Generative Algorithm for Quantum Phase Estimation

Griffin Shea

December 9, 2022

1. Introduction

The task of solving a linear system of equations is a frequent subroutine found in the algorithms used in almost every branch of pure and applied science and mathematics. The objective is to find the value of a vector $\vec{x} \in \mathbb{C}^N$, given the vector $\vec{b} \in \mathbb{C}^N$ and matrix $\mathrm{A} \in \mathbb{C}^{N\times N}$ such that the following equation holds true:

$$\mathrm{A}\vec{x} = \vec{b}$$

$$\begin{bmatrix} a_{0,0} & \cdots & a_{N-1,0} \\ \vdots & \ddots & \vdots \\ a_{0,N-1} & \cdots & a_{N-1,N-1} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} b_0 \\ \vdots \\ b_{N-1} \end{bmatrix}$$

This problem is heavily related to matrix inversion, as the equation above may also be written equivalently as:

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

One of the best known classical algorithms for matrix inversion (and solving a linear system of equations, by extension) is the conjugate gradient method, which has a runtime of $\mathrm{O}\left(s\sqrt{c}N\right)$ where A is s-sparse (i.e., s is the maximum number of nonzero values per row of A) and where c is the condition number (i.e. c is the ratio between the largest and smallest eigenvalues of A).

The HHL-algorithm (named after Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd and first described in [1]) is an algorithm designed to solve linear systems of equations on a quantum computer within a runtime of $\mathrm{O}(s^2c^2 \log N)$ (ignoring error rate). This algorithm therefore has a better-than-classical dependence on N but a worse-than-classical dependence on s and c [2]. Given A is 1-sparse with a condition number of one, it would then seem that the HHL-algorithm can produce the vector $\vec{x}$ in less time than it would take to write out that vector in $\mathrm{O}(N)$ time. However, HHL does not produce $\vec{x}$ in its classical representation, but rather the quantum state $|\vec{x}\rangle$:

$$|\vec{x}\rangle = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} \vdots \\ x_j \\ \vdots \end{bmatrix}$$

Nevertheless, the algorithm situationally (dependant on s and c) provides a significant increase in computational efficiency over all known classical methods and it may very well be a useful subroutine in quantum programs which require $|\vec{x}\rangle$ as an input to a subsequent function.

The HHL-algorithm relies heavily on another quantum algorithm called QPE (quantum phase estimation), which itself uses the inverse QFT (quantum Fourier transform), or QFT^{-1}, as a subroutine.

This report begins some of the work required to generate the gate configuration of the HHL-algorithm which would be required to program the algorithm on a quantum computer to solve arbitrarily sized linear systems of equations on the fly. Such a generator is necessary for applications where the size of a linear system is not set in stone until runtime. For example, physics simulations often require solving linear systems of equations as part of a quadratic programming problem where the size of N will depend on the number of resting contacts between rigid-body objects, which can vary frame-by-frame.

We focus on creating a generator function in MatLab using the QIT (quantum information toolkit) library to generate the QPE function for an arbitrary $n \times n$ unitary operator using m estimation qubits. But first, to verify correctness and QIT's implementation, we put together an algorithm to generate the encapsulated gate QFT^{-1} for 2 qubits, and then, for n qubits. The original goal was to develop the full HHL-algorithm for size N, but we did not accomplish this.

Section 2 will review the QPE algorithm in depth, and then section 3 analyzes our implementations. Section 4 is an evaluation of the project and section 5 concludes this paper.

2. Background Information

Quantum phase estimation (QPE) [4] estimates the phase θ_j of a given eigenvector $|\psi_j\rangle$ (with some rate of error) for a Hermitian matrix U where, for all eigenvalues λ_j: $|\lambda_j|^2 = 1$. The phase θ_j can then be fed into the function a function we will define as $\dot{\rho}(x) = e^{2\pi i x}$ to calculate the associated eigenvalue λ_j for $|\psi_j\rangle$:

$$U \cdot |\psi_j\rangle = \dot{\rho}(\theta_j) \cdot |\psi_j\rangle = \lambda_j \cdot |\psi_j\rangle$$

$$\dot{\rho}(\theta_j) = \lambda_j$$

One key property of this $\dot{\rho}(x)$ is that multiplication is additive, like so:

$$\dot{\rho}(x) \cdot \dot{\rho}(y) = \dot{\rho}(x + y)$$

The three main stages of the algorithm are marked **I**, **II**, and **III** in Figure 1:

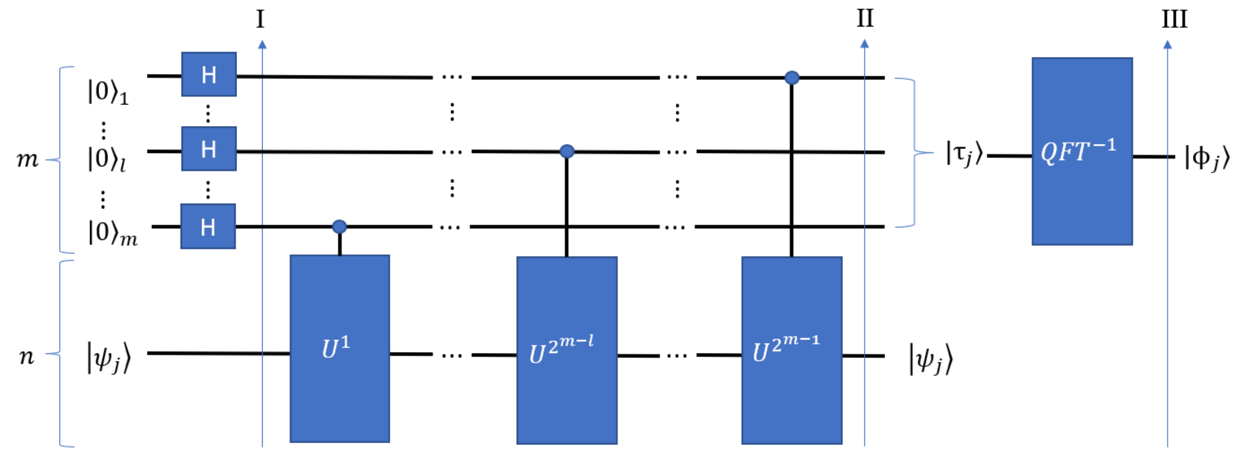


Figure 1: Quantum Phase Estimation Circuit. Described in section 2.

To set up QPE, one must first decide on a number of qubits m which will determine the precision of the phase estimation measurement. A qubit register of size m is set to $|0\rangle$ and then the Hadamard gate is applied to this register. A second qubit register of size n holds the eigenvector $|\psi_j\rangle$. The state of the complete register at stage $\mathbf{I}$ can therefore be represented as:

$$\frac{1}{2^{\frac{m}{2}}} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_m \otimes |\psi_j\rangle$$
$$= \frac{1}{2^{\frac{m-1}{2}}} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{m-1} \otimes \frac{1}{\sqrt{2}} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes |\psi_j\rangle$$
$$= \frac{1}{2^{\frac{m-1}{2}}} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{m-1} \otimes \frac{|0\rangle \otimes |\psi_j\rangle + |1\rangle \otimes |\psi_j\rangle}{\sqrt{2}}$$

Next, a series of m exponentiated U gate operations are applied to $|\psi_j\rangle$, each controlled by one of the qubits in the first register. In general, for each control qubit l, a $U^{2^{m-l}}$ operation is applied to $|\psi_j\rangle$ if the control qubit l is in the $|1\rangle$ state and no operation is applied if that qubit is in the $|0\rangle$ state.

When the first controlled-U gate operation is applied to $|\psi_j\rangle$, the final qubit in the first register labelled m acts as the control qubit; Set $l = m$. The operation $U^{2^{m-m}} = U^1$ is applied to $|\psi_j\rangle$ only if qubit m is in the $|1\rangle$ state. From the equations above, we know $U \cdot |\psi_j\rangle = \dot{\rho}(\theta_j) \cdot |\psi_j\rangle$. So the result of the controlled-U gate operation will be the following quantum state:

$$\frac{1}{2^{\frac{m-1}{2}}} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{m-1} \otimes \frac{|0\rangle \otimes |\psi_j\rangle + |1\rangle \otimes U^1 \cdot |\psi_j\rangle}{\sqrt{2}}$$

$$= \frac{1}{2^{\frac{m-1}{2}}} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{m-1} \otimes \frac{|0\rangle \otimes |\psi_j\rangle + |1\rangle \otimes \dot{\rho}(\theta_j) \cdot |\psi_j\rangle}{\sqrt{2}}$$

$$= \frac{1}{2^{\frac{m-1}{2}}} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{m-1} \otimes \frac{|0\rangle \otimes |\psi_j\rangle + \dot{\rho}(\theta_j) \cdot |1\rangle \otimes |\psi_j\rangle}{\sqrt{2}}$$

$$= \frac{1}{2^{\frac{m-1}{2}}} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{m-1} \otimes \frac{|0\rangle + \dot{\rho}(\theta_j) \cdot |1\rangle}{\sqrt{2}} \otimes |\psi_j\rangle$$

$$= \frac{1}{2^{\frac{m-1}{2}}} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{m-1} \otimes \frac{1}{\sqrt{2}} \cdot \begin{bmatrix} 1 \\ \dot{\rho}(\theta_j) \end{bmatrix} \otimes |\psi_j\rangle$$

$$= \frac{1}{2^{\frac{m-2}{2}}} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{m-2} \otimes \frac{1}{\sqrt{2}} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \cdot \begin{bmatrix} 1 \\ \dot{\rho}(\theta_j) \end{bmatrix} \otimes |\psi_j\rangle$$

$$= \frac{1}{2^{\frac{m-2}{2}}} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{m-2} \otimes \frac{1}{2} \cdot \begin{bmatrix} 1 \\ \dot{\rho}(\theta_j) \\ 1 \\ \dot{\rho}(\theta_j) \end{bmatrix} \otimes |\psi_j\rangle$$

Notice that $|\psi_j\rangle$ remains isolated; The m sized qubit register accumulates the phase of the eigenvector. Next, set $l = m - 1$. The operation $U^{2^{m-(m-1)}} = U^2$ is applied to $|\psi_j\rangle$ only when qubit $m-1$ is in the $|1\rangle$ state. Also, $U^2 \cdot |\psi_j\rangle = \dot{\rho}(2\theta_j) \cdot |\psi_j\rangle$. The register's state after applying the controlled-U gate:

$$\frac{1}{2^{\frac{m-2}{2}}} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{m-2} \otimes \frac{1}{2} \cdot \begin{bmatrix} 1 \\ \dot{\rho}(\theta_j) \\ \dot{\rho}(2\theta_j) \\ \dot{\rho}(2\theta_j) \cdot \dot{\rho}(\theta_j) \end{bmatrix} \otimes |\psi_j\rangle$$

$$= \frac{1}{2^{\frac{m-2}{2}}} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{m-2} \otimes \frac{1}{2} \cdot \begin{bmatrix} 1 \\ \dot{\rho}(\theta_j) \\ \dot{\rho}(2\theta_j) \\ \dot{\rho}(3\theta_j) \end{bmatrix} \otimes |\psi_j\rangle$$

$|\psi_j\rangle$ remains constant as the algorithm carries on until stage **II** when we are left with the quantum state

$$|\tau_j\rangle \otimes |\psi_j\rangle$$

where $|\tau_j\rangle$ is defined such that:

$$|\tau_j\rangle = \frac{1}{2^{\frac{m}{2}}} \cdot \sum_{k=0}^{2^m-1} \dot{\rho}(k \cdot \theta_j) \cdot |k\rangle = \frac{1}{2^{\frac{m}{2}}} \cdot \begin{bmatrix} \vdots \\ \dot{\rho}(k \cdot \theta_j) \\ \vdots \end{bmatrix}_m$$

$$\left|\tau_j\right\rangle = \begin{bmatrix} \vdots \\ \tau_{j,k} \\ \vdots \end{bmatrix}_m \quad ; \quad \tau_{j,k} = \frac{1}{2^{\frac{m}{2}}} \cdot \dot{\rho}\left(k \cdot \theta_j\right)$$

This becomes the input for the inverse quantum Fourier transform (QFT^{-1}) which will generate the final quantum state, which we call $\left|\phi_j\right\rangle$, at stage **III**. In general, QFT^{-1} is defined as such (see [3]):

$$\mathrm{QF}T^{-1}\left(\begin{bmatrix} \vdots \\ \mathrm{x}_k \\ \vdots \end{bmatrix}_m\right) = \begin{bmatrix} \vdots \\ y_k \\ \vdots \end{bmatrix}_m = \frac{1}{2^{\frac{m}{2}}} \cdot \begin{bmatrix} \vdots \\ \sum_{l=0}^{2^m-1} x_l \cdot \dot{\rho}\left(\frac{-l \cdot k}{2^m}\right) \\ \vdots \end{bmatrix}_{\mathrm{m}}$$

Inputting $\left|\tau_j\right\rangle$ gives produces:

$$\mathrm{QF}T^{-1}\left(\left|\tau_j\right\rangle\right) = \left|\phi_j\right\rangle = \begin{bmatrix} \vdots \\ \phi_{j,k} \\ \vdots \end{bmatrix}_m$$

$$\begin{bmatrix} \vdots \\ \phi_{j,k} \\ \vdots \end{bmatrix}_m = \frac{1}{2^{\frac{m}{2}}} \cdot \begin{bmatrix} \vdots \\ \sum_{l=0}^{2^m-1} \tau_{j,l} \cdot \dot{\rho}\left(\frac{-l \cdot k}{2^m}\right) \\ \vdots \end{bmatrix}_{\mathrm{m}}$$

$$\begin{bmatrix} \vdots \\ \phi_{j,k} \\ \vdots \end{bmatrix}_m = \frac{1}{2^{\frac{m}{2}}} \cdot \begin{bmatrix} \vdots \\ \sum_{l=0}^{2^m-1} \frac{1}{2^{\frac{m}{2}}} \cdot \dot{\rho}\left(l \cdot \theta_j\right) \cdot \dot{\rho}\left(\frac{-l \cdot k}{2^m}\right) \\ \vdots \end{bmatrix}_{\mathrm{m}}$$

$$\begin{bmatrix} \vdots \\ \phi_{j,k} \\ \vdots \end{bmatrix}_m = \frac{1}{2^{\frac{m}{2}}} \cdot \frac{1}{2^{\frac{m}{2}}} \cdot \begin{bmatrix} \vdots \\ \sum_{l=0}^{2^m-1} \dot{\rho}\left(l \cdot \theta_j\right) \cdot \dot{\rho}\left(\frac{-l \cdot k}{2^m}\right) \\ \vdots \end{bmatrix}_{\mathrm{m}}$$

$$\begin{bmatrix} \vdots \\ \phi_{j,k} \\ \vdots \end{bmatrix}_m = \frac{1}{2^m} \cdot \begin{bmatrix} \vdots \\ \sum_{l=0}^{2^m-1} \dot{\rho}\left(l \cdot \theta_j\right) \cdot \dot{\rho}\left(\frac{-l \cdot k}{2^m}\right) \\ \vdots \end{bmatrix}_{\mathrm{m}}$$

$$\begin{bmatrix} \vdots \\ \phi_{j,k} \\ \vdots \end{bmatrix}_m = \frac{1}{2^m} \cdot \begin{bmatrix} \vdots \\ \sum_{l=0}^{2^m-1} \dot{\rho}\left(l \cdot \theta_j - \frac{l \cdot k}{2^m}\right) \\ \vdots \end{bmatrix}_{\mathrm{m}}$$

(A more in-depth look at $\mathrm{QF}T^{-1}$ is provided in sections **3.1** and **3.2**.) The final step of QPE is to make a measurement on $|\phi_j\rangle$ (), collapsing the wave function into a state $\mu = (0, \ldots 2^m)$ in a standard base-2 classical bit representation. The phase θ_j of the given eigenvector $|\psi_j\rangle$ can then be estimated as $\tilde{\theta}_j = \frac{\mu}{2^m}$ and then the eigenvalue can be estimated as $\tilde{\lambda}_j = \dot{\rho}(\tilde{\theta}_j) = e^{2\pi i \tilde{\theta}_j}$. If θ_j is a value that falls into $\left(\frac{0}{2^m}, \frac{1}{2^m}, \ldots \frac{m-1}{2^m}\right)$, the precision and accuracy of the measurement are absolute: $\tilde{\theta}_j = \theta_j$ one hundred per-cent of the time. Otherwise, the measurement μ is likely, though not certain, to be measured as the closest m-bit integer to the true value of $\theta_j \cdot 2^m$.

3. Results

3.1 Inverse Quantum Fourier Transform for Two Qubits

We began by producing an algorithm to simulate $\mathrm{QF}T_n^{-1}$ where $n = 2$, i.e., for two qubits, using the quantum information toolkit (QIT) library in MatLab. Figure 2 shows a diagram of the $\mathrm{QF}T_2^{-1}$ quantum circuit:

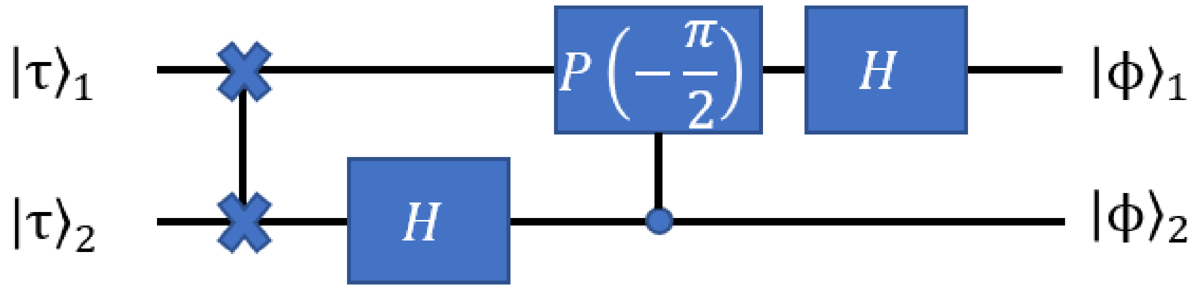


Figure 2: Quantum Fourier transform for two qubits. Described in section 3.1.

First, $\mathrm{QF}T_n^{-1}$ must use swapping gates to reverse the order of the input qubits, we only need to use one SWAP gate here. Then, the Hadamard gate is applied to the second qubit, which then controls the phase shift gate $\mathrm{P}\left(-\frac{\pi}{2}\right)$ applied to the first qubit. The phase shift gate is defined:

$$P(\varphi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix}$$

Finally, Hadamard is applied to the first qubit. Such a gate is very simple to implement in QIT and we have done so in the file invQFT2.m. Our implementation simulates the algorithm by multiplying each subsequent gate together in order to produce a gate representing the entire procedure (each program presented in this paper is created this way), and then this gate, in matrix representation, can be applied to a qubit register to produce a result. The one-gate representations of $\mathrm{QF}T_n^{-1}$ and QFT_2^{-1} are (from [3]):

$$\mathrm{QFT}_n^{-1} = \begin{bmatrix} q_{0,0} & \vdots & q_{n-1,0} \\ \cdots & \ddots & \cdots \\ q_{0,n-1} & \vdots & q_{n-1,n-1} \end{bmatrix} = \begin{bmatrix} \ddots & \vdots & \iddots \\ \cdots & q_{k,l} & \cdots \\ \iddots & \vdots & \ddots \end{bmatrix} = \frac{1}{2^{\frac{n}{2}}} \cdot \begin{bmatrix} \ddots & \vdots & \iddots \\ \cdots & \dot{\rho}\left(\frac{-l \cdot k}{2^n}\right) & \cdots \\ \iddots & \vdots & \ddots \end{bmatrix}$$

$$\mathrm{QFT}_2^{-1} = \frac{1}{2} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$

3.2 Inverse Quantum Fourier Transform for n Qubits

Next, we take advantage of the doubly recursive properties of QFT_n^{-1} to write invQFTN.m. Figure 4 illustrate these recursive properties:

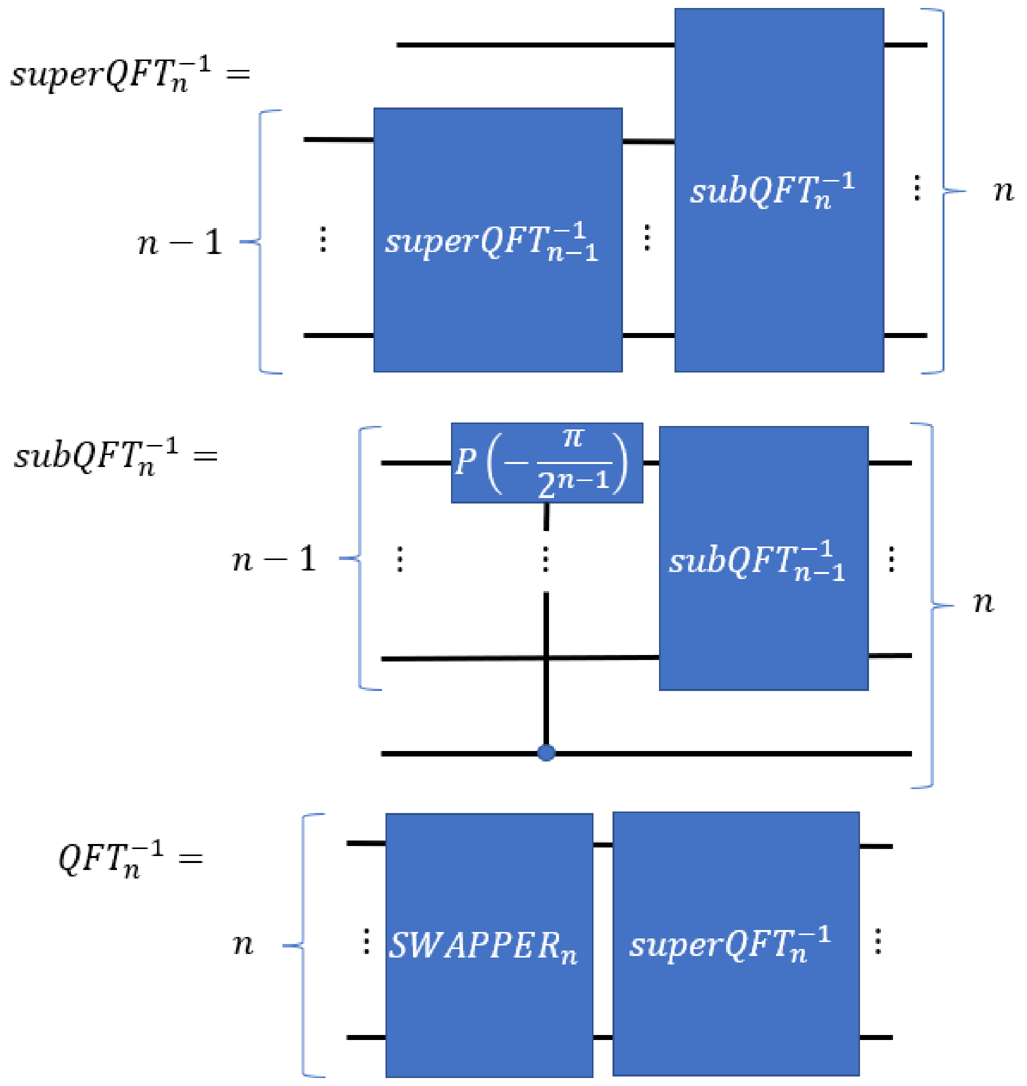


Figure 3: Inverse quantum Fourier transform defined recursively. Described in section 3.2.

Here, $super\mathrm{QFT}_n^{-1} = \mathrm{I} \otimes superQ\mathrm{FT}_{n-1}^{-1} \cdot \mathrm{subQFT}_n^{-1}$ and $\mathrm{subQFT}_n^{-1} = P^* \cdot \mathrm{subQFT}_{n-1}^{-1} \otimes \mathrm{I}$, where P^* is a phase shift gate $P\left(-\frac{\pi}{2^{n-1}}\right)$ controlled by the n^{th} qubit, which is the bottom qubit in the register. The base case of both of these recursive gates $super\mathrm{QFT}_1^{-1}$ and subQFT_1^{-1} are simply the Hadamard gate. Finally, QFT_n^{-1} itself is given as a $SWAPPER_n$ gate followed by $super\mathrm{QFT}_n^{-1}$. $SWAPPER_n$ simply reverses the order of n qubits such that:

$$\mathrm{SWAPPE}R_n(|\psi\rangle_1, |\psi\rangle_2, \dots |\psi\rangle_n) = |\psi\rangle_n, |\psi\rangle_{n-1}, \dots |\psi\rangle_1$$

We implement this gate by "filling" n columns of the circuit with SWAP gates that do not cross any lines. By this we mean, the first column contains $\left\lfloor\frac{n}{2}\right\rfloor$ SWAP gates connecting the pairs of qubits (1,2), (3,4) (5,6), and so on (or rather, until n or $n-1$ appears in the sequence). The second column then contains $\left\lfloor\frac{n-1}{2}\right\rfloor$ SWAP gates connecting the pairs of qubits (2,3), (4,5), (6,7) and so on (or again until n or $n-1$ appears in the sequence). All subsequent odd columns copy the first, and all subsequent even columns copy the second for $n-2$ additional columns (naturally, the method is defined only for values of $n > 2$). This ensures that all qubits are re-ordered in precisely n columns.

3.3 Quantum Phase Estimation for $\boldsymbol{n}, \boldsymbol{m}$ Qubits and $\boldsymbol{U}$

The program QPENM.m generates a gate representing QPE with m estimation qubits and a unitary operator U of size n. It follows the procedure set out in section **2** and was not difficult to implement once invQFTN.m was out of the way. First we create a large gate $H^{\otimes m} \otimes I^{\otimes n}$ to accumulate later operations. We iterate from 1 to m and add on (through matrix multiplication) each controlled-U, exponentiated accordingly. The last step is to add on $QFT_n^{-1} \otimes I^{\otimes n}$.

4 Evaluation

In this section we evaluate the work produced for this project. Specifically, we will demonstrate the correctness of our quantum algorithm implementations.

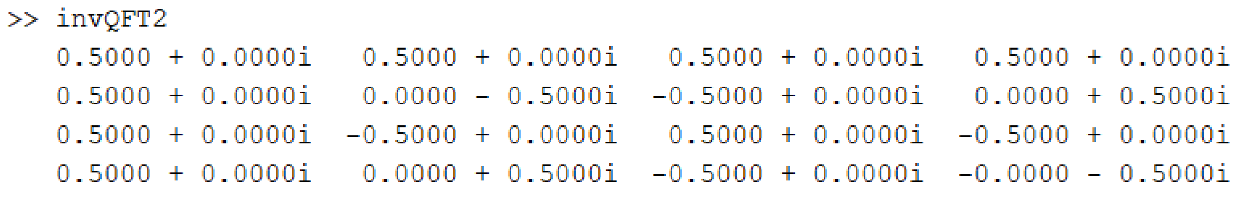
4.1 Validating invQFT2.m

Recall the one-gate definition for the QFT_n^{-1} and QFT_2^{-1}:

$$QFT_n^{-1} = \begin{bmatrix} q_{0,0} & \vdots & q_{n-1,0} \\ \cdots & \ddots & \cdots \\ q_{0,n-1} & \vdots & q_{n-1,n-1} \end{bmatrix} = \begin{bmatrix} \ddots & \vdots & \iddots \\ \cdots & q_{k,l} & \cdots \\ \iddots & \vdots & \ddots \end{bmatrix} = \frac{1}{2^{\frac{n}{2}}} \cdot \begin{bmatrix} \ddots & \vdots & \iddots \\ \cdots & \dot{\rho}\left(\frac{-l \cdot k}{2^n}\right) & \cdots \\ \iddots & \vdots & \ddots \end{bmatrix}$$

$$QFT_2^{-1} = \frac{1}{2} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$

As we said before, invQFT2.m and our other programs implement the algorithms we describe as one encapsulated gate operation. To show the correctness of this program, we produce figure 3, showing the console output when we print the matrix in MatLab. Comparing this to QFT_2^{-1}, we see the very same values, so the implementation is correct.

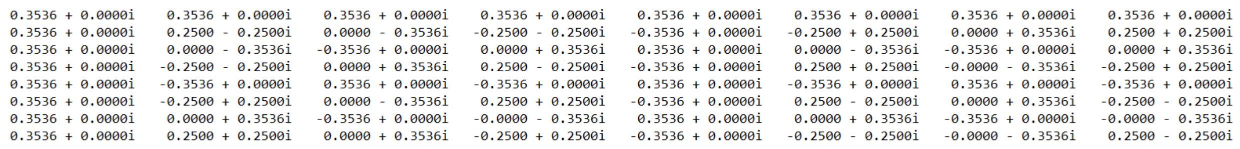


```
>> invQFT2
   0.5000 + 0.0000i   0.5000 + 0.0000i   0.5000 + 0.0000i   0.5000 + 0.0000i
   0.5000 + 0.0000i   0.0000 - 0.5000i  -0.5000 + 0.0000i   0.0000 + 0.5000i
   0.5000 + 0.0000i  -0.5000 + 0.0000i   0.5000 + 0.0000i  -0.5000 + 0.0000i
   0.5000 + 0.0000i   0.0000 + 0.5000i  -0.5000 + 0.0000i  -0.0000 - 0.5000i
```

Figure 4: Output matrix of a call to invQFT2.m. Referenced in section 4.1.

4.2 Validating invQFTN.m

We demonstrate the correctness of invQFTN.m is validated by calling the function with the argument 3, generating the matrix in figure 5:



```
 0.3536 + 0.0000i   0.3536 + 0.0000i   0.3536 + 0.0000i   0.3536 + 0.0000i   0.3536 + 0.0000i   0.3536 + 0.0000i   0.3536 + 0.0000i   0.3536 + 0.0000i
 0.3536 + 0.0000i   0.2500 - 0.2500i   0.0000 - 0.3536i  -0.2500 - 0.2500i  -0.3536 + 0.0000i  -0.2500 + 0.2500i   0.0000 + 0.3536i   0.2500 + 0.2500i
 0.3536 + 0.0000i   0.0000 - 0.3536i  -0.3536 + 0.0000i   0.0000 + 0.3536i   0.3536 + 0.0000i   0.0000 - 0.3536i  -0.3536 + 0.0000i   0.0000 + 0.3536i
 0.3536 + 0.0000i  -0.2500 - 0.2500i   0.0000 + 0.3536i   0.2500 - 0.2500i  -0.3536 + 0.0000i   0.2500 + 0.2500i  -0.0000 - 0.3536i  -0.2500 + 0.2500i
 0.3536 + 0.0000i  -0.3536 + 0.0000i   0.3536 + 0.0000i  -0.3536 + 0.0000i   0.3536 + 0.0000i  -0.3536 + 0.0000i   0.3536 + 0.0000i  -0.3536 + 0.0000i
 0.3536 + 0.0000i  -0.2500 + 0.2500i   0.0000 - 0.3536i   0.2500 + 0.2500i  -0.3536 + 0.0000i   0.2500 - 0.2500i   0.0000 + 0.3536i  -0.2500 - 0.2500i
 0.3536 + 0.0000i   0.0000 + 0.3536i  -0.3536 + 0.0000i  -0.0000 - 0.3536i   0.3536 + 0.0000i   0.0000 + 0.3536i  -0.3536 + 0.0000i  -0.0000 - 0.3536i
 0.3536 + 0.0000i   0.2500 + 0.2500i   0.0000 + 0.3536i  -0.2500 + 0.2500i  -0.3536 + 0.0000i  -0.2500 - 0.2500i  -0.0000 - 0.3536i   0.2500 - 0.2500i
```

Figure 5: Output matrix of a call to invQFTN.m passing 3 as the argument. The output was reformatted but the values are undisturbed. Referenced in section 4.2.

This matches the gate given by QFT_3^{-1} using the QFT_n^{-1} definition from above:

$$QFT_3^{-1} = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \frac{1}{\sqrt{2}}(1-i) & -i & \frac{1}{\sqrt{2}}(-1-i) & -1 & \frac{1}{\sqrt{2}}(-1+i) & i & \frac{1}{\sqrt{2}}(1+i) \\ 1 & -i & -1 & i & 1 & -i & -1 & i \\ 1 & \frac{1}{\sqrt{2}}(-1-i) & i & \frac{1}{\sqrt{2}}(1-i) & -1 & \frac{1}{\sqrt{2}}(1+i) & -i & \frac{1}{\sqrt{2}}(-1+i) \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & \frac{1}{\sqrt{2}}(-1+i) & -i & \frac{1}{\sqrt{2}}(1+i) & -1 & \frac{1}{\sqrt{2}}(1-i) & i & \frac{1}{\sqrt{2}}(-1-i) \\ 1 & i & -1 & -i & 1 & i & -1 & -i \\ 1 & \frac{1}{\sqrt{2}}(1+i) & i & \frac{1}{\sqrt{2}}(-1+i) & -1 & \frac{1}{\sqrt{2}}(-1-i) & -i & \frac{1}{\sqrt{2}}(1-i) \end{bmatrix}$$

4.3 Validating QPENM.m

To validate the correctness of QPENM.m, we run an experiment in main.m included with the project files. QPENM.m is used to generate a QPE gate with 2 estimation qubits and a matrix U defined like so:

$$\mathrm{U} = \begin{bmatrix} \dot{\rho}(0) & 0 & 0 & 0 \\ 0 & \dot{\rho}\left(\frac{1}{2}\right) & 0 & 0 \\ 0 & 0 & \dot{\rho}\left(\frac{1}{4}\right) & 0 \\ 0 & 0 & 0 & \dot{\rho}\left(\frac{1}{8}\right) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & \mathrm{i} & 0 \\ 0 & 0 & 0 & \frac{(1+\mathrm{i})}{\sqrt{2}} \end{bmatrix}$$

A quick analysis will show that the values lying along the diagonal of this matrix are the eigenvalues and their associated eigenvectors are the simply the basis vectors:

$$(\lambda_0, \lambda_1, \lambda_2, \lambda_3) = \left(\dot{\rho}(0), \dot{\rho}\left(\frac{1}{2}\right), \dot{\rho}\left(\frac{1}{4}\right), \dot{\rho}\left(\frac{1}{8}\right)\right)$$

$$|\psi_0\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, |\psi_1\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, |\psi_2\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, |\psi_3\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Also, the real values for θ_j are $(\theta_0, \theta_1, \theta_2, \theta_3) = \left(0, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}\right)$. We expect the quantum states generated by our QPE gate will match the following values which are calculable using the equations provided in this paper:

$$|\phi_0\rangle = |00\rangle$$

$$|\phi_1\rangle = |10\rangle$$

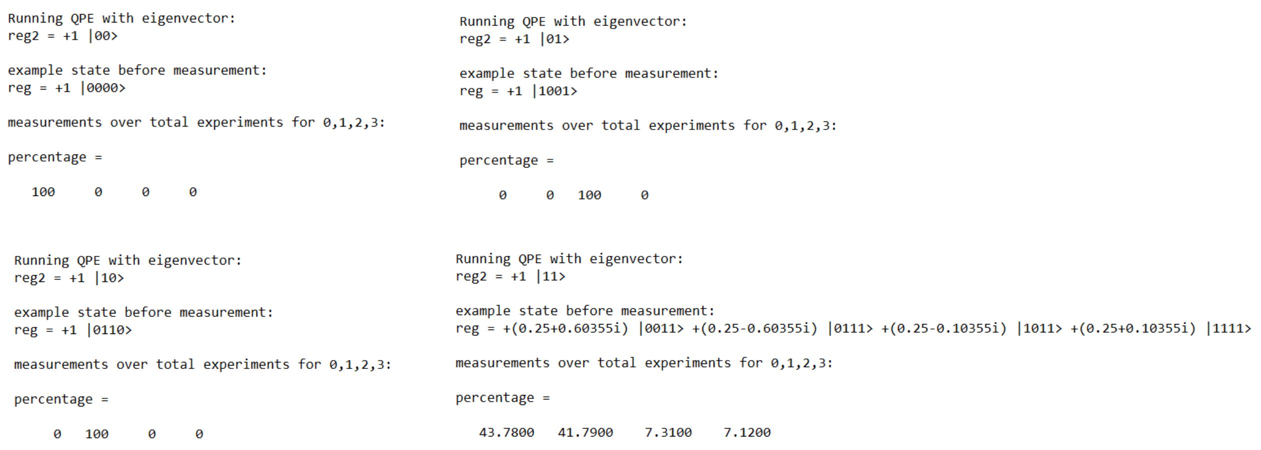
$$|\phi_2\rangle = |01\rangle$$

$$|\phi_3\rangle = \frac{1}{4} \cdot \begin{bmatrix} (1+\sqrt{2})i + 1 \\ -(1+\sqrt{2})i + 1 \\ (1-\sqrt{2})i + 1 \\ -(1-\sqrt{2})i + 1 \end{bmatrix} = \begin{bmatrix} 0.25 + 0.60\mathrm{i} \\ 0.25 - 0.60\mathrm{i} \\ 0.25 - 0.10\mathrm{i} \\ 0.25 + 0.10\mathrm{i} \end{bmatrix}$$

The outlier here is $|\phi_3\rangle$ which is the only state in a superposition. We can calculate the probability of each outcome $k \in (0,1,2,3)$ using the measurement operator $\langle \phi_3 | M_{k,k} | \phi_3 \rangle$.

$$\langle \phi_3 | M_{0,0} | \phi_3 \rangle = \langle \phi_3 | M_{1,1} | \phi_3 \rangle = 42.68\%$$
$$\langle \phi_3 | M_{2,2} | \phi_3 \rangle = \langle \phi_3 | M_{3,3} | \phi_3 \rangle = 7.32\%$$

Figure 6 shows the output of our little experiment run in main.m. The eigenvectors $|\psi_0\rangle$ through $|\psi_2\rangle$ indeed produce the predicted pure quantum states. In these three cases, θ_j can be estimated with zero error: $\tilde{\theta}_1 = 0$, $\tilde{\theta}_2 = \frac{1}{2}$, and $\tilde{\theta}_2 = \frac{1}{4}$. With regard to $|\psi_3\rangle$, QIT's internal representation of the superposition also matches our calculation. Moreover, the measurement distribution also aligns with our calculated probabilities. In estimating θ_3, we are likely to calculate either $\tilde{\theta}_3 = 0$ or $\tilde{\theta}_3 = \frac{1}{4}$ with equal probability. Obviously, there is an significant error of $\frac{1}{8}$ difference between the actual phase and either estimation, but this is the result of using only two estimation qubits. On the basis that the output of the experiment matched the calculations, we claim our implementation correctly models the QPE algorithm.



```
Running QPE with eigenvector:
reg2 = +1 |00>

example state before measurement:
reg = +1 |0000>

measurements over total experiments for 0,1,2,3:

percentage =

   100     0     0     0
```

```
Running QPE with eigenvector:
reg2 = +1 |01>

example state before measurement:
reg = +1 |1001>

measurements over total experiments for 0,1,2,3:

percentage =

     0     0   100     0
```

```
Running QPE with eigenvector:
reg2 = +1 |10>

example state before measurement:
reg = +1 |0110>

measurements over total experiments for 0,1,2,3:

percentage =

     0   100     0     0
```

```
Running QPE with eigenvector:
reg2 = +1 |11>

example state before measurement:
reg = +(0.25+0.60355i) |0011> +(0.25-0.60355i) |0111> +(0.25-0.10355i) |1011> +(0.25+0.10355i) |1111>

measurements over total experiments for 0,1,2,3:

percentage =

   43.7800   41.7900    7.3100    7.1200
```

Figure 6: Quantum phase estimation demonstrative experiment. Described in section 4.3.

4.4 SWAPPER Gate Implementation Efficiency

Our implementation of the $\mathrm{SWAPPER}_n$ gate, which can reorder n qubits in precisely n steps, is an improvement on implementations which rely on "long SWAPs" that swap the pairs (1, n), (2, $n-1$), (3, $n-2$), and so on individually. These long SWAPs swap the position of a qubit pair (i,j) by applying $i-j-1$ SWAP gates to move qubit i down to the j position and then applying $i-j-2$ more SWAP gates to move what was qubit j (but is now in position $j-1$) back up to the i position. This takes a total of $2i-2j-3$ gates, or $2n-3, if\ we\ set\ n = i - j$, and only to change the positions of the i^{th} and j^{th} qubits (and not any qubit in between). For the purpose of a complete re-ordering of n qubits, our $\mathrm{SWAPPER}_n$ provides a significant improvement. On the other hand, this $\mathrm{SWAPPER}_n$ gate can often be avoided in practice by simply flipping QFT^{-1} and all subsequent gates, rather than reordering the qubits themselves.[1]

[1] Furthermore, knowing that the SWAP gate decomposes into three CNOT gates, there could be an even better solution. [5] displays a method for swapping the 1st and 10th qubits with 19 CNOT gates, which is an improvement over the 33 CNOT gates needed by the "long swap" method.

5. Conclusion

In this project, we focussed on building algorithms to generate a one-gate representation to simulate the inverse quantum Fourier transform for some size m qubits as well as quantum phase estimation for a given number of m estimation qubits and a given unitary operator U of size n. We feel that being able to generate these quantum algorithms for an arbitrary size on the fly would be a valuable capability moving forward. While we did not manage to achieve our original goal of simulating the HHL-algorithm for solving linear equations, we believe the work we did was relevant to the course material and quite engaging. In this paper, we are most proud of the decomposition of the inverse Fourier transform into its recursive definition and our SWAPPER gate implementation.

References

[1] A. W. Harrow, A. Hassidim, and S. Lloyd, “Quantum Algorithm for Linear Systems of Equations,” *Physical Review Letters*, vol. 103, no. 15, Oct. 2009, doi: 10.1103/physrevlett.103.150502.

[2] A. Ambainis, *Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations*. arXiv, 2010. doi: 10.48550/ARXIV.1010.4458.

[3] M. Barbeau. Hands-on Quantum Communications and Networking. September 4, 2022 edition. Unpublished.

[4] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, “Quantum algorithms revisited,” *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1969, pp. 339–354, Jan. 1998, doi: 10.1098/rspa.1998.0164.

[5] C. Gidney. (2017, August. 22). Breaking Down the Quantum Swap [Online]. Available: https://algassert.com/post/1717