limits of the algorithm with larger matrices and more clock qubits.

I followed up with some more reading on the limits and applicability of HHL, starting from Aaronson's well-known paper [1].

The remainder of this report is in several parts.

- ✿ Part II discusses implementation details of the more general 4x4 numerical example, in particular the handling of the ancilla rotation.
- ✿ Part III describes some experiments with larger systems and more clock qubits.
- ✿ Part IV reviews literature on developments of the HHL algorithm with regard to applications to real word problems.

II Implementation Notes

This report will not repeat all the details from [12]. I will focus instead on what I found by generalising the example to a larger matrix and number of clock qubits.

II.1 Example 4x4 problem

The HHL algorithm is designed to solve the linear system

$$Ax = b \tag{1}$$

for a Hermitian matrix A.

For the sample problem, I constructed a 4x4 Hermitian matrix A with "friendly" eigenvalues in ratio 1:2:3:4, so that they could be represented exactly using 3 clock qubits.

I defined the matrix like this:

$$A = H_4 \begin{pmatrix} \frac{1}{3} & 0 & 0 & 0 \\ 0 & \frac{2}{3} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{4}{3} \end{pmatrix} H_4^\dagger = \begin{pmatrix} \frac{5}{6} & -\frac{1}{3} & 0 & -\frac{1}{6} \\ -\frac{1}{3} & \frac{5}{6} & -\frac{1}{6} & 0 \\ 0 & -\frac{1}{6} & \frac{5}{6} & -\frac{1}{3} \\ -\frac{1}{6} & 0 & -\frac{1}{3} & \frac{5}{6} \end{pmatrix} \tag{2}$$

where H_4 is a 4x4 Hadamard matrix. (There are different possible 4x4 Hadamard matrices, and a common choice is $H_4 = H_2 \otimes H_2$, but I used Mathematica's `HadamardMatrix[4]`, which is different.)

For b I used

$$b = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 2 \end{pmatrix} \tag{3}$$

The solution is given by

$$A^{-1}b = \frac{1}{4} \begin{pmatrix} 19 \\ 23 \\ 29 \\ 25 \end{pmatrix} \tag{4}$$

The HHL algorithm uses amplitude encoding for representing the input vector b and the result x.

We prepare the input using a scaled version of b:

$$\tilde{b} = \frac{b}{\|b\|}$$

When we run the circuit, we will observe outcome frequencies that correspond to the squared amplitudes of a state vector that is the amplitude encoding of x. Essentially we can only retrieve the ratios of the elements of x. I have adopted a convention of scaling the result so that the elements of $\tilde{x}$ are proportions that sum to 1:

For the example problem we have

$$\tilde{x} = \begin{pmatrix} 0.1979 \\ 0.2396 \\ 0.3021 \\ 0.2604 \end{pmatrix} \tag{5}$$

II.2 Circuit details

My HHL circuit for a 4x4 matrix and three clock qubits is shown in figures 1, 3 and 4. The first register, labelled b, with two qubits, begins with the input state $\tilde{b}$, and carries the result $\tilde{x}$ at the end. The second register, labelled q, with three qubits, is the clock register. The third register, labelled a, with one qubit, is the ancilla register.

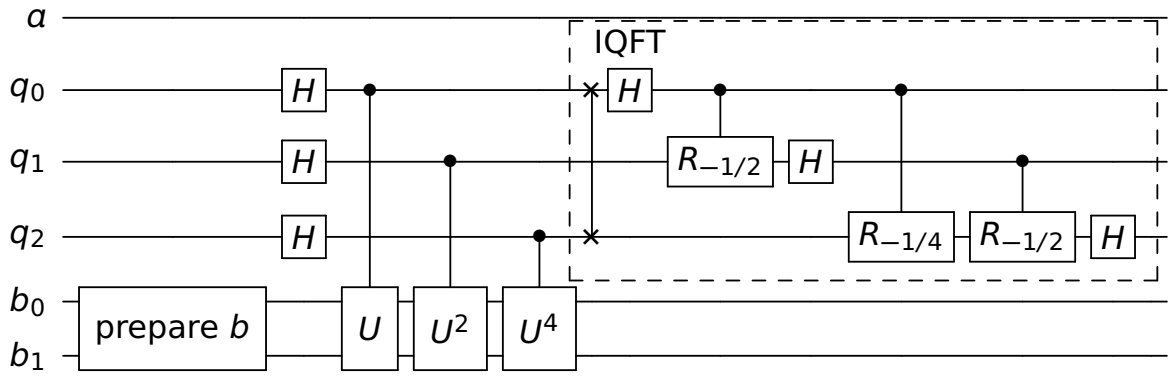


Figure 1: State preparation and QPE

Figure 1 shows the state preparation and QPE. Because the goal of my implementation is to understand the algorithm, not to provide a practical implementation, I have provided the input state directly in Mathematica using the `QuantumTensorProduct[]` function:

```
psi1 =
    QuantumTensorProduct[
        Join[{QuantumState[Normalize[b]]},
              ConstantArray[QuantumState[{1, 0}], n + 1]]]
```

In real world applications, state preparation is a nontrivial problem, as discussed in [1] and [4].

For the phase estimation part of the circuit, following [12], I calculated the eigenvalues λ_j of the matrix A and a matrix V of the eigenvectors of A, and then the unitary operator U as follows:

$$t = \frac{\pi}{\max\{\lambda_j\}} = \frac{3\pi}{4}$$

$$\Lambda = V^\dagger A V \qquad \text{(Diagonal matrix of eigenvalues)}$$

$$U = e^{i\Lambda t} \qquad \text{(Unitary operator)}$$

The details of these calculations are not shown here, but can be found in the accompanying Mathematica notebook.

This part of the circuit is conceptually the same as in [12], but because there are four values in the input $\tilde{b}$ and four unknowns in $\tilde{x}$, the unitary operation in the QPE has two target qubits instead of one. Also, because there are three clock qubits, in QPE we apply U, U^2 and U^4, and the IQFT subcomponent is extended to three qubits.

The 2x2 example in [12], shown in figure 2, uses a simple ancilla rotation with two separate rotations on the ancilla qubit, one for each clock qubit.

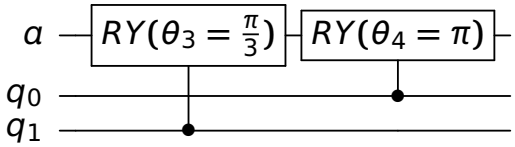


Figure 2: Ancilla rotation for 2x2

As discussed in class, I was confused about the number of rotations on the ancilla qubit for larger problems. It seemed to me from the pattern of control bits and rotations in the 2x2 circuit that there is one rotation per clock qubit.

However, for the algorithm to work perfectly and invert all the eigenvalues, there needs to be a rotation per eigenvalue. This can be achieved using a more complex arrangement of control qubits, and utilising a combination of “control-on”

and “control-off” control qubits. Figure 3 shows the ancilla rotation for the example 4x4 problem. In this circuit there are four rotations, with angles θ_1, θ_2, θ_3 and

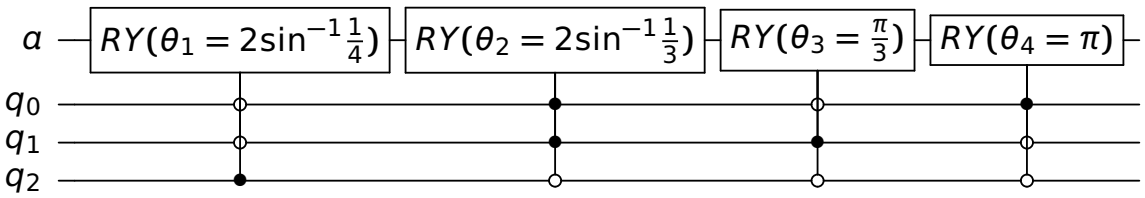


Figure 3: Ancilla rotation

θ_4, which correspond to the eigenvalues λ_1, λ_2, λ_3 and λ_4 of the matrix A. They are controlled by the clock qubits according to the binary representation of the eigenvalue phases on the clock qubits.

For example, the largest eigenvalue is $\lambda_1 = \frac{4}{3}$. In the Hamiltonian $e^{i\Lambda t}$ this eigenvalue has corresponding value $e^{i\lambda_1 t} = e^{i\frac{4}{3}\times\frac{3\pi}{4}} = e^{2\pi i\frac{1}{2}}$. Therefore the phase for this eigenvalue is $\phi_1 = \frac{1}{2}$, which in binary is 0.100b. Therefore the rotation for this eigenvalue should occur when qubits q_2, q_1, q_0 are in the pattern 100, i.e. q_2 is set and qubits q_1 and q_0 are not set. The rotation angle is obtained via the formula from [12]: $\theta_1 = 2\sin^{-1}\bar{\lambda_1}$, where $\bar{\lambda_1}$ is the eigenvalue scaled by a factor that makes all the eigenvalues integers:

$$\bar{\lambda_j} = \frac{2^n \lambda_j t}{2\pi}$$

In this way we obtain the phases, controlling qubits, and rotation angles for all the eigenvalues, shown in the table below:

Eigenvalue	$e^{i\lambda_j t} = e^{2\pi i\phi_j}$	Phase	Qubits	Angle
$\lambda_1 = \frac{4}{3}$	$e^{i\lambda_1 t} = e^{i\frac{4}{3}\times\frac{3\pi}{4}} = e^{2\pi i\frac{1}{2}}$	$\phi_1 = \frac{1}{2}$ =0.100b	$q_2\overline{q_1}\overline{q_0}$	$\theta_1 = 2\sin^{-1}\frac{1}{4}$
$\lambda_2 = 1$	$e^{i\lambda_2 t} = e^{i1\times\frac{3\pi}{4}} = e^{2\pi i\frac{3}{8}}$	$\phi_2 = \frac{3}{8}$ =0.011b	$\overline{q_2}q_1q_0$	$\theta_2 = 2\sin^{-1}\frac{1}{3}$
$\lambda_3 = \frac{2}{3}$	$e^{i\lambda_3 t} = e^{i\frac{2}{3}\times\frac{3\pi}{4}} = e^{2\pi i\frac{1}{4}}$	$\phi_3 = \frac{1}{4}$ =0.010b	$\overline{q_2}q_1\overline{q_0}$	$\theta_3 = 2 = 2\sin^{-1}\frac{1}{2} = \frac{\pi}{3}$
$\lambda_4 = \frac{1}{3}$	$e^{i\lambda_4 t} = e^{i\frac{1}{3}\times\frac{3\pi}{4}} = e^{2\pi i\frac{1}{8}}$	$\phi_4 = \frac{1}{8}$ =0.001b	$\overline{q_2}\overline{q_1}q_0$	$\theta_4 = 2 = 2\sin^{-1}\frac{1}{1} = \pi$

The control qubit configurations and angles in the table correspond to the controlled rotations on the ancilla qubit in the circuit portion in Figure 3.

In real world implementations, where there might be billions of eigenvalues, it would be prohibitive to have separate rotations for each eigenvalue. By being selective and careful about the rotations, an approximate solution satisfies the accuracy bounds in [4]. A recent paper, [7], discusses filtering rotations “by

relevance” to maintain circuit depth advantages. Other approaches are suggested elsewhere, including some of the papers discussed in section IV.

Figure 4 shows the reverse QPE and measurement. As in [12], if the ancilla qubit is measured in state $|1\rangle$, then the b register holds the correct $\tilde{x}$ result. If the ancilla qubit is measured in state $|0\rangle$, the result is discarded and the circuit must be run again.

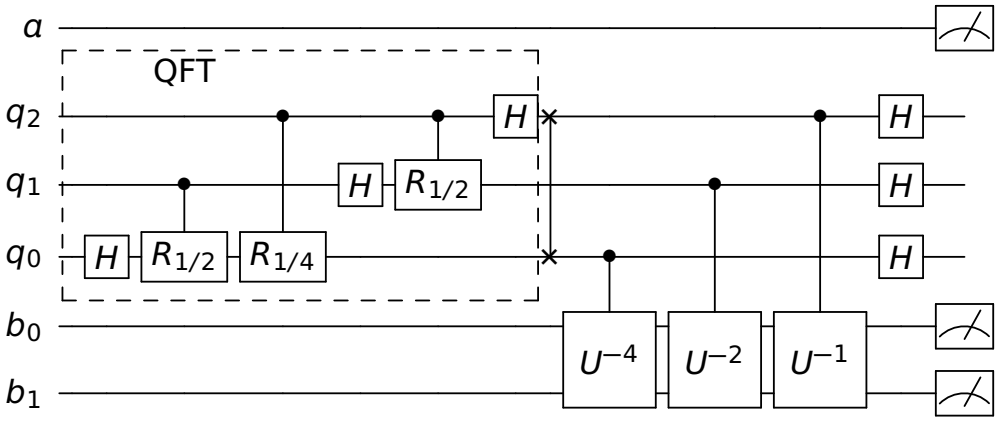


Figure 4: Inverse QPE and measurement

The accompanying Mathematica notebook shows all of the calculations for the example and verifies the circuit results. The Mathematica version of the circuit is shown in figure 5. (The Mathematica circuit is “upside down” because Mathematica uses a different convention for the order of qubits in the circuit diagram: it places the least-significant qubit at the bottom.)

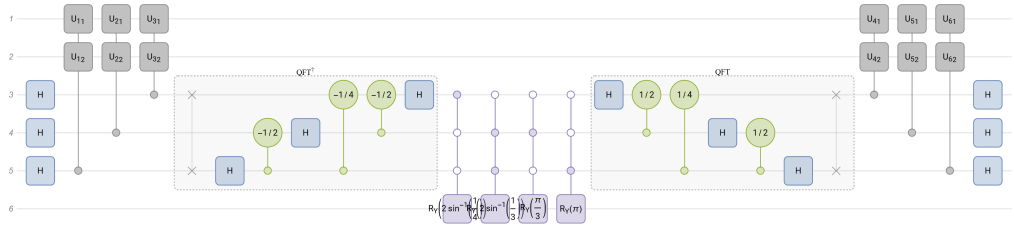


Figure 5: Mathematica circuit for 4x4 problem

II.3 Qiskit circuit implementation

I wrote a Qiskit program to construct the same circuit for the 4x4 problem and run it on the IBM Quantum simulator. A sample run looks like this:

```
bin/project_hhl_4x4_qiskit.py --shots=10000 --filename=project_hhl_4x4.png
000000  434
```

```
000001  1345    0.1476  0.1944
010000  1
010001  2053    0.2252  0.2402
100000  450
100001  3274    0.3592  0.3033
110001  2443    0.2680  0.2620
Prob(ancilla |0>) = 0.9115
```

The first part of the output is a table in four columns:

- ✿ The measurement bits: $b_1 b_0 q_2 q_1 q_0 a$.
- ✿ The number of times that measurement was observed (the count).
- ✿ The frequency of that measurement: count/total.
- ✿ The square root of the frequency, corresponding to the estimate for the corresponding $\tilde{x}_i$.

Measurements with the ancilla bit (the least significant, or rightmost bit in the output) having value 1 are counted in the result, and measurements with the ancilla bit having value 0 are discarded.

Following this table is the frequency that the ancilla bit was measured in state $|0\rangle$ (having value 1), in this case 0.9115, indicating a good effectiveness for the algorithm with these data.

The output from this run is shown more clearly in table 1, along with the actual x values for comparison. We can see from the results that with 10,000 shots, we have got pretty good estimates for x.

Measurement	Count	Frequency	$\tilde{x}$ estimate	$\tilde{x}$ actual
000000	434			
000001	1345	0.1476	0.1944	0.1979
010000	1			
010001	2053	0.2252	0.2402	0.2396
100000	450			
100001	3274	0.3592	0.3033	0.3021
110001	2443	0.2680	0.2620	0.2604

Table 1: Results for 4x4 system

Figure 6 shows the Qiskit circuit for the 4x4 problem.

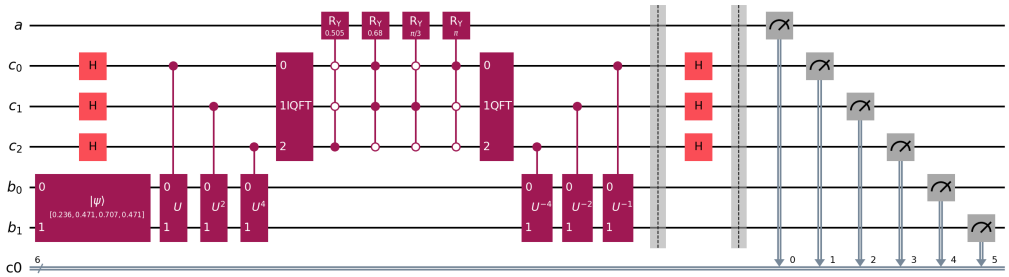


Figure 6: Qiskit circuit for 4x4 problem

III Generalization

I created a more general Qiskit program to explore the behaviour and limits of the algorithm with larger matrices and more clock qubits. I also created an 8x8 Hermitian matrix using a similar procedure to above. This Qiskit program prints the actual $\tilde{x}$ solution values (scaled appropriately), as well as the estimates, to make comparison easier.

I verified the program on the 2x2 example of [12]:

```
bin/project_hhl_qiskit.py --shots=10000 --clockqubits=2 \
    data/project_hhl/A_2x2.csv data/project_hhl/b_2x2.csv
bits    count   freq    x estimate      x actual
0000    1931
0001    600     0.0600  0.2472  0.2500
1000    1904
1001    5565    0.5565  0.7528  0.7500
0.6165  Prob(ancilla |0>)
```

These results, both the 0.25/0.75 values for x, and the ancilla probability, agree with the results in [12].

The 2x2 circuit it produces is shown in figure 7.

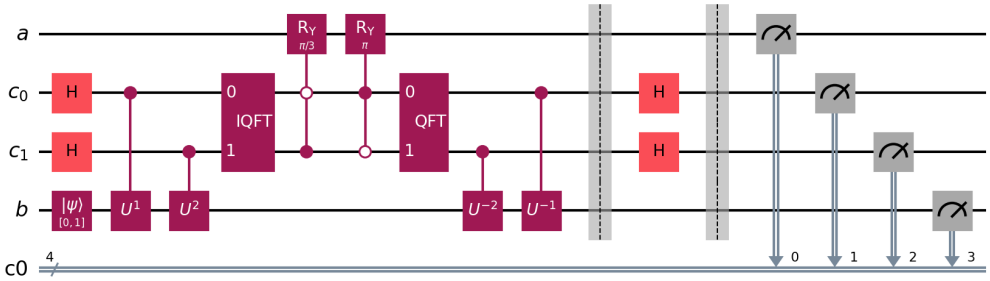


Figure 7: Qiskit circuit for 2x2 problem

I also verified the program on my 4x4 example:

```
bin/project_hhl_qiskit.py --shots=10000 --clockqubits=3 \
    data/project_hhl/A_4x4.csv data/project_hhl/b_4x4.csv
bits    count   freq    x estimate      x actual
000000  438
000001  1380    0.1380  0.1972  0.1979
010001  2015    0.2015  0.2383  0.2396
100000  477
100001  3268    0.3268  0.3034  0.3021
```

```
110000  1
110001  2421    0.2421  0.2612  0.2604
0.9084  Prob(ancilla |0>)
```

For the 8x8 example I ran the program as follows:

```
bin/project_hhl_qiskit.py --shots=1000000 --clockqubits=6 \
    data/project_hhl/Ahalf_8x8.csv data/project_hhl/b_8x8.csv
```

The output is shown in table 2.

Measurement	Count	Frequency	x estimate	x actual
0000000000	11019			
0000000001	26	0.0000	0.0091	0.0091
0010000000	77671			
0010000001	609	0.0006	0.0440	0.0435
0100000000	80853			
0100000001	10871	0.0109	0.1859	0.1858
0110000000	12558			
0110000001	451	0.0005	0.0379	0.0377
1000000000	230535			
1000000001	4079	0.0041	0.1139	0.1141
1010000000	50226			
1010000001	370	0.0004	0.0343	0.0329
1100000000	110700			
1100000001	6789	0.0068	0.1469	0.1476
1110000000	345591			
1110000001	57652	0.0577	0.4281	0.4293

Table 2: Results for 8x8 system

The frequency of the ancilla bit being measured in state $|0\rangle$ was only 0.0808, which is why I needed to run with 1,000,000 shots to get an accurate result. The fact that the result was correct, yet the ancilla frequency is so low, is troubling. I tried several things to understand the cause of this:

- ✿ Ensure the matrix is well-conditioned, as described in [4].
- ✿ Ensure the matrix is sparse.
- ✿ Construct a Hermitian matrix more directly as two blocks of a a 4x4 matrix and its conjugate transpose.

None of these measures made much difference to results: the x estimate was always good, but the ancilla frequency was always low.

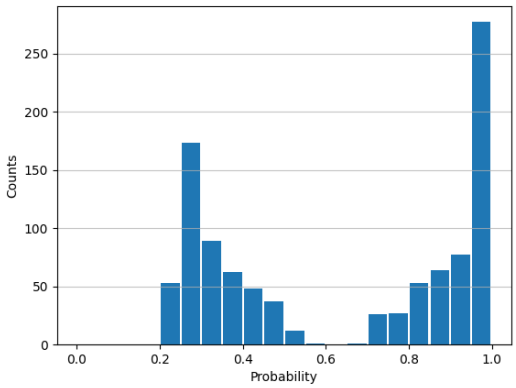
I also tried two variations of the eigenvalue scaling and clock qubit arrangement. In my default setup I scale the eigenvalues so that the largest one is scaled to $\frac{1}{2}$=0.100... (base 2), following the example in [12]. But this is wasteful, because it effectively reduces the range of values that can be represented by the clock qubits by half. So I tried another scheme where I scaled so that the largest eigenvalue is scaled to 0.111... (base 2), making maximum use of the clock qubits. (For this I needed to use a different input matrix, chosen to make the eigenvalue ratios fit the clock qubit scheme.)

I noticed that although I got a high probability of about 0.9 of measuring the ancilla qubit in the $|0\rangle$ in my 4x4 example, for the 2x2 example in [12] the probability was only $\frac{5}{8}$ = 0.625. Therefore, I decided to examine the distribution of the ancilla probability for a variety of random problems. The probability can in principle be calculated analytically for a given problem, but I obtained a statistical distribution from random trials by actually running the circuits (in the simulator).

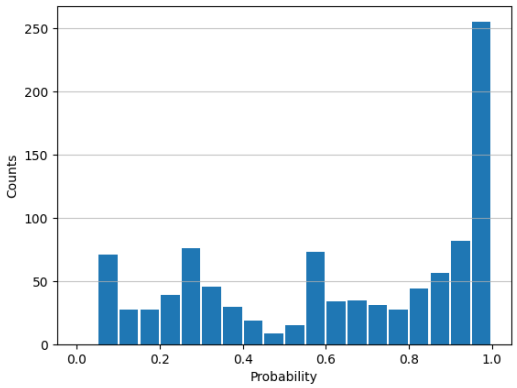
Figure 8 shows the distribution of the ancilla probability for random 2x2 problems with 2, 3, 4 and 5 clock qubits. It appears that the number of clock qubits does not affect the distribution much. But the distribution is quite spread, and the particular shape probably is a result of the way I constructed random instances.

The performance and accuracy guarantees of [4] places bounds on the ancilla probability that depend on features of the specific problem, so I think my 8x8 example was just a bit unlucky in its parameters, even though I was not able to improve things by tuning the parameters. By comparison, my 4x4 example happened to be a lucky one with a relatively high ancilla probability. It would be interesting to study the way the ancilla probability depends on the problem parameters in more detail.

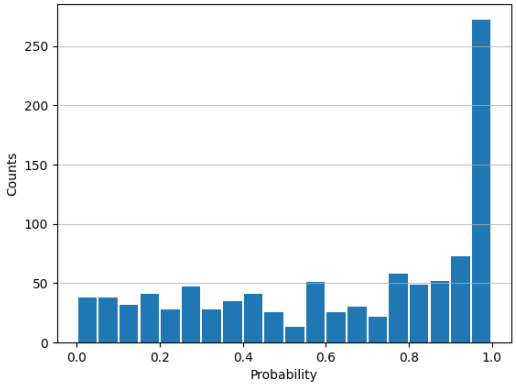
Figures 9 and 10 show some distributions of the ancilla probability for random 4x4 and 8x8 problems.



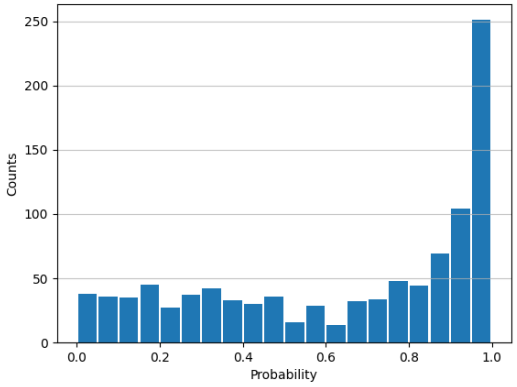
2 clock qubits



3 clock qubits

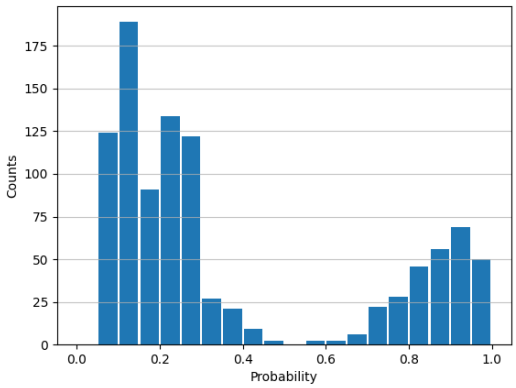


4 clock qubits

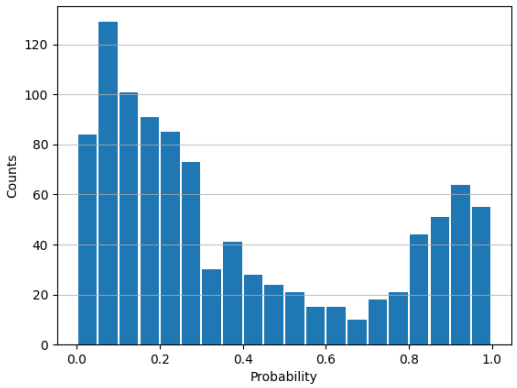


5 clock qubits

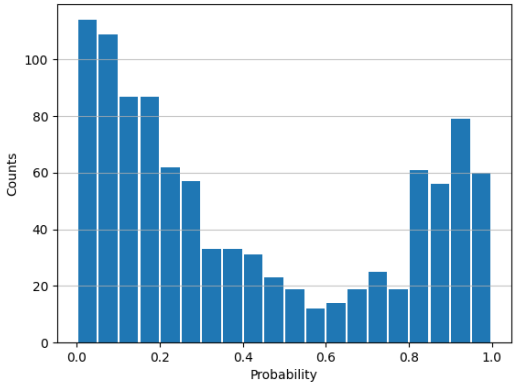
Figure 8: Ancilla state $|0\rangle$ probability distribution for random 2x2 problems



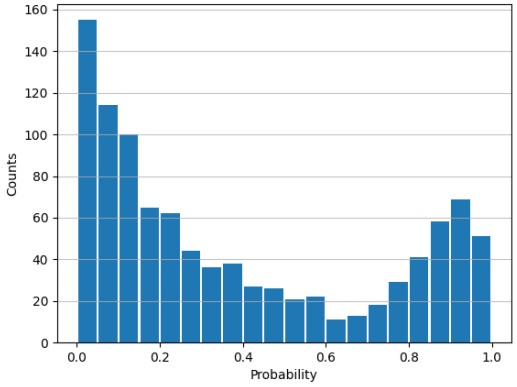
2 clock qubits



3 clock qubits

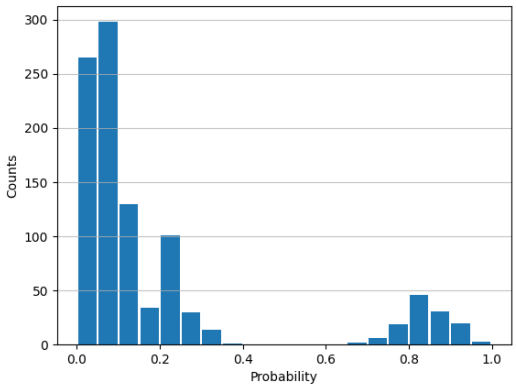


4 clock qubits

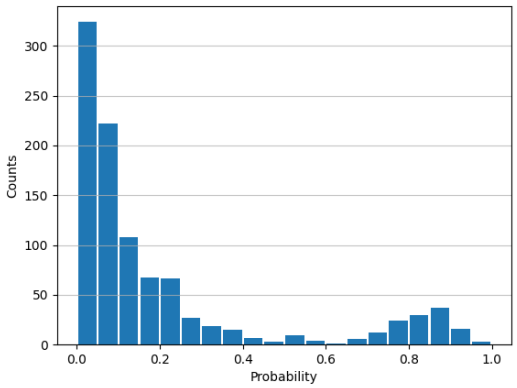


5 clock qubits

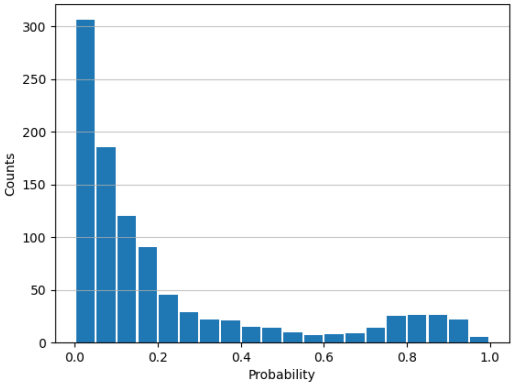
Figure 9: Ancilla state |0⟩ probability distribution for random 4x4 problems



4 clock qubits



5 clock qubits



6 clock qubits

Figure 10: Ancilla state |0⟩ probability distribution for random 8x8 problems

IV Developments

I made extensive use of Google Scholar, ChatGPT and Google Gemini in searching for, citing, and summarizing literature in this section. Although some of summaries are generated by LLMs, this was a result of considerable searching, filtering and prompting on my part.

Scott Aaronson's paper "Read the fine print" [1] is a good starting point for understanding the limits and applicability of the HHL algorithm. The paper presents a "HHL checklist of caveats":

- How to make the input vector b available efficiently to the quantum circuit.
- How to provide the unitary transformations e^{iAt} efficiently in the quantum circuit.
- Conditions on the input matrix A, that it must be sparse and well-conditioned.
- Limits on output: if every x_i is required, the quantum speed-up is lost.

Aaronson states that (as of 2015) he knew of only "two attempts to work out potential applications of the HHL template from start to finish": the papers by Clader, Jacobs and Sprouse [3] and Wang [11].

Clader, Jacobs and Sprouse generalize HHL in several respects. They:

- develop a state preparation routine that can initialize generic states,
- show how simple ancilla measurements can be used to calculate many quantities of interest, and
- integrate a quantum-compatible preconditioner that greatly expands the number of problems that can achieve exponential speedup over classical linear systems solvers.

They provide an application in electromagnetic scattering to demonstrate an exponential speedup over classical methods.

Wang's paper is focused on the problem of analyzing large sparse electrical networks. Wang describes two classes of quantum algorithms for this problem:

- ✿ The first class is based on extracting information from solutions to linear systems. This approach builds on the Childs, Kothari and Somma[10] improvement of HHL. Wang's approach avoids some of the disadvantages of HHL and other QLSAs by outputting a single number, which might be the norm of the solution, the norm of one entry of the solution, or the norm of the difference of two entries of the solution.
- ✿ The second class of algorithms use quantum random walks, an important quantum algorithmic technique, to analyze the graph structure of the problem.

I did not find any further developments in the applications of electromagnetic scattering or electrical networks. But I did learn about several other interesting developments in relation to HHL.

IV.1 Resource requirements

Scherer et al[8] conducted a detailed resource analysis of the quantum linear system algorithm (QLSA) proposed by Harrow, Hassidim, and Lloyd (HHL) and its generalization by Clader, Jacobs, and Sprouse. The goal of this analysis was to estimate the resources required to implement the QLSA on a realistic quantum computer for a specific problem size where the quantum algorithm is expected to outperform classical methods.

The authors focused on a concrete problem: calculating the electromagnetic scattering cross-section of a 2D target using the finite element method (FEM). They analyzed the algorithm for a problem size of N=332,020,680, which is beyond the estimated "cross-over point" where the quantum algorithm is expected to surpass classical algorithms in terms of speed.

The resource estimation involved determining the number of qubits, gate operations, circuit width, circuit depth, and T-depth (the number of time steps with T-gates) required for the algorithm. The analysis was done for a target accuracy of ε=0.01.

The key findings of the analysis were:

- ✿ Resource requirements are high: The analysis revealed that a significant amount of resources would be needed for the QLSA to outperform classical methods. The estimated circuit depth was on the order of 10^{25} (excluding

oracle costs) or 10^{29} (including oracle costs), and the number of qubits required was around 340 or 10^8, respectively.

- ✿ Oracle costs are substantial: The resource requirements for implementing the oracles used in the algorithm were found to be considerable, contributing significantly to the overall resource estimates.
- ✿ Lack of parallelism: The algorithm's design showed a lack of parallelism, meaning that most gate operations must be performed sequentially rather than simultaneously, limiting the potential for speedup.
- ✿ Hamiltonian simulation is a bottleneck: The Hamiltonian simulation step was identified as the most resource-intensive part of the algorithm, contributing the most to the overall circuit depth.

The authors concluded that while their estimates might be conservative, they provide a baseline for future research aimed at reducing the resource requirements of the QLSA. They emphasized that a substantial reduction in resources is necessary for the algorithm to become practical for real-world applications. The study also highlighted the importance of considering the resource costs of oracle implementations, which are often overlooked in theoretical analyses.

IV.2 Improved precision dependence

Somma, Childs, and Kothari[10] present two quantum algorithms that enhance the Harrow, Hassidim, and Lloyd (HHL) algorithm for solving linear systems of equations. The key improvement lies in the exponential reduction of the dependence on precision (i.e., the error tolerance, ε) from poly$(\frac{1}{\varepsilon})$ to poly$(\log(\frac{1}{\varepsilon}))$. This is achieved while maintaining a similar dependence on other parameters like the condition number (κ) and sparsity (d) of the matrix.

The authors achieve this improvement by employing two distinct approaches:

1. Fourier Approach: This approach decomposes the inverse of the matrix A^{-1} as a linear combination of unitary matrices e^{-At_j}, where t_j represents different evolution times. These unitaries can be implemented using Hamiltonian simulation techniques. By approximating the inverse function ($1/x$) as a linear combination of complex exponentials, the algorithm can directly apply the inverse without relying on phase estimation to extract eigenvalues.

2. Chebyshev Approach: This approach utilizes a Chebyshev polynomial expansion to approximate A^{-1}. These polynomials can be efficiently implemented using quantum walks. The algorithm directly uses the oracle for the matrix entries, bypassing the need for Hamiltonian simulation and phase estimation as a subroutines. Chebyshev polynomials offer a more efficient representation of the inverse function, leading to better dependence on ε and κ.

Phase estimation is a quantum algorithm used to estimate eigenvalues of unitary operators. While powerful, it has a significant drawback: its complexity scales linearly with the inverse of the desired precision ($1/\varepsilon$). This means that as the desired precision increases, the running time of the algorithm increases proportionally.

By avoiding phase estimation, Somma et al. achieve an exponential improvement in the dependence on precision. Their algorithm's complexity scales polylogarithmically with $1/\varepsilon$, meaning that the running time increases much more slowly as the desired precision increases. This improvement is particularly beneficial when the QLSA is used as a subroutine multiple times, as the overall error accumulates with each use.

Their two approaches offer a trade-off in terms of applicability and efficiency. The Fourier approach is more general, working with any efficiently simulable Hamiltonian, even if it's not sparse. The Chebyshev approach, while more efficient in terms of κ and ε dependence, is limited to sparse Hamiltonians. The authors also address the issue of ill-conditioned matrices (where the condition number is high) by incorporating a technique called variable-time amplitude amplification. This technique allows for a more efficient solution when the input state is a superposition of eigenvectors with a wide range of eigenvalues.

Overall, the work of Somma et al. significantly enhances the practicality of the HHL algorithm by improving its precision dependence and offering alternative approaches for implementation. These improvements have implications for various applications, including speeding up the finite element method and estimating hitting times of Markov chains.

IV.3 Singular value estimation

Kerenidis and Prakash[6] introduced a quantum algorithm for singular value estimation (SVE) that operates in a time complexity of $O(\text{polylog}(mn)/\varepsilon)$, where m

and n are the dimensions of the input matrix and ϵ is the precision parameter. This algorithm is a key component in their proposed quantum recommendation system.

The core idea behind their approach is to leverage a quantum data structure that allows for efficient storage and access to matrix elements in superposition. This data structure enables the quantum algorithm to create quantum states corresponding to the rows or columns of the matrix efficiently.

The SVE algorithm then proceeds by mapping the input vector onto the space spanned by the singular vectors of the matrix. This is achieved by applying specific isometries (norm-preserving transformations) that can be efficiently implemented on a quantum computer.

The algorithm then utilizes phase estimation to estimate the eigenvalues of a unitary matrix derived from the input matrix. These eigenvalues are directly related to the singular values of the input matrix. By obtaining estimates of the eigenvalues, the algorithm effectively estimates the singular values.

Finally, the algorithm uncomputes the intermediate steps to recover the original quantum state, now enriched with information about the singular values.

The key advantage of this quantum SVE algorithm is its polylogarithmic time complexity in the matrix dimensions, which is exponentially faster than classical SVE algorithms for matrices with low rank. This efficiency is crucial for the quantum recommendation system proposed by Kerenidis and Prakash, as it allows for the efficient sampling of high-value elements from a low-rank approximation of the preference matrix, leading to personalized recommendations for users.

IV.4 Circulant preconditioner

Shao and Xiang[9] extend the work of Kerenidis and Prakash by applying the SVE technique to solve linear systems with a circulant preconditioner. While Kerenidis and Prakash focused on using SVE for quantum recommendation systems, Shao and Xiang adapt and modify the SVE method to address the specific challenges of preconditioned linear systems, particularly in the context of general dense non-Hermitian matrices. This modification allows them to efficiently construct the quantum state of the preconditioner and solve the preconditioned linear system, demonstrating the versatility of the SVE technique in tackling various linear algebraic problems in quantum computing.

The authors highlight that the circulant preconditioner, denoted as C, offers several advantages. It is applicable to general dense linear systems, unlike some other preconditioners that are limited to specific matrix types. Additionally, the circulant preconditioner possesses a simple structure, allowing it to be diagonalized by the quantum Fourier transform, which can be efficiently implemented on a quantum computer.

The primary challenge in constructing the circulant preconditioner lies in the computation of its eigenvalues. Direct calculation of these eigenvalues would be computationally expensive, negating the quantum speedup. To address this, the authors propose a method to construct the quantum state of the preconditioner's diagonal elements efficiently, avoiding the need for explicit calculation.

The proposed quantum algorithm aims to solve the preconditioned linear system $C^{-1}Ax = C^{-1}b$. The modified SVE technique is employed to obtain the singular value decomposition of $C^{-1}A$, which is crucial for solving the preconditioned system. The authors emphasize that their modified SVE method is particularly suitable for cases where the matrix is provided as quantum information, as it avoids the need to expand the matrix into a Hermitian form.

The time complexity of the algorithm depends on the condition numbers of both the circulant preconditioner C and the preconditioned matrix $C^{-1}A$, as well as the Frobenius norm of the original matrix A. The authors suggest that for a well-chosen preconditioner, the condition numbers of C and $C^{-1}A$ can be assumed to be relatively small, potentially leading to a simplified complexity expression.

Overall, Shao and Xiang's work presents a novel quantum algorithm for solving linear systems using a circulant preconditioner. The modified SVE technique and the efficient construction of the preconditioner's quantum state are key contributions. The authors also discuss potential future directions, such as improving the algorithm's dependence on the Frobenius norm of the input matrix and exploring the possibility of achieving even better precision dependence.

References

[1] Scott Aaronson. "Read the fine print". In: *Nature Physics* 11.4 (2015), pp. 291–293.

[2] Belal Ehsan Baaquie and Leong-Chuan Kwek. *Quantum computers: Theory and algorithms*. Springer Nature, 2023.

[3] B David Clader, Bryan C Jacobs, and Chad R Sprouse. “Preconditioned quantum linear system algorithm”. In: *Physical review letters* 110.25 (2013), p. 250504.

[4] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. “Quantum Algorithm for Linear Systems of Equations”. In: *Phys. Rev. Lett.* 103 (15 Oct. 2009), p. 150502. DOI: 10.1103/PhysRevLett.103.150502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.103.150502.

[5] Jack D. Hidary. *Quantum Computing: An Applied Approach*. 1st ed. Springer, 2021. ISBN: 978-3-030-83273-5.

[6] Iordanis Kerenidis and Anupam Prakash. “Quantum recommendation systems”. In: *arXiv preprint arXiv:1603.08675* (2016).

[7] Jack Morgan, Eric Ghysels, and Hamed Mohammadbagherpoor. “An Enhanced Hybrid HHL Algorithm”. In: *arXiv preprint arXiv:2404.10103* (2024).

[8] Artur Scherer, Benoît Valiron, Siun-Chuon Mau, Scott Alexander, Eric Van den Berg, and Thomas E Chapuran. “Concrete resource analysis of the quantum linear-system algorithm used to compute the electromagnetic scattering cross section of a 2D target”. In: *Quantum Information Processing* 16 (2017), pp. 1–65.

[9] Changpeng Shao and Hua Xiang. “Quantum circulant preconditioner for a linear system of equations”. In: *Physical Review A* 98.6 (2018), p. 062321.

[10] Rolando Somma, Andrew Childs, and Robin Kothari. “Quantum linear systems algorithm with exponentially improved dependence on precision”. In: *APS March Meeting Abstracts*. Vol. 2016. 2016, H44–001.

[11] Guoming Wang. “Efficient quantum algorithms for analyzing large sparse electrical networks”. In: *Quantum Information & Computation* 17.11-12 (2017), pp. 987–1026.

[12] Anika Zaman, Hector Jose Morrell, and Hiu Yung Wong. “A Step-by-Step HHL Algorithm Walkthrough to Enhance Understanding of Critical Quantum Computing Concepts”. In: *IEEE Access* (2023).