Implementing and Evaluating the Efficiency of a Quantum Linear Systems Algorithm

Dhruv Yalamanchi

North Carolina School of Science and Mathematics

Research Objectives

- To implement the HHL algorithm in Qiskit and determine its dependence on input size N, condition number κ, and sparsity s
- To experimentally determine and compare the time complexities for both the HHL algorithm and Gaussian elimination with partial pivoting
- To approximate the threshold ranges for N at which the HHL algorithm surpasses the classical algorithm in efficiency

Motivation

- Linear systems of equations play a key role in many areas of science and technology including machine learning, approximation models, and electrical circuit analysis

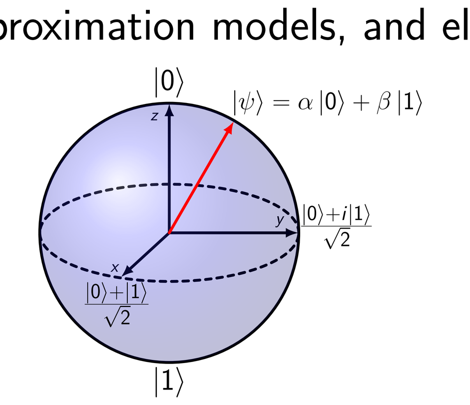

Figure 1: Bloch sphere representation of a single qubit in an arbitrary superposition state. The red arrow indicates the qubit's statevector.

- Figure 1 illustrates how a pure qubit state $|\psi\rangle$ with $|\alpha|^2 + |\beta|^2 = 1$ can be mapped to a point on the surface of a unit sphere as a superposition of the basis states $|0\rangle$ and $|1\rangle$
- By leveraging properties such as superposition and entanglement, quantum computing has been able to achieve considerable speedups over classical methods for numerous tasks

Quantum Linear Systems Problem (QLSP):

Given the $N \times N$ system

$$\underbrace{\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & \cdots & a_{N,N} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}}_{\vec{b}}$$

where only $A \in \mathbb{C}^{N\times N}$ and $\vec{b} \in \mathbb{C}^N$ are known, prepare a quantum state $|x\rangle$ proportional to the solution vector $\vec{x}$

- The Harrow-Hassidim-Lloyd (HHL) algorithm is a quantum algorithm that treats the QLSP when A is sparse and Hermitian
- The complexity and accuracy of the HHL output depend on characteristics such as N and A's sparsity s (defined as the percentage of elements that equal zero) and condition number κ (defined as the ratio of the matrix's largest and smallest eigenvalues)

HHL Algorithm

Input:

- A, an $N \times N$ Hermitian matrix with N distinct eigenvalues $\lambda_j \in \mathbb{R}$ and eigenvectors $|u_j\rangle \in \mathbb{R}^N$
- $\vec{b}$, an N-dimensional vector that, when normalized, has coefficients $b_j \in \mathbb{C}$
- n, the number of qubits in the clock register

Procedure:

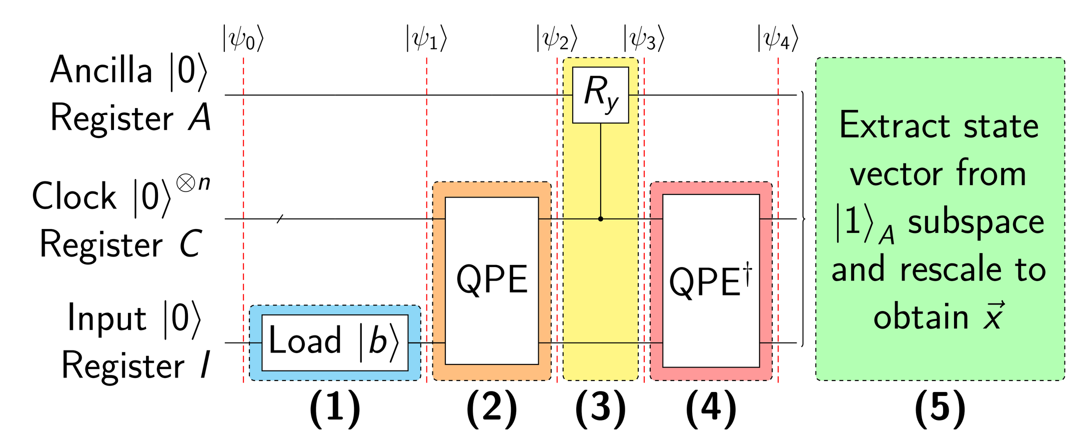

Figure 2: Quantum circuit diagram for the HHL algorithm. Ancilla register A contains one ancillary qubit upon which we perform the conditional rotation R_y. Clock register C contains the n qubits we use to estimate A's eigenvalues. Input register I contains $\log_2 N$ qubits and will store $|x\rangle$. Slice $|\psi_i\rangle$ indicates the total state after step i. The system is initialized to $|\psi_0\rangle = |0\rangle_A \otimes |0\rangle_C^{\otimes n} \otimes |0\rangle_I = |0\rangle_A |0\rangle_C^{\otimes n} |0\rangle_I$.

(1) $|\psi_0\rangle \xrightarrow{\text{Encode } |b\rangle \text{ into register } I} |\psi_1\rangle = \sum_{j=0}^{N-1} b_j |0\rangle_A |0\rangle_C^{\otimes n} |u_j\rangle_I$

(2) $|\psi_1\rangle \xrightarrow{\text{Perform phase estimation}} |\psi_2\rangle = \sum_{j=0}^{N-1} b_j |0\rangle_A |\tilde{\lambda}_j\rangle_C |u_j\rangle_I$

(3) $|\psi_2\rangle \xrightarrow{\text{Apply controlled rotation}}$

$$|\psi_3\rangle = \sum_{j=0}^{N-1} b_j \left(\sqrt{1 - \frac{\gamma^2}{\tilde{\lambda}_j^2}} |0\rangle_A + \frac{\gamma}{\tilde{\lambda}_j} |1\rangle_A \right) |\tilde{\lambda}_j\rangle_C |u_j\rangle_I$$

(4) $|\psi_3\rangle \xrightarrow{\text{Uncompute clock register}}$

$$|\psi_4\rangle = \sum_{j=0}^{N-1} b_j \left(\sqrt{1 - \frac{\gamma^2}{\tilde{\lambda}_j^2}} |0\rangle_A + \frac{\gamma}{\tilde{\lambda}_j} |1\rangle_A \right) |0\rangle_C^{\otimes n} |u_j\rangle_I$$

(5) $\underbrace{\sum_{j=0}^{N-1} \gamma \frac{b_j}{\tilde{\lambda}_j} |1\rangle_A |0\rangle_C^{\otimes n} |u_j\rangle_I}_{|1\rangle_A \text{ subspace of system}} \xrightarrow{\text{Rescale first } N \text{ elements}} \vec{x} \approx A^{-1}\vec{b}$

Output:

- Solution vector $\vec{x}$

Qiskit

- Qiskit is an open source software development kit that allows us to work with quantum computers, giving us access to multiple fundamental circuits, algorithms, simulators, and quantum hardware architectures
- We implement a general HHL algorithm circuit and execute it using the statevector simulator provided by Qiskit Aer

Complexity Analyses

- To compare against HHL, we implement Gaussian elimination with partial pivoting (GEPP), one of the most efficient classical linear systems algorithms, in Python
- GEPP computes the exact solution to a linear system by converting A into upper triangular form and then solving using back substitution
- To conduct the complexity analysis for GEPP, we randomly generated different matrices A (of specified sparsity) and vectors $\vec{b}$ for each $N \in \{32, 64, \ldots, 2048, 4096\}$

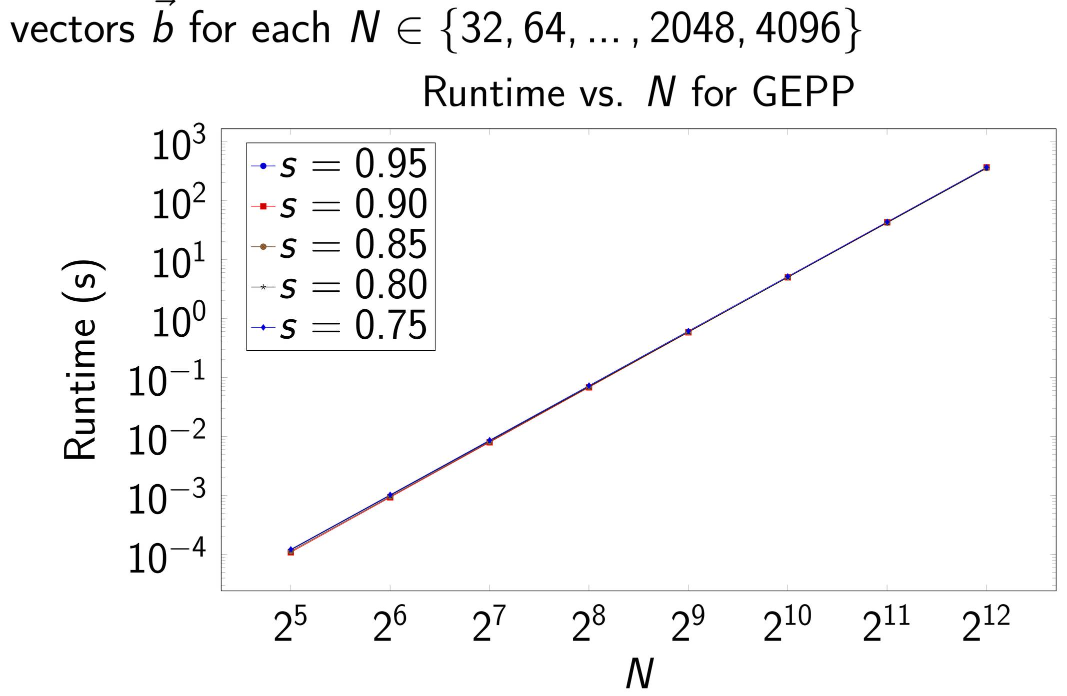

Figure 3: Scaling with N for the GEPP algorithm. Both runtime and N are plotted on a logarithmic scale. Lines of best fit are shown for $s = 0.75, 0.80, 0.85, 0.90, 0.95$. In each case, we averaged 30 runs of the GEPP algorithm.

- The lines of best fit shown in Figure 3 are nearly identical with average slope 3.0799 and intercept -8.5719
- Each data set has a correlation $r > 0.9998$, suggesting that GEPP's runtime (T) scales polynomially in N according to

$$T(N) = (2.6800 \times 10^{-9})N^{3.0799} \implies T(N) \approx O(N^3)$$

- s and κ had no observable impact on runtime
- To evaluate the HHL algorithm, we randomly generated different Hermitian matrices A (with sparsities specified in Table 1) and vectors $\vec{b}$ for each $N \in \{4, 8, 16, 32\}$

	Class #1	Class #2	Class #3	Class #4
$N=4$	12/16	11/16	10/16	9/16
$N=8$	56/64	54/64	52/64	50/64
$N=16$	240/256	232/256	224/256	216/256
$N=32$	992/1024	960/1024	928/1024	896/1024

Table 1: Table of sparsity classes for the HHL algorithm

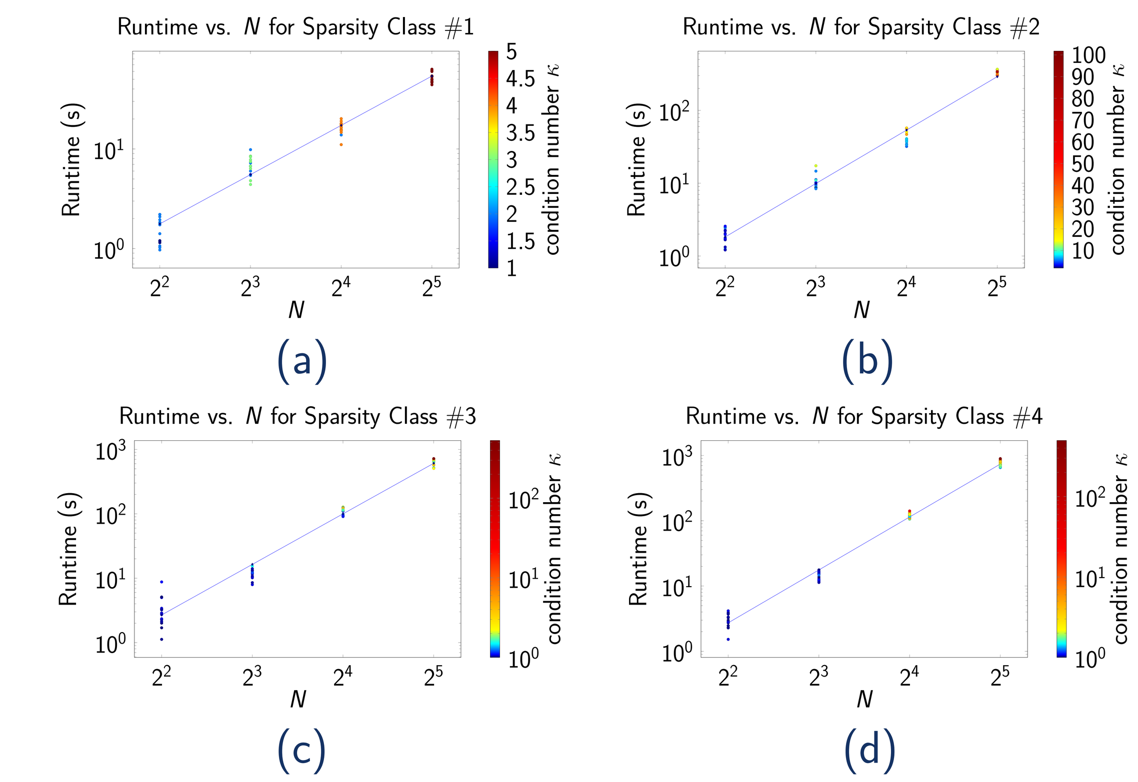

Figure 4: Scaling with N for the HHL algorithm for the four sparsity classes outlined in Table 1. For all graphs, both runtime and N are plotted on a logarithmic scale. Lines of best fit are shown for each graph. For each N and sparsity class, there are 15 data points.

	Class #1	Class #2	Class #3	Class #4
Correlation	0.9954	0.9972	0.9953	0.9980
Slope	1.6428	1.7381	1.8477	1.9531
Intercept	-0.7411	-1.2061	-1.1404	-1.1852
$T(N)$	$0.1815N^{1.6428}$	$0.0622N^{1.7381}$	$0.0724N^{1.8477}$	$0.0653N^{1.9531}$

Table 2: Table summarizing the results from Figure 4(a-d)

Conclusions and Future Work

- The HHL algorithm provided a polynomial speedup over GEPP, running in $O(N^m)$ with $1.5 < m < 2.0$ for all four sparsity classes
- Decreasing s resulted in more rapid scaling
- At all values of N there is a clear positive correlation between κ and runtime
- Decreasing s and raising κ lowered solution accuracy
- Based on our experimental time complexities, HHL would overtake GEPP at approximately $N = 2.8 \times 10^5, 3.1 \times 10^5, 1.1 \times 10^6, 6.3 \times 10^6$ for sparsity classes #1-#4, respectively
- We aim to assess the HHL algorithm's efficiency and accuracy for larger matrix sizes such as $N = 64$ and $N = 128$ to further support the observed trends in runtime
- We can model the accuracy and runtime tradeoff in the HHL algorithm when we introduce additional clock register qubits

Acknowledgements

- Dr. Jonathan Bennett, NC School of Science and Math
- Dr. Iman Marvian, Duke University
- NCSSM Foundation