ECE 396: Quantum Computing **Professor Ali Javadi-Abhari, Professor Hakan E. Türeci**

HHL Quantum Data Fitting

Name: David Weisberg, Tyler Benson Data: 03/29/23

Introduction

HHL

The HHL algorithm is a quantum algorithm for solving a linear system of the form

$$A\vec{x} = \vec{b}, \tag{1}$$

where A is a given square $N \times N$ matrix of coefficients. In an ideal world, one just inverts the matrix and gets

$$\vec{x} = A^{-1}\vec{b}. \tag{2}$$

However, inverting a matrix classically is computationally intensive and not practical. HHL however, takes advantage of representing transformations in an eigenbasis in order to do matrix inversion with quantum operators, ultimately leading to an exponential speedup.

HHL begins by assuming an invertible hermitian A. This assumption is reasonable, as we can define a new system with a hermitian matrix related to A. Indeed, if a given A is non-hermitian, then we can construct a new system

$$\tilde{A}\tilde{\vec{x}} = \tilde{\vec{b}}, \tag{3}$$

where

$$\tilde{A} = \begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix} \to \tilde{A}^\dagger = \tilde{A}; \tilde{\vec{b}} = \begin{bmatrix} \vec{b} \\ 0 \end{bmatrix}. \tag{4}$$

The matrix product in (3) works out to

$$\begin{bmatrix} A\tilde{\vec{x}}_1 \\ 0 \end{bmatrix} = \begin{bmatrix} \vec{b} \\ 0 \end{bmatrix} \text{ for } \tilde{\vec{x}} = \begin{bmatrix} \tilde{\vec{x}}_1 \\ \tilde{\vec{x}}_2 \end{bmatrix}. \tag{5}$$

The first row is solved, by definition in (1), and the second row is inconsequential, so we can thus let

$$\tilde{\vec{x}}_1 = \vec{x}, \tilde{\vec{x}}_2 = \vec{0}. \tag{6}$$

Thus, this new system solves for the solution we originally wanted in (1), but is compatible with HHL by using a hermitian matrix. So HHL can assume a hermitian A without loss of generality.

HHL also assumes, for simplicity, that the eigenvalues of A are between 0 and 1, and that the magnitudes of $\vec{x}$ and $\vec{b}$ are 1. This can all be done without loss of generality by merely scaling A.

In order to the invert the given matrix, HHL makes use of the following facts from linear algebra. Because A is hermitian and invertible, it admits an eigendecomposition of

$$A = \sum_{i=0}^{N-1} \lambda_i |u_i\rangle\langle u_i|, \tag{7}$$

where $|u_i\rangle$ are the orthonormal eigenvectors of A represented as quantum states and λ_i are its eigenvalues. This spectral decomposition for A makes sense, because for any desired vector in the eigenbasis,

$$y = \sum_{i=0}^{N-1} y_i |u_i\rangle, \tag{8}$$

applying the transformation A gives

$$A\vec{y} = \sum_{i=0}^{N-1} \lambda_i |u_i\rangle\langle u_i| \sum_{j=0}^{N-1} y_j |u_j\rangle = \sum_{j=0}^{N-1}\sum_{i=0}^{N-1} y_j \lambda_i |u_i\rangle\langle u_i|u_j\rangle = \sum_{i=0}^{N-1} y_i \lambda_i |u_i\rangle, \tag{9}$$

which is exactly the linear transformation A on its eigenvectors, just a scaling by its eigenvalues.
Since $\{u_i\}$ are an independent, orthogonal basis (due to invertibility), the inverse is found by just transforming back to the original eigenbasis of A, which is done by doing an inverse eigenvalue scaling of the eigenbases:

$$A^{-1} = \sum_{i=0}^{N-1} \frac{1}{\lambda_i} |u_i\rangle\langle u_i| \tag{10}$$

Thus, with $\vec{b}$ having coefficients β_i in the eigenbasis of A, the ultimate solution can be written down as

$$|x\rangle = A^{-1}|b\rangle = \sum_{i=0}^{N-1} \frac{1}{\lambda_i} |u_i\rangle\langle u_i| \sum_{j=0}^{N-1} \beta_j |u_j\rangle = \sum_{i=0}^{N-1} \frac{1}{\lambda_i} \beta_i |u_i\rangle. \tag{11}$$

Thus, following the linear algebra blueprint above, HHL seeks to, in principle, perform the following steps on a quantum circuit:

(a) Identify the eigenvalues of A.

(b) Invert the eigenvalues.

(c) Perform the arithmetic in (11), by doing doing a linear combination of the eigenvectors with coefficients as products between the inverted eigenvalues and the coefficients of $\vec{b}$ in the eigenbasis.

In order to do this computation, HHL represents the vectors as quantum states, where the normalized vector entries correspond to the amplitudes,

$$|b\rangle = \sum_{i=0}^{N-1} b_i |i\rangle. \tag{12}$$

Ultimately, the solution $\vec{x}$ will also be output as a unit normal quantum state vector, and thus the solution will be off by a normalization factor. This can be recovered, if desired by doing an appropriate measurement.

HHL uses Quantum Phase Estimation (QPE) in order to do (a), by solving for eigenphases of a unitary. In particular, for a unitary U and eigenvector $|u\rangle$, QPE solves for eigenvalue $e^{2\pi i\theta}$ by outputting the phase θ in binary form. It is important to note here that $0 \le 2\pi\theta < 2\pi \to 0 \le \theta < 1$ because it's a phase, so θ is always a fraction. Thus the output of QPE is always a fractional binary approximation $2^{n_l}\theta$ for n_l binary digits, or in this case qubits. HHL makes use of QPE by exponentiating A to e^{iA} so as to make it a unitary matrix suitable for QPE, which has the same eigenvectors as A, and eigenvalues $e^{i\lambda}$, where λ are the eigenvalues of A. This is the case because

$$U|u\rangle = e^{iA}|u\rangle = \left(I + \sum_{k=1}^{\infty} \frac{i^k}{k!} A^k\right)|u\rangle = \left(|u\rangle + \sum_{k=1}^{\infty} \frac{i^k}{k!} A^k |u\rangle\right) = \left(|u\rangle + \sum_{k=1}^{\infty} \frac{i^k}{k!} \lambda^k |u\rangle\right) = e^{i\lambda}|u\rangle. \tag{13}$$

HHL accomplishes this matrix exponentiation via quantum simulation, using the Lie product formula to do e^{iAt}. Thus, for an n_l qubit approximation for this U, QPE outputs $\tilde{\lambda} = 2^{n_l}\frac{\lambda t}{2\pi}$.
The question now becomes: how do we find the eigenvectors of U so as to input it into QPE in order to find the eigenvalues? HHL cleverly circumvents this question by directly applying QPE on $\vec{b}$. In particular, via (12), reexpressing $\vec{b}$ in the eigenbasis of A, and realizing the fact that QPE is a linear operation, we get that

$$\mathrm{QPE}\left(U, |b\rangle, |0\rangle_{n_l}\right) = \mathrm{QPE}\left(U, \sum_{k=0}^{N-1} \beta_k |u_k\rangle, |0\rangle_{n_l}\right) = \sum_{k=0}^{N-1} \beta_k \mathrm{QPE}\left(U, |u_k\rangle, |0\rangle_{n_l}\right) \tag{14}$$

$$= \sum_{k=0}^{N-1} \beta_k \mathrm{QPE}\left(U, |u_k\rangle, |0\rangle_{n_l}\right) = \sum_{k=0}^{N-1} \beta_k |u_k\rangle |\tilde{\lambda}_k\rangle_{n_l}, \tag{15}$$

where N is the dimension of A. Thus, this accomplishes step (a) by expressing the eigenvalues times a constant on a register of n_l qubits.

To accomplish steps (b) and (c), HHL entangles one more ancilla qubit in order to invert the eigenvalues and perform the final matrix inversion. HHL accomplishes this by doing a rotation on this ancilla qubit as a function of the eigenvalue represented in the $|\tilde{\lambda}_k\rangle_{n_l}$ register,

$$\sum_{k=0}^{N-1} \beta_k |u_k\rangle |\tilde{\lambda}_k\rangle_{n_l} \left(\cos\theta_k |0\rangle + \sin\theta_k |1\rangle\right). \tag{16}$$

In particular, HHL cleverly lets

$$\sin\theta_k = \frac{C}{\lambda_k} \text{ and } \cos\theta_k = \sqrt{1 - \frac{C^2}{\lambda_k^2}}, \tag{17}$$

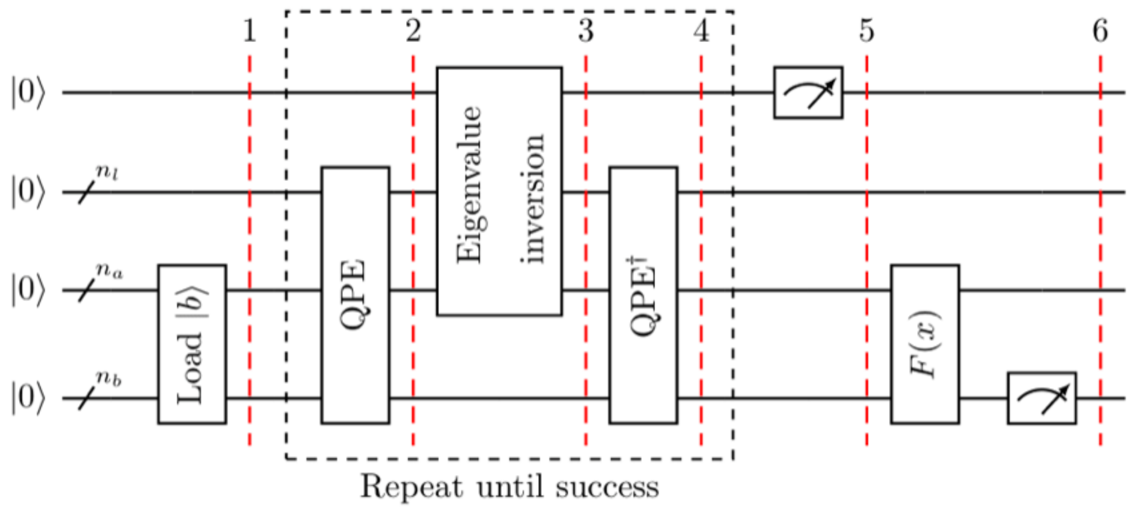
such that when a $|1\rangle$ is measured in this ancilla qubit, the state collapses to

$$\sum_{k=0}^{N-1} \frac{C}{\lambda_k} \beta_k |u_k\rangle |\tilde{\lambda}_k\rangle_{n_l} \theta_k |1\rangle, \tag{18}$$

where

$$C < \min_{k=0}^{N-1} \lambda_k \tag{19}$$

is a chosen constant, to be tuned that should be as big as possible in order to increase the chances that the conditional rotation (i.e. eigenvalue inversion) has succeeded. Indeed, the amplitudes on (18) in linear combination with the eigenvectors is the desired solution to the linear system we started out with, up to a normalization constant and C. The complete HHL circuit below from qiskit recapitulates all the desired operations, plus some less essential details.



Load $|b\rangle$ is the state preparation discussed by loading it into the register with n_b qubits. QPE does the operation discussed and stores the binary approximation in the register with n_l qubits. Eigenvalue inversion does the conditional rotation discussed onto the ancilla qubit up top, which is ultimately measured to determine if the inversion was succesful. QPE† merely restores the n_l qubits register so as to return it to a register of 0's. The operation F(x) is not too relavent to the proof of concept here, but in practice, for large theoretical N, sampling the vector to learn the amplitudes that correspond to the solution vector is not practical. The computational speedup of HHL is only maintained by doing some measurement operation on some function of the state.

Runtime

A hallmark of HHL is that it affords an exponential speedup on the order of the size of the matrix, N, on the problem of matrix inversion, i.e. solving linear systems, compared to the best classical algorithms. In particular, the runtime of HHL is

$$O(\kappa^2 \log(N) s^2/\epsilon), \tag{20}$$

compared to

$$O(Ns\sqrt{\kappa}\log 1/\epsilon), \tag{21}$$

the best classical algorithm. However, there are several practical considerations in HHL that impact its fidelity and practicability that can prove large within the runtime.

The first is C, the constant associated with egienvalue inversions, should be as great as possible in order to increase the likelihood of successful eigenvalue inversion. A poor C can negatively impact the run time, requiring more shots to have a successful output. Thus, the runtime is affected by C/λ_k for the largest eigenvalue. In particular, since C must be less than the smallest eigenvalue, C is $O(1/\kappa)$. Since C^2 corresponds to the probability of success, $1/C^2$ corresponds to the complexity, i.e. the order of the total shots required to accumulate successful eigenvalue inversion. Thus, the runtime with respect to κ is at least $\Omega(\kappa^2)$ (since it's the reciprocal), which is what we see in (20).

Thus constructing a matrix such that the condition number κ, the ratio of the magnitude of the biggest to the samllest eigenvalues, is small. Otherwise, a significant κ will severely worsen the runtime, and ruin the computational speedup.

s, the sparsness of a matrix, i.e. the maximum number of nonzero elements in any row or column, also looms large if uncontrolled, going quadratically as well. This further restricts the diversity of matrices reasonably solvable by HHL in the extreme cases. ϵ, the tolerated error must also be considered, and is associated with the size of the error tolerated from the true solution.

Problem & Methods

The proof of concept here involves two parts.

First, to implement HHL for a 3×3 invertible but non-hermitian A. Since A is non-hermitian, as shown in (4), we can convert it into a 6×6 hermitian matrix with the corresponding system described. Furthermore, the matrix must be of order 2^N, for integer N, so that the system solution can be represented as a product state of qubits. This can be done by just padding the matrix into an 8×8 matrix with 1's along the main diagonal and 0's everywhere else. We then proceed to implement the circuit described above. For the sake of testing the success of our implementation, we do sample to learn the amplitudes of the solution vector. We also make sure to choose a C that is a small margin from the smallest eigenvalue, as well as choose a matrix that is relatively sparse while still nonhermitian. The matrix is also scaled such that the eigenvalues are between 0 and 1. The implementation of QPE and state loading is standard. The conditional rotation is accomplished by using U gates controlled by the matching configuration of binary approximation qubits, and rotating by an angle

$$\arcsin\left(2^{n_l}\frac{C * 2\pi}{\tilde{\lambda}_k * T}\right), \tag{22}$$

where $\tilde{\lambda}_k$ is the binary aproximation output from QPE, as described previously. We used $n_l = 8$ qubits to approximate the expression for the eigenvalues output from QPE. For the quantum simulation for exponentiating the matrix, we used $T = 150$. Greater T values yield higher fidelity results by better approximating the constructed unitary from exponentiating the matrix A.

One tricky detail in implementing HHL for a non-hermitian matrix is that the newly constructed system yields negative eigenvalues. Negative eigenvalues result in complementary binary representations by QPE. Thus, negative binary representations had to be manually identified by viewing the real eigenvalues, and calculating from their complementary binary representations to find the appropriate negative rotation angle to the entangled control state, and thus corresponding eigenvector.

Greater implementation details and detailed circuit diagrams of HHL can be found in the attached notebook code below.

Second, we need to solve a data fitting problem with an $\mathbf{F} \in \mathbb{R}^{4x3}$ of rank 3 where $\mathbf{y} \in \mathbb{R}^4$ should not be a linear combination of columns of $\mathbf{F}$.

For further explanation of the methods applied to the data fitting problem, please see the appendix code section "Project 1.2: Data Fitting Problem".

Results

Application of HHL to a 3×3 invertible, non-hermitian matrix obtained an output solution vector that, after normalization of both actual and experimental vectors, obtained a normalized difference of $\boxed{0.049}$. We were able to get up to 70% success on eigenvalue inversion by careful tuning of the parameters described above.

For the data fitting problem, the application of HHL was able to achieve a normalied difference of $\boxed{0.09}$ between the theoretical $|\lambda\rangle$ and the experimental $|\lambda\rangle$.

Conclusion

In conclusion, HHL is able to successfully solve a system of linear equations for a 3×3 invertible, non-hermitian matrix.

While HHL holds promise, several restrictions include the consequences of negative eigenvalues that result from the necessary Hermitian expansion of non-hermitian matrices. As explained in the Problems & Methods section, these negative eigenvalue pairs introduced some issues in interpretation and results. Additionally, HHL stores the solution vector in a quantum-state. That is, a user wanting to inspect the solution would be unable to do so in faster than $O(n)$ time, ruining the time efficiency that HHL went through so much trouble to achieve.

Another drawback to the HHL algorithm comes during the step of conditional rotation. The control gates that define the conditional rotation need to be constructed depending on the output of QPE which detracts from the quantum-ness of the algorithm itself. These constructions require repeated measurement of the n_l registers, which provide the encoded eigenvalues that then a quantum engineer would need in order to manually construct in the system.

Similarly, in order to deal with the problem of negative/positive eigenvalue pairs, the *actual* eigenvalues were manually compared to those output through QPE when constructing the conditional rotation angles. A quantum algorithm that requires the use of classically determining eigenvalues seems a bit contradictory; however, for this proof-of-concept this method was sufficient. This process was tedious and likely took 30 minutes in total summation during the completion of this project. A point of future work would include a search algorithm to match the encoded eigenvalue pairs with their relative parity of actual eigenvalue.

Code

See https://github.com/tybens/quantum-data-fitting-HHL

References

https://journals.aps.org/prl/pdf/10.1103/PhysRevLett.103.150502
https://journals.aps.org/prl/pdf/10.1103/PhysRevLett.109.050505
https://qiskit.org/textbook/ch-applications/hhl_tutorial.htmlE.-Quantum-Phase-Estimation-(QPE)-within-HHL-

https://arxiv.org/pdf/1804.03719.pdf