The Quantum Fourier Transform and Applications

Palash Pandya
201002162
IIIT-Hyderabad
# Contents

1 Classical Fourier Transform ............................................. 2
   1.1 Discrete Fourier Transform (DFT) .................................. 2
      1.1.1 Introduction .................................................. 2
      1.1.2 FFT ............................................................. 3
      1.1.3 Analysis of FFT ................................................. 3

2 Quantum Fourier Transform ............................................... 4
   2.1 Quantum Fourier Transform ......................................... 4
      2.1.1 Introduction .................................................. 4
      2.1.2 Unitarity of QFT .............................................. 5
      2.1.3 Product Representation of QFT ................................. 6
   2.2 Efficient Circuit for QFT ........................................... 8
      2.2.1 The circuit ..................................................... 8
      2.2.2 Working of the Circuit ....................................... 8
      2.2.3 Analysis of the Circuit ...................................... 9

3 Applications of Quantum Fourier Transform .......................... 10
   3.1 Applications of QFT ................................................ 10
      3.1.1 Iterative Phase Estimation Algorithm ........................ 10
Chapter 1

Classical Fourier Transform

1.1 Discrete Fourier Transform (DFT)

1.1.1 Introduction

Consider a vector \( f \) of \( N \) complex numbers, \( f_k, k \in \{0, 1, \ldots, N - 1\} \). Then the DFT is a map from these \( N \) complex numbers to \( N \) complex numbers. The fourier transformed coefficients, denoted by, \( \tilde{f}_j \), are:

\[
\tilde{f}_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{-jk} f_k
\]

where \( \omega = exp\left(\frac{2\pi i}{N}\right) \). Then the inverse DFT is given by:

\[
f_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{jk} \tilde{f}_k
\]

For each component of the new vector, we do \( N \) multiplications, and then add them, so we have \( N^2 \) complex multiplications and \( N (N - 1) \) complex additions in total to compute the DFT. This is \( O(N^2) \). While Danielson and Lanczos, in 1942, proposed a lemma, which computes the DFT in \( O(N \log N) \) time, known as Fast Fourier Transform (FFT). FFT is very widely used and has many applications in many areas.
1.1.2 FFT

Consider the coefficients in DFT, \( \omega^k ( = e^{2\pi ik/N} ) \), we find that:

\[
\omega^{k+N/2} = -\omega^k \quad \text{and} \quad \omega^{k+N} = \omega^k
\]

Using this symmetry of coefficients, we can split the components of vector \( f \) into smaller, \( N/2 \) sized vectors, which have even and odd coefficients, \( e \) and \( o \) respectively, as (with \( N = 2^p \)):

\[
\tilde{f}_j = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} \omega^{-ij} f_i
\]

\[
= \frac{1}{\sqrt{N}} \left( \sum_{i=0}^{N/2-1} \omega^{-2ij} e_i + \sum_{i=0}^{N/2-1} \omega^{-(2i+1)j} o_i \right)
\]

\[
= \frac{1}{\sqrt{N}} \left( \sum_{i=0}^{N/2-1} \omega^{-ij} e_i + \omega_N^{-j} \sum_{i=0}^{N/2-1} \omega^{-ij} o_i \right)
\]

where \( \omega_{N/2} = e^{\frac{2\pi i}{N} \frac{N}{2}} \), \( \omega_N = e^{\frac{2\pi i}{N}} \).

1.1.3 Analysis of FFT

The previous equation describes the DFT of \( f \) as the sum of DFTs of \( e \) and \( o \):

\[
\tilde{f}_j = \tilde{e}_j + \omega_N^{-j} \tilde{o}_j
\]

Further, both \( e \) and \( o \) can be continually splitted into two, till \( N = 2 \). To compute \( \tilde{e} \) and \( \tilde{o} \) we need \( 2 \frac{N^2}{2} = \frac{N^2}{2} \) multiplications. And then to compute \( \omega_N^{-j} \tilde{o}_j \), another \( N/2 \) multiplications.

So in total, \( \left( \frac{N^2}{2} + \frac{N}{2} \right) \) multiplications, which is less by a factor of 2 for large \( N \), compared to \( N^2 \) for DFT. If \( T_N \) is the number of complex multiplications to be done for \( N = 2^n \), then \( T_n = 2T_{n-1} + 2^{n-1} \), which is bounded by \( T_n \leq 2T_{n-1} + 2^n \) and has solution \( T_n \leq 2^n n \).

Which means that running time is bounded by \( N \log N \), an improvement compared to DFT.
Chapter 2

Quantum Fourier Transform

2.1 Quantum Fourier Transform

2.1.1 Introduction

In the Quantum Fourier Transform (QFT), we do a DFT on the amplitudes of a quantum state. Consider the orthonormal basis set $|j\rangle$, where $|j\rangle$ is $|0\rangle, |1\rangle, |2\rangle, \ldots, |N-1\rangle$.

Then, QFT would have the following effect on the basis state $|j\rangle$:

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i j k}{N}} |k\rangle$$

And Action on an arbitrary state:

$$\sum_{j} \alpha_j |j\rangle \rightarrow \sum_{k} \tilde{\alpha}_k |k\rangle,$$

where $\tilde{\alpha}_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{\frac{2\pi i j k}{N}} \alpha_j$.

And amplitudes $\tilde{\alpha}_k$ are the DFT of the amplitudes $\alpha_j$. 

4
2.1.2 Unitarity of QFT

QFT can only be implemented physically if it is a unitary transformation. If it were denoted by the operator \( \hat{F} \). Then,

\[ |\tilde{\psi}\rangle = \hat{F} |\psi\rangle , \quad \hat{F}^\dagger \hat{F} = \hat{I} \]

where \( |\psi\rangle \) is any arbitrary state with \( |j\rangle \) as basis. Then we can write \( \hat{F} \) in outer product notation as:

\[ \hat{F} = \frac{1}{\sqrt{N}} \sum_{j,k=0}^{N-1} e^{2\pi i jk/N} |k\rangle \langle j | \]

It is easy to verify that this has the required effect, i.e.

\[ |\psi\rangle \rightarrow |\tilde{\psi}\rangle \]

Proof of Unitarity:

\[ \hat{F} = \left( \frac{1}{\sqrt{N}} \sum_{j,k} e^{-2\pi ijk/N} |j\rangle \langle k | \right) , \quad \hat{F}^\dagger = \left( \frac{1}{\sqrt{N}} \sum_{j',k'} e^{-2\pi ij'k'/N} |j'\rangle \langle k' | \right) \]

\[ \hat{F}^\dagger \hat{F} = \frac{1}{N} \sum_{j,k,j',k'} e^{2\pi i (jk-j'k')/N} |j'\rangle \langle j | \delta_{kk'} \]
\[ = \frac{1}{N} \sum_{j,k,j'} e^{2\pi i (j-j')k/N} |j'\rangle \langle j | \]
\[ = \sum_{j,j'} e^{2\pi i (j-j')/N} |j'\rangle \langle j | \delta_{jj'} \]
\[ = \sum_{j,j'} |j\rangle \langle j | = \hat{I}. \]

Hence, \( \hat{F} \) is Unitary.
2.1.3 Product Representation of QFT

The product representation, also considered as the definition of QFT, is very useful in constructing quantum circuits for the transformation. Consider \( N = 2^n \) where \( n \) is some integer, and the \( n \)-qubit basis for computation, \( |0\rangle, \ldots, |2^n - 1\rangle \).

Also, we can represent state \( |j\rangle \) using binary representation \( j = j_1j_2 \ldots j_n \) as,

\[
|j\rangle = |j_1j_2 \ldots j_n\rangle
\]

Adopting another notation, binary fraction, \( 0.j_1j_2 \ldots j_m \) to represent \( j_1/2 + j_1/4 + \cdots + j_m/2^{m-l+1} \). Then the product representation is:

\[
|j_1 \ldots j_n\rangle \rightarrow \frac{|0\rangle + e^{2\pi i 0. j_n}|1\rangle(|0\rangle + e^{2\pi i 0. j_{n-1}j_n}|1\rangle)\ldots (|0\rangle + e^{2\pi i 0. j_1 \ldots j_n} |1\rangle)}{2^n/2}
\]
Equivalence of Representations

The equivalence of the representations (product and the one defined before) can be shown as follows:

\[ |j\rangle \rightarrow \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi i j k}{2^n}} |k\rangle \]

\[ = \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi i j} \sum_{l=1}^{n} k_l 2^{-l} |k_1 \ldots k_n\rangle \]

\[ = \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \]

\[ = \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[ |0\rangle + 2^{2\pi i j 2^{-l}} |1\rangle \right] \]

\[ = \frac{(|0\rangle + e^{2\pi i 0,j_n} |1\rangle)(|0\rangle + e^{2\pi i 0,j_{n-1}j_n} |1\rangle) \ldots (|0\rangle + e^{2\pi i 0,j_1 \ldots j_n} |1\rangle)}{2^{n/2}} \]

Thus we started with the representation similar to DFT and showed that by using simple algebra we get the product representation for QFT.
2.2 Efficient Circuit for QFT

2.2.1 The circuit

The product representation eases the derivation an efficient circuit for QFT (see figure).

\[
\begin{array}{cccccccc}
|\alpha_1\rangle & |H\rangle & R_2 & \cdots & R_{n-1} & R_n & |0\rangle + e^{2\pi i (\alpha_1 \alpha_2 \cdots \alpha_n)} |1\rangle \\
|\alpha_2\rangle & |H\rangle & R_2 & \cdots & R_{n-1} & R_n & |0\rangle + e^{2\pi i (\alpha_2 \alpha_3 \cdots \alpha_n)} |1\rangle \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
|\alpha_{n-1}\rangle & |H\rangle & R_2 & \cdots & R_{n-1} & R_n & |0\rangle + e^{2\pi i (\alpha_{n-1} \alpha_n \cdots \alpha_1)} |1\rangle \\
|\alpha_n\rangle & |H\rangle & R_2 & \cdots & R_{n-1} & R_n & |0\rangle + e^{2\pi i (\alpha_n \alpha_1 \cdots \alpha_{n-2})} |1\rangle \\
\end{array}
\]

Figure 2.1: Efficient Circuit for QFT. Here $\alpha_i$ is equivalent to $j_i$ used before.

2.2.2 Working of the Circuit

We define a gate, $R_k$, that denotes the unitary transformation:

\[
R_k \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i / 2^k} \end{bmatrix}.
\]

Input state is $|j_1 \ldots j_n\rangle$. Applying Hadamard gate to the first bit, gives:

\[
\frac{1}{2^{1/2}} \left( |0\rangle + e^{2\pi i 0 . j_1} |1\rangle \right) |j_2 \ldots j_n\rangle
\]

Then the controlled $R_2$ gate, produces the state:

\[
\frac{1}{2^{1/2}} \left( |0\rangle + e^{2\pi i 0 . j_1 j_2} |1\rangle \right) |j_2 \ldots j_n\rangle
\]

Continually applying the controlled-$R_2$, $R_3$, through $R_n$ gates, each of which adds an extra bit in the phase of the first $|1\rangle$, finally gives the state:

\[
\frac{1}{2^{1/2}} \left( |0\rangle + e^{2\pi i 0 . j_1 j_2 \ldots j_n} |1\rangle \right) |j_2 \ldots j_n\rangle
\]

Now, similar procedure on the second qubit, the Hadamard gate results in,

\[
\frac{1}{2^{2/2}} \left( |0\rangle + e^{2\pi i 0 . j_1 j_2 \ldots j_n} |1\rangle \right) \left( |0\rangle + e^{2\pi i 0 . j_2} |1\rangle \right) |j_3 \ldots j_n\rangle
\]
and the controlled-$R_2$ through $R_n$ gates produce the state,

$$\frac{1}{2^{2/2}} \left( |0\rangle + e^{2\pi i 0,j_1,j_2,...,j_n}|1\rangle \right) \left( |0\rangle + e^{2\pi i 0,j_2,...,j_n}|1\rangle \right) |j_3...j_n\rangle$$

Continuing in this fashion for each qubit, the achieved final state is,

$$\frac{1}{2^{n/2}} \left( |0\rangle + e^{2\pi i 0,j_1,j_2,...,j_n}|1\rangle \right) \left( |0\rangle + e^{2\pi i 0,j_2,...,j_n}|1\rangle \right) \ldots \left( |0\rangle + e^{2\pi i 0,j_n}|1\rangle \right)$$

This is exactly in the opposite order compared to the product representation of QFT. Swap operations (Not shown in the figure) are used to get it in order,

$$\frac{1}{2^{n/2}} \left( |0\rangle + e^{2\pi i 0,j_n}|1\rangle \right) \left( |0\rangle + e^{2\pi i 0,j_{n-1}|1}\rangle \right) \ldots \left( |0\rangle + e^{2\pi i 0,j_1,j_2,...,j_n}|1\rangle \right)$$

Which is the desired output.

### 2.2.3 Analysis of the Circuit

For the first qubit we use one Hadamard gate and $n-1$ conditional rotations, amounting to $n$ gates. For the second qubit we apply total $n-1$ gates (one Hadamard and $n-2$ conditional rotations).

And so on for the rest. This gives us total number of gates required as $n + (n-1) + \cdots + 1 = n(n+1)/2$.

Then we have atmost $n/2$ swap gates, each of which can be realized by using 3 CNOT gates. Thus, this circuit provides $\Theta(n^2)$ algorithm for performing QFT, while the fastest classical algorithm FFT computes DFT using $\Theta(n2^n)$ gates. This means exponential reduction in the number of operations.
Chapter 3

Applications of Quantum Fourier Transform

3.1 Applications of QFT

QFT by itself has very little use, as the amplitudes in a quantum computer cannot be accessed by measurement, leaving no way to determine the fourier transformed amplitudes.

It is mainly used as a component of bigger algorithms, such as Iterative Phase Estimation Algorithm (IPEA), Order-finding and factoring, Period-finding, Discrete logarithms, etc.

Description of some of these applications follows.

3.1.1 Iterative Phase Estimation Algorithm

QFT plays a key role in the phase estimation algorithm, which in turn is used as a subroutine or module in many other quantum algorithms. If we have a unitary operator, $U$ with eigenvector and eigenvalue as $|u\rangle$ and $e^{2\pi i \varphi}$ respectively, where $\varphi$ is unknown, then the goal of phase estimation is to estimate the phase $\varphi$.

Two registers are used in the procedure, first contains $t$ qubits all in the state $|0\rangle$, and the second register begins with the state $|u\rangle$ with as many qubits as necessary to store $|u\rangle$. The value of $t$ for the first register depends on the number of digits of accuracy we want and the probability with which we want the procedure to succeed.
Phase estimation is then done in two stages. First we apply the circuit in figure 3.1. Begins by applying the Hadamard transform to the first register, followed by controlled-$U$ operations on the second register, with $U$ raised to powers of two. The final state of the first register is then:

$$\frac{1}{2^{t/2}} \left( |0\rangle + e^{2\pi i 2^{-1} \cdot 0} |1\rangle \right) \left( |0\rangle + e^{2\pi i 2^{-2} \cdot 0} |1\rangle \right) \ldots \left( |0\rangle + e^{2\pi i 2^{t} \cdot 0} |1\rangle \right) = \frac{1}{2^{t/2}} \sum_{k=0}^{2^{t}-1} e^{2\pi i \phi k} |k\rangle.$$

The state of the second register remains same, $|u\rangle$, during the computation.

In the second stage we apply Inverse QFT on the first register. This is implemented by reversing the circuit for QFT, described before. This can be done in $\Theta(t^2)$ steps.

The final step is to measure the state of the first register in the computational basis. The full schematic of the circuit for IPEA is given in the figure 3.2.

If we express $\phi$ in $t$ bits as, $\phi = 0, \phi_1 \ldots \phi_t$. So, the state from the first register may be written again as,

$$\frac{1}{2^{t/2}} \left( |0\rangle + e^{2\pi i 0, \phi_1} |1\rangle \right) \left( |0\rangle + e^{2\pi i 0, \phi_1 \cdots \phi_t} |1\rangle \right) \ldots \left( |0\rangle + e^{2\pi i 0, \phi_1 \phi_2 \cdots \phi_t} |1\rangle \right).$$
Figure 3.2: Overall circuit for IPEA. The top $t$ qubits are the first register, and the bottom qubits are the second register. The output of the measurement is an approximation to $\varphi$ accurate to $t - \lceil \log 2 + \frac{1}{2\epsilon} \rceil$ bits, with $P(\text{success}) \geq 1 - \epsilon$.

If we compare this expression with the product representation of Fourier transform, we can see that when we apply the Inverse Fourier transform we get the state, $|\varphi_1 \ldots \varphi_t\rangle$. A measurement in the computational basis then exactly gives us $\varphi$.

So in the ideal case stated here, the phase estimation algorithm allows us to estimate the phase $\varphi$ of an eigenvalue of a unitary operator $U$, given the corresponding eigenvector $|u\rangle$, using the ability of the Inverse Fourier Transform to perform the transformation,

$$\frac{1}{2^{t/2}} \sum_{j=0}^{2^t-1} e^{2\pi i \varphi j} |j\rangle |u\rangle \rightarrow |\tilde{\varphi}\rangle |u\rangle,$$

where $|\tilde{\varphi}\rangle$ denotes a state which is a good estimate for $\varphi$ when measured.
Algorithm: Quantum phase Estimation

Inputs: 1. A black box which performs a controlled-$U_j$ operation,
   2. an eigenstate $|u\rangle$ of $U$ with eigenvalue $e^{2\pi i \varphi_u}$, and
   3. $t = n + \log \left(2 + \frac{1}{2\epsilon}\right)$ qubits initialized to $|0\rangle$.

Outputs: An $n$-bit approximation $\tilde{\varphi}_u$ to $\varphi_u$.

Runtime: $O(t^2)$ operations and one call to controlled-$U^j$ black-box. Succeeds with probability at least $1 - \epsilon$.

Procedure:

1. $|0\rangle|u\rangle$       Initial state

2. $\rightarrow \frac{1}{2^{t/2}} \sum_{j=0}^{2^t-1} |j\rangle|u\rangle$       Create Superposition

3. $\rightarrow \frac{1}{2^{t/2}} \sum_{j=0}^{2^t-1} |j\rangle U^j |u\rangle$       Apply Black box

   $= \frac{1}{2^{t/2}} \sum_{j=0}^{2^t-1} e^{2\pi i \varphi_u} |j\rangle|u\rangle$       Result of black box

4. $\rightarrow |\tilde{\varphi}\rangle|u\rangle$       Apply Inverse Fourier Transform

5. $\rightarrow |\tilde{\varphi}\rangle$       Measure first register
References


- *The Quantum Fourier Transform and Jordan’s Algorithm*, Dave Bacon, University of Washington.