QUANTUM FOURIER TRANSFORM AND FAST FOURIER TRANSFORM

DEFINITIONS, COMPARISONS AND CODE REPORT FOR QFT SIMULATION

GALLUCCI NICOLA, GARZONI STEFANO

INDEX

FAST FOURIER TRANSFORM

1. Definition

Fast Fourier Transform algorithm is an efficient algorithm to compute discrete Fourier Transform and its inverse. Given a numeric sequence of N complex numbers $x_0, x_1, x_2, \dots, x_n$, DFT is defined as follows:

$$X_k = \sum_{q=0}^{N-1} x_q e^{-j\frac{2\pi}{N}qk}$$

Where X_k is a discrete sequence of frequencies of period N.

If we consider the numeric sequence $x_0, x_1, x_2, \dots, x_n$ as a column vector x, DFT can be expressed as a matrix multiplication:

$$X = F_N x$$

Where F_N is a matrix such as:

$$\begin{bmatrix} \omega_N^{0\cdot 0} & \cdots & \omega_N^{0\cdot(N-1)} \\ \vdots & \ddots & \vdots \\ \omega_N^{(N-1)\cdot 0} & \cdots & \omega_N^{(N-1)\cdot(N-1)} \end{bmatrix}$$

$\omega_N^{k\cdot s}$ are vectors of a orthonormal basis of C^n defined as: $\omega_N^{k\cdot s} = e^{-j\frac{2\pi}{N}ks}$

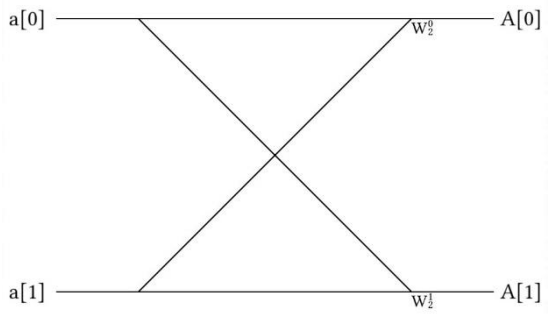
Hence, DFT converts a signal from its original domain (often time) to frequency domain and IDFT does the opposite operation. This operation is useful in many fields but the computation using the definition is quite slow. Indeed, evaluating the definition directly requires $O(n^2)$ operations: there are n outputs X_k and each output requires a sum of n terms. Consequently, FFT has been introduced to optimize DFT algorithm reducing its complexity to $O(n \log n)$ operations.

2. Cooley-Tukey algorithm

The most commonly used FFT is the Cooley-Tukey algorithm, which is a *dividi- et-impera* algorithm. This method recursively breaks down a DFT of any composite size $n = n_2 n_1$ into smaller DFTs size n_2 and n_1, along with $O(n)$ multiplications by

complex roots of unity ω_N. The best way to use this algorithm is to divide the transform into two pieces of size n/2 at each step (*radix-2 case*), but any factorization can be used in general. To have a visual representation of the algorithm, Butterfly Diagrams can be used. These diagrams show where each element in the array goes before, during and after the FFT.

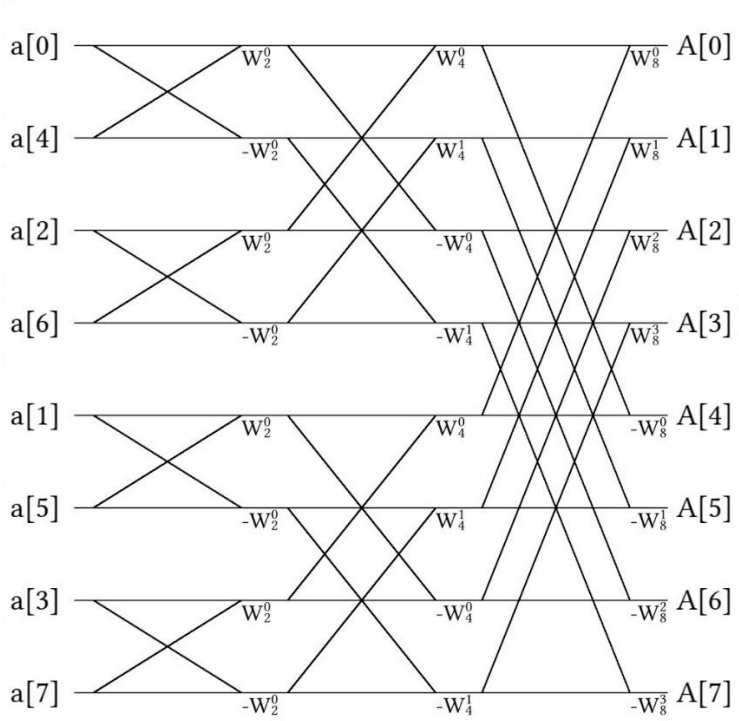
For example, imagine we need to perform an FFT of an array of 2 elements. We can represent this addition with the diagram:



However, it can be proved that the first part of the array of ω_N values is always the negative of the second half $\omega_2^0 = -\omega_2^1$. In practical, the diagram means the following:

$$\begin{cases} b_0 = a_0 + \omega_2^0 a_1 \\ b_1 = a_0 - \omega_2^1 a_1 \end{cases}$$

Imagine now we need to combine more elements (for instance 8 elements). The butterfly diagram will be:



From the graph it's possible to observe that the number of stages of operations is 3 $= \log 8$. Therefore, we perform $O(n)$ multiplications for the roots of unity for $O(\log n)$ stages.

3. **Computational issues**

The complexity of fast Fourier transforms (FFTs) remains a theoretical challenge. While lower bounds exist, it's unclear if DFTs require $O(n \log n)$ operations. Real multiplication bounds are well-understood, but practical implementations face challenges due to excessive additions. Lower bounds on additions are less clear. Efforts to minimize arithmetic complexity led to algorithms like split-radix FFT, but recent work has made significant improvements. Despite focus on complex-data FFTs, advancements would benefit related problems.

In addition, FFT algorithms, like Cooley-Tukey, have small errors with finite-precision floating-point arithmetic due to their structure. Cooley-Tukey's error is $O(\epsilon \log n)$,while the naive DFT's is $O(\epsilon\, n^{3/2})$. Root mean square errors are better: $O(\epsilon \sqrt{\log n})$ for Cooley-Tukey and $O(\epsilon\, n^{1/2})$for naive DFT.

QUANTUM FOURIER TRANSFORM

1. Definition

The Quantum Fourier Transform maps a quantum state $x\rangle = \sum_{i=0}^{N-1} x_i\,|i\rangle$ to $|y\rangle = \sum_{i=0}^{N-1} y_i\,|i\rangle$ in accordance with the formula:

$$y_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n e^{-j2\pi \frac{nk}{N}}$$

QFT can be expressed also in this form: _

$$\mathrm{QFT}(|s\rangle) = \frac{1}{\sqrt{2^n}} \bigotimes_{j=1}^{n} (|0\rangle + e^{2\pi i[0.s_j s_{j+1} \ldots s_n]}|1\rangle) \tag{1}$$

This form represents a time evolution operator constituted by the composition of n^2 gates in parallel, soma Hadamard gates and some controlled phase shift gates as we'll see in the next paragraph.

2. Quantum circuit

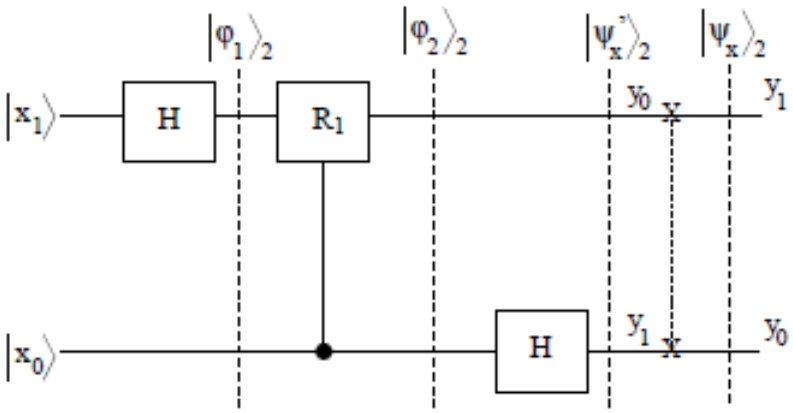
In this section we'll build the quantum circuit to perform a QFT initially with only 2 qubits and, afterwards, we'll generalize for three or more qubits. Consider the Hadamard gate H with the following matrix:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Furthermore, we define a phase gate R_d which makes no change acting on $|0\rangle$, while changes the phase by $\pi/2^d$ acting on $|1\rangle$. This gate has the following matrix:

$$R_d = \begin{pmatrix} 1 & 0 \\ 0 & e^{\pi j/2^d} \end{pmatrix}$$

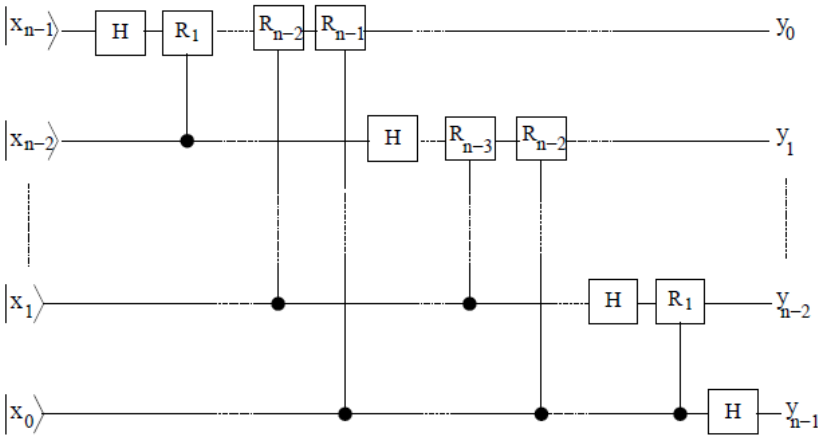
Eventually, the quantum circuit of QFT given 2 qubits is shown below:



The initial state on the left is the single quantum state $|x\rangle_2 = |x_1x_2\rangle$ in the computational basis. The final state on the right is the superposition $|\psi'_x\rangle_2 = \frac{1}{2}\sum_{y=0}^{3} e^{(\pi jxy/2)}|y_0y_1\rangle$,which is almost $|\psi'\rangle_2$, the QFT of $|x\rangle_2 = |x_1x_2\rangle$ given in the definition, except that the order of the bits in the final state is the reverse of what it should be according to definition. This can be corrected by a swap gate as shown in the last part of the picture.

We have shown QFT for a fixed basis state x (in the computational basis) inputted to the left of the circuit diagrams. However, since the circuit is linear (because quantum mechanics is linear) the circuit will act in parallel on a linear superposition of basis states. This is where the power of the QFT lies.

Broadly speaking, for an n-qubit QFT one needs n Hadamard gates. The number of controlled phase gates is $\frac{n(n-1)}{2}$. Also, n/2 swaps are required. Hence, the quantum circuit for a n-qubit QFT is the following:



Please note that in the picture the swap gates are not shown for clarity.

3. Complexity

In equation (1), the larger tensor product symbol functions as an indexed product of the Kronecker tensor product. Via decomposition, the QFT on 2n amplitudes can be implemented as a quantum circuit consisting of $O(n^2)$ Hadamard gates and controlled phase shift gates for a number of qubits n. The previously classical DFT would, in our implementation, require $O(n2^n)$ gates, which is exponentially greater than the aforementioned quantum complexity. This may also be compared with O(n log n), the number of gates required by the most efficient known quantum Fourier transform algorithms for an approximate result.

This analysis, however, does not take into account the complexity of preparing the input state and measuring the output states, which both have complexity $O(N \log 1/\varepsilon)$ for a required resolution It is therefore evident that in the case of the best-known QFT algorithm, the complexity is completely dominated by these measurements. Furthermore, if one wants to read out the full output vector the complexity becomes $O(N^2 \log^2(1/\varepsilon))$.

Up to now we have discussed about the gate-accurate simulation time. We now discuss the complexity of eq. (1) given an input s. We now consider the complexity of computing the Kronecker products in eq. 18. This tensor product is effectively just a form of multiplication: using the associative property, we see that first product involves two vectors of size 2, resulting in a vector of size 4. The second product involves a vector of size 4 and a vector of size 2, resulting in a vector of size 8. This continues to the result of size $N = 2^n$.

To find the full complexity, we sum and take the greater term:

$$O(n^2 + 2^n) = O(2^n) = O(N)$$

This possible implementation of the algorithm is faster than the gate accurate simulation previously discussed. Yet, in simulation we are allowed to copy vectors therefore this does not affect the overall running time.

However, this complexity doesn't appear in an actual quantum computer. Indeed, we start with our initial state X, i.e. a superposition of base vectors. In the actual computation, the calculation of each component is parallelized at no cost, but we can

only measure one coefficient. We therefore need to rebuild the output state N times to be able to perform the N measurements needed to obtain all the coefficients because the wave function of QFT collapses at each measurement. Each of the $n^2 = \log^2 N$ gates require O(N) classical operations to be simulated, thus, the total cost is O ($N \log^2 N$).

In the following table we compare the different complexities in the different cases taken into account in this section.

FFT	$O(N \log N)$
QFT (sim. + no meas.)	$O(\log^2 N)$
QFT (actual quant. Comp.)	$O(N \log^2 N)$
QFT (sim. + prep. + meas.)	$O(N \log^2 1/\varepsilon)$

The previous results appear to confirm that true quantum algorithms and mere simulations of them have significantly different running times. By the principles of quantum information, to acquire results more comparable to those given by the classical DFT, we must make extra preparations to our black box that, in turn, cost us a substantial amount of computing power. More specifically, if we want to measure each coefficient, we must redo our operations for each coefficient (since our wave function will collapse for every measurement). This is not necessary for the classical DFT; upon making these modifications to the theoretical QFT such that it becomes more analogous to the FFT, it is possible that (in the case of Fourier analysis) quantum computing is no better than classical computing. However, this does not necessarily mean that quantum speedup is nonexistent; in the case of integer factorization, wherein we only desire a single answer that meets are conditions, the parallelization of quantum computing can become extremely useful.

COMPARISON BETWEEN FFT AND QFT FOR N=4

As seen in the paragraph "definition" of section 1, FFT can be expressed in a matrix form:

$$\vec{y} = U\vec{x}$$

Where $U = \frac{1}{2}\begin{matrix} 1 & 1 & 1 & 1 \\ 1 & i & i^2 & i^3 \\ 1 & i^2 & 1 & i^2 \\ 1 & i^3 & i^2 & 1 \end{matrix}$

It can be proved that the matrix U can be calculated as the product of two matrices $U_1 \; and \; U_2$, where:

$$U_1 = \frac{1}{\sqrt{2}}\begin{matrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & i^2 & 0 \\ 0 & 1 & 0 & i^2 \end{matrix}$$

$$U_2 = \frac{1}{\sqrt{2}}\begin{matrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & i \\ 1 & i^2 & 0 & 0 \\ 0 & 0 & 1 & i^3 \end{matrix}$$

We will now see that there is a close connection between the FFT and the QFT, in particular, that the transformations U1 and U2 correspond to different parts of the diagram seen in the paragraph "quantum circuit" of the section 2.

The total effect of the quantum circuit, reading from left to right on the circuit, is

given by the matrix product SH1R1Hu = U where S is the matrix of swap gate, Hl is the matrix of Hadamard gate on the lower qubit and Hu on the upper qubit, R1 is the matrix of controlled phase gate. The Hadamard gate H_u has matrix:

$$H_u = \frac{1}{\sqrt{2}}\begin{matrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{matrix} = U_1$$

Hence the first operation U1 in the FFT for N = 4 corresponds, in the QFT, to the Hadamard on the upper qubit, while the second operation U2 in the FFT corresponds to the remaining operations in the QFT: the controlled phase gate on the upper qubit, the Hadamard on the lower qubit, and the swap. To conclude this section, we have seen that there is close connection between the breakup used in the FFT and that used in the QFT. This should not be a surprise. In the FFT we iteratively divide the FT into two FTs of half the length, while in the QFT we have a binary representation of the states and treat each bit in turn, so clearly there is a connection.

CODE REPORT

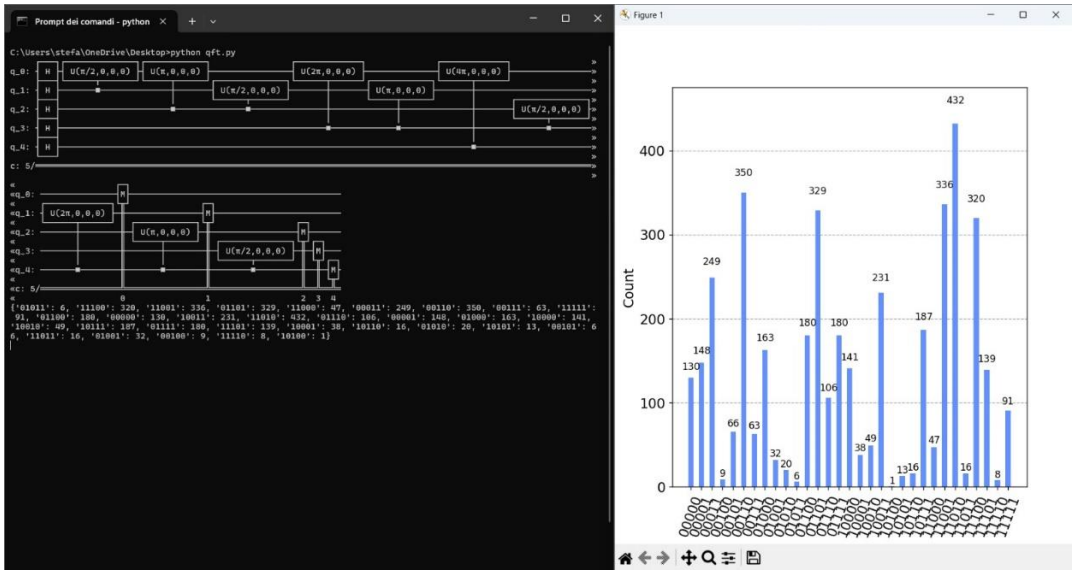
We are looking at a Quantum Fourier Transform (QFT) simulation on a system of 5 qubits.

The circuit created and executed 4096 times, recreates the QFT. Each qubit q_i undergoes a Hadamard gate H, followed by a series of controlled rotation gates U that depend on subsequent qubits q_j (with j>i). This pattern is repeated for all qubits, creating a superposition of states that corresponds to the Fourier transform of the input values.

The simulation results represent the probabilities of measuring each of the possible states of the 5-qubit system after performing the QFT. For example, the state '11010' was measured 432 times, which means that, according to the simulation, this is the most likely state after running QFT on the input state.

QFT transforms basic states into a superposition of states, where the phase of each basic state is proportional to the binary value it represents. Then, the probability distribution we get reflects the Fourier transform of the input values.

We can also note the composition of the circuit and the output states from the histogram and the output received from our simulation program. (image below)



To create the code, we followed these steps:
First, we imported the qiskit libraries necessary for simulating quantum circuits.
Immediately afterwards we are going to define the variables we will need:
the number of qubits and 2 quantum registers created with the number of qubits just defined, which will be used to give shape to the quantum circuit.
Once this is done, we apply the Hadamard transform to each qubit in the circuit using a for and the "h" method.
Then, through two fors we apply the QFT phase using unitary control gates to the right angular phases, also defining a control register and a target register for each application.
At this point the circuit is created and with the "measure" method we can measure the qubits.
Now all we must do is simulate the circuit and then output the results found.

ACKNOWLEDGMENTS

1. **Documents**

MUSK Damian, "*A Comparison of Quantum and Traditional Fourier Transform Computations"*, 23 Nov 2020.

YOUNG Pete, "The Quantum Fourier Transform and a Comparison with the Fast Fourier Transform", 2 Nov 2019.

BARLETTA Luca, MAGARINI Maurizio, "*Lecture notes - Quantum information processing 101*"

2. Websites

https://www.algorithm-archive.org/contents/cooley_tukey/cooley_tukey.html

https://en.wikipedia.org/wiki/Fast_Fourier_transform

https://www.ibm.com/quantum/qiskit