Quantum Fourier Transforms: Conceptual Underpinnings, Computational Implementation, and Significance

Priti Rangnekar

PHYSICS 14N, Winter 2021

Abstract

In this paper, I discuss the Quantum Fourier Transform, which is emphasized as "the single most important quantum subroutine" in *Quantum Computing: A Gentle Introduction* by Eleanor Rieffel and Wolfgang Polak. First, I describe the motivation behind classical discrete Fourier transforms, along with the key mathematical concepts behind their implementation. Second, I provide a walkthrough of the mathematical and conceptual implementation of a Quantum Fourier Transform (QFT). Third, I explain my computational implementation of the QFT using Qiskit, which is an open-source framework for quantum computing, along with an analysis of the results (given through a link to documented code with results). Fourth, I provide a general overview of the applications of the QFT for phase estimation and Shor's algorithm. Finally, I review recent developments and novel approaches that have emerged regarding QFT research and applications.

1 Classical Discrete Fourier Transforms

1.1 Motivation

Imagine you were given a smoothie and were asked to determine all of the ingredients, along with the quantity of each. At first thought, this may seem like an insurmountable task, unless one had particularly sensitive and discerning taste buds. However, if you could run the smoothie through a filter and extract each ingredient individually, the task would be more straightforward and methodical to tackle. Challenges of this nature are not uncommon in the real world. A musician listening to a song may want to identify the constituent frequencies and amplitudes of the notes that sum to produce the complex notes. A computer scientist may want to compress an image by eliminating frequency components that provide pixel-to-pixel variation that our eyes cannot notice. Each one of these problems can be addressed by applying the Fourier transform, named after Joseph Fourier, who showed in 1822 that some functions can be written as an infinite sum of harmonics. The Fourier transform is a mathematical transform that is used to decompose a function that depends on space or time into constituent functions that depend on spatial or temporal frequency.

1.2 Discrete Fourier Transform

We will focus our discussion on the discrete Fourier transform (DFT), which operates on a discrete complex-valued function and produces a different discrete complex-valued function. This transform is best understood mathematically. A vector $(x_0, ..., x_{N-1})$ is mapped to the vector $(y_0, ..., y_{N-1})$ using the following formulas:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega_N^{jk} \tag{1}$$

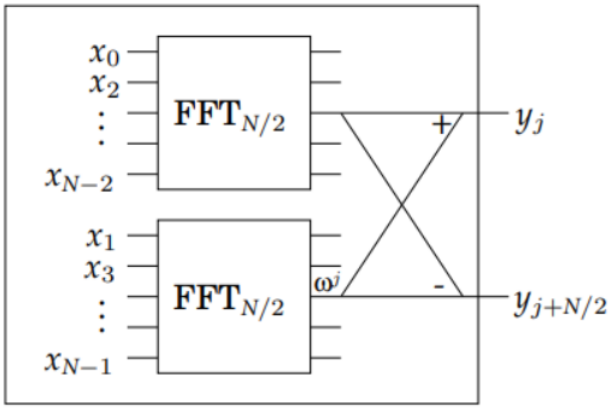
$$\omega_N^{jk} = e^{2\pi i \frac{jk}{N}} \tag{2}$$

The term with "$e^{2\pi i}$" is particularly of interest, as it presents a parallel to Euler's formula $e^{\pm i\theta} = \cos\theta \pm i \sin\theta$. This is due to the fact that the idea behind the Fourier Transform is that we can represent a periodic function as a sum of these complex exponentials - as a combination of sine waves and cosine waves.

Thus, the DFT can also be thought of as a linear transformation using the following matrix.

$$W = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}$$

A naive implementation of this matrix-vector multiplication would take $O(N^2)$ operations. However, the technique of the Fast Fourier Transform (FFT) can reduce this complexity to $O(N(logN))$ by taking a recursive approach. The crucial observation is that it is possible to decompose the N x N matrix F(n) in terms of Fourier transforms for lower powers of 2. One of the most common means of implementation is the radix-2 decimation-in-time (DIT) FFT, a form of the Cooley-Tukey FFT algorithm. This technique first computes the DFTs of even-indexed inputs (x_0, x_2, and so on) and the odd-indexed inputs (x_1, x_3, and so on). These results are then combined to produce the DFT of the whole sequence; this idea is performed recursively to keep expressing a DFT of length N in terms of two DFTs of size N/2. By reusing results of intermediate computations to compute several DFT outputs, the complexity is reduced to $O(N(logN))$.



2 Introduction to Quantum Fourier Transforms

Having discussed the motivations, concept, and mathematics surrounding classical discrete Fourier transforms, we now look to the quantum Fourier transform (QFT). The QFT is very similar but works with state vectors. The QFT acts on a quantum state and maps it to a different quantum state as shown below.

$$|x\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega_N^{xy} |y\rangle \tag{3}$$

This can also be expressed with the following unitary matrix.

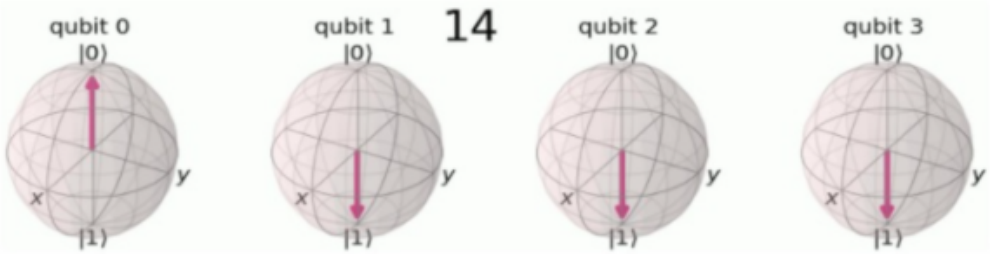
$$U_{QFT} = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \omega_N^{xy} |y\rangle\langle x| \tag{4}$$

An intuitive manner of understanding the QFT is by noting that the QFT transforms between the computational basis and the Fourier basis. Each basis vector is transformed to a superposition state in which the amplitudes are the same as those in the DFT. After the QFT, the incoming amplitude of a basis vector in the original space becomes distributed among all basis vectors in the Fourier space. In addition, the matrix is unitary; since the computational basis is orthonormal, the transformed basis must also be orthonormal.

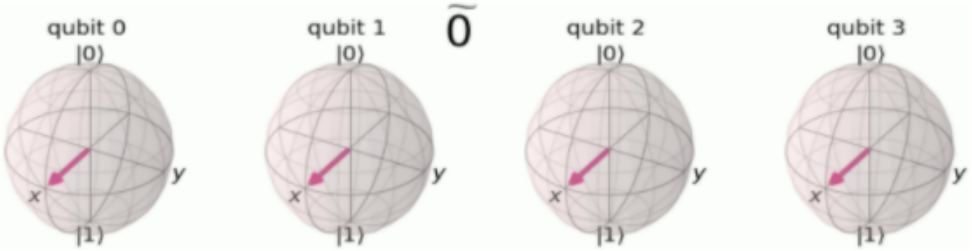
2.1 Quantum Fourier Transforms and Rotations

Another powerful means of understanding the QFT is through visualizing rotations on the Bloch sphere. As mentioned earlier, the QFT transforms between the computational basis and the Fourier basis. We will use this fact in order to better understand the QFT through a methodical process.

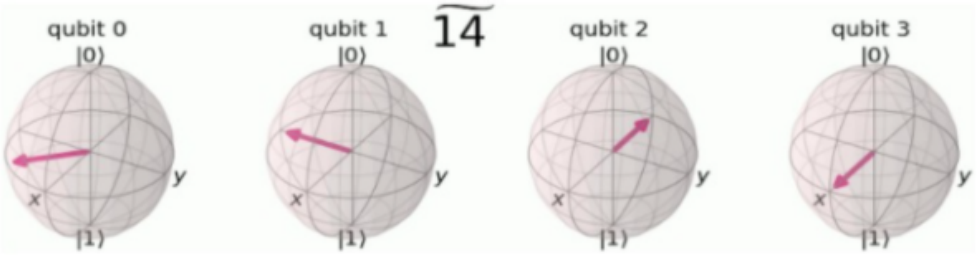
First, we must choose a number and express it in binary. As an example, we can use the decimal number 14, which is expressed as 1110 in binary. Second, we encode it in the computational basis; we can choose to use 4 qubits. As shown on the Bloch spheres below, the qubits go from 0 to 3 from left to right, with the arrows pointing to either $|0\rangle$ or $|1\rangle$ in the standard basis. In this case, qubit 0 refers to the least significant qubit - it has the least effect on the value of the number. In the case of 1110, the least significant bit is rightmost and is 0.



The third step is to determine how much each qubit needs to rotate counterclockwise by about the Z-axis. In the Fourier basis, the "0" state is represented by $|\widetilde{0}\rangle$ and has all qubits in the state $|+\rangle$.



Other states in the Fourier basis are represented through rotations for each qubit with respect to this original $|\widetilde{0}\rangle$ state. Qubit 0 refers to the leftmost qubit on the Bloch spheres below. For a given n-qubit state $|\widetilde{D}\rangle$, where n is the total number of qubits and D is the number to encode, qubit i is rotated by $\frac{D}{2^{n-i}}(2\pi)$ radians. For example, in the case of $|\widetilde{14}\rangle$, qubit 0 is rotated by $\frac{14}{2^{4-0}}(2\pi)$ radians, which is $\frac{14}{16}$ of a full turn counterclockwise about the Z axis. Qubit 1 is rotated by $\frac{14}{2^{4-1}}(2\pi)$ radians, which is $\frac{14}{8}$ of a full turn, or $\frac{3}{4}$ of a full turn in net displacement. Qubit 2 is rotated by $\frac{14}{2^{4-2}}(2\pi)$ radians, which is $\frac{14}{4}$ of a full turn, or $\frac{1}{2}$ of a full turn in net displacement. Qubit 3 is rotated by $\frac{14}{2^{4-3}}(2\pi)$ radians, which is $\frac{14}{2}$ of a full turn, or 0 full turns in net displacement.



We now have our state in the Fourier basis.

2.2 Example of the 1-qubit QFT

In order to see a simple example of how these rotations are applied, we will consider the 1-qubit QFT.

To begin, we have a single qubit state. $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Our vector can be represented with $x_0 = \alpha, x_1 = \beta$. Additionally, since we have n=1 qubit, N = 2^n is N=2.

By using the QFT equations for y_0 and y_1 and incrementing j and k, we see the following.

$$y_0 = \frac{1}{\sqrt{2}}\left(\alpha \exp\left(2\pi i \frac{0\times 0}{2}\right) + \beta \exp\left(2\pi i \frac{1\times 0}{2}\right)\right) = \frac{1}{\sqrt{2}}(\alpha+\beta)$$

$$y_1 = \frac{1}{\sqrt{2}}\left(\alpha \exp\left(2\pi i \frac{0\times 1}{2}\right) + \beta \exp\left(2\pi i \frac{1\times 1}{2}\right)\right) = \frac{1}{\sqrt{2}}(\alpha-\beta)$$

Thus, the resulting superposition is given by the final state below.

$$U_{QFT}|\psi\rangle = \frac{1}{\sqrt{2}}(\alpha+\beta)|0\rangle + \frac{1}{\sqrt{2}}(\alpha-\beta)|1\rangle$$

If we had applied the Hadamard operator (H) on the same qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, we would get the same result. Thus, the Hadamard gate essentially performs the 1-qubit QFT, and the discrete Fourier transform for N=2 on the state's amplitudes; it writes each amplitude in the form of a sum.

3 Quantum Fourier Transform: Mathematics

We will now demonstrate the precise mathematical steps behind the QFT. Before we begin, we know the following conditions. We have n qubits, and the maximum number of possible states is given by N.

$$N = 2^n$$
$$|x\rangle = |x_1 \dots x_n\rangle$$

We start with the definition of the QFT.

$$QFT_N|x\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1}\omega_N^{xy}|y\rangle$$

We also know the following:

$$\omega_N^{xy} = e^{2\pi i \frac{xy}{N}} \quad N = 2^n$$

Thus, the right hand side can be rewritten, as shown below.

$$\frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} e^{2\pi i xy/2^n}|y\rangle$$

Additionally, the following are known, with the second equation following from the first.

$$y = [y_1 y_2 \dots y_n] = 2^{n-1}y_1 + 2^{n-2}y_2 + \dots + 2^0 y_n = \textstyle\sum_{k=1}^{n} y_k 2^{n-k}$$
$$y/2^n = \textstyle\sum_{k=1}^{n} y_k/2^k$$

This allows us to, once again, rewrite the right-hand side as shown below.

$$\frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} e^{2\pi i\left(\sum_{k=1}^{n} y_k/2^k\right)x}|y_1 \dots y_n\rangle$$

After expanding the exponential of a sum to a product of exponentials, rearranging the sum and products, and expanding the sums, we are left with the following expression for the QFT of the initial state.

$$\frac{1}{\sqrt{N}}\left(|0\rangle + e^{\frac{2\pi i}{2}x}|1\rangle\right)\otimes\left(|0\rangle + e^{\frac{2\pi i}{2^2}x}|1\rangle\right)\otimes\dots\otimes\left(|0\rangle + e^{\frac{2\pi i}{2^{n-1}}x}|1\rangle\right)\otimes\left(|0\rangle + e^{\frac{2\pi i}{2^n}x}|1\rangle\right)$$

4 Quantum Fourier Transform: Circuit Gates

The QFT can be implemented using a circuit that uses 2 gates. The first is the Hadamard gate. In fact, the Hadamard gate can be shown to perform the Fourier transform for N=2 on the state's amplitudes. Thus, we have the following equation.

$$H\left|x_k\right\rangle = \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + \exp\left(\frac{2\pi i}{2}x_k\right)\left|1\right\rangle\right)$$

In addition, a 2 -qubit controlled rotation CROT_k is required. The UROT $_k$ gate is used to change the relative phase, equivalent to a 1 -qubit rotation about the Z axis.

$$CROT_k = \begin{bmatrix} I & 0 \\ 0 & UROT_k \end{bmatrix} \quad UROT_k = \begin{bmatrix} 1 & 0 \\ 0 & \exp\left(\frac{2\pi i}{2^k}\right) \end{bmatrix}$$

Suppose we have the following 2 -qubit state.

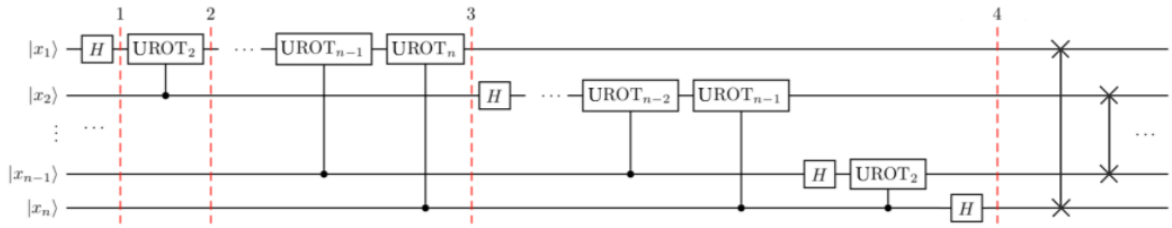
$$\left|x_j x_k\right\rangle$$

Since the first qubit is the control, and the second qubit is the target, the following equations arise. In the first case, the gate merely acts as the identity.

$$CROT_k\left|0x_j\right\rangle = \left|0x_j\right\rangle$$
$$CROT_k\left|1x_j\right\rangle = \exp\left(\tfrac{2\pi i}{2^k}x_j\right)\left|1x_j\right\rangle$$

5 Quantum Fourier Transform: Quantum Circuit and Proof

In order to implement this Quantum Fourier Transform computationally, we can use the circuit below for an n-qubit QFT.



The initial n -qubit input state is given below.

$$\left|x_1 x_2 \ldots x_n\right\rangle$$

After the first Hadamard gate is applied on the first qubit, the state is transformed to the one below.

$$\frac{1}{\sqrt{2}}\left[\left|0\right\rangle + \exp\left(\frac{2\pi i}{2}x_1\right)\left|1\right\rangle\right] \otimes \left|x_2 x_3 \ldots x_n\right\rangle$$

The application of the UROT_2 gate, in which qubit 2 is the control and qubit 1 is the target as shown, causes the state to be further transformed.

$$\frac{1}{\sqrt{2}}\left[\left|0\right\rangle + \exp\left(\frac{2\pi i}{2^2}x_2 + \frac{2\pi i}{2}x_1\right)\left|1\right\rangle\right] \otimes \left|x_2 x_3 \ldots x_n\right\rangle$$

The last UROT_n gate is then applied, we are left with the following state.

$$\frac{1}{\sqrt{2}}\left[\left|0\right\rangle + \exp\left(\frac{2\pi i}{2^n}x\right)\left|1\right\rangle\right] \otimes \left|x_2 x_3 \ldots x_n\right\rangle$$

When the procedure of gates is applied for the remaining qubits, a final state is obtained.

$$\frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2^n}x\right)|1\rangle\right] \otimes \frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2^{n-1}}x\right)|1\rangle\right] \otimes \ldots \otimes \frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2^2}x\right)|1\rangle\right] \otimes \frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2^1}x\right)|1\rangle\right]$$

Clearly, this state is the same as the one that was obtained through mathematical manipulation earlier, as long as we reverse the order of qubits.

6 Computational Implementation in Qiskit

Qiskit is an open-source framework for quantum computing that includes tools for creating quantum programs and circuits. In addition, it provides the capability for running them on quantum computers and simulators.

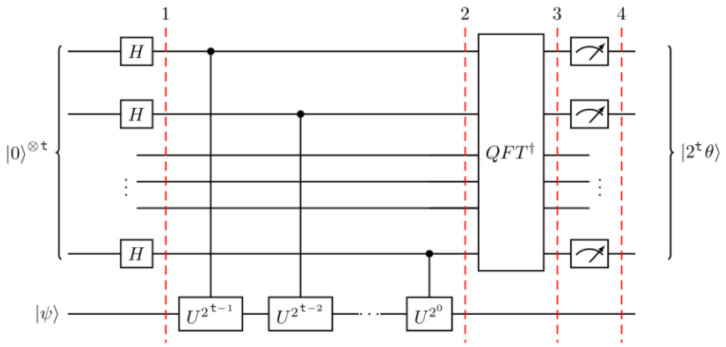
`https://github.com/PritiRangnekar/Phys14NFinal` includes 1) functional code in Qiskit, 2) visualizations of circuits, the Bloch sphere, and statistical results, and 3) documentation explaining the steps taken and analyzing the results, so that readers can follow. The results have been confirmed to be correct and match expectations. I recommend referring to section 2.1 of this article in order to visualize some of the expected results, as produced by Qiskit's Bloch Sphere visualization simulations.

7 Applications of QFTs for Important Algorithms and Methods

The QFT plays a crucial role in the implementation of important algorithms and subroutines throughout quantum computing. I will focus on two: quantum phase estimation and Shor's algorithm.

7.1 Quantum Phase Estimation

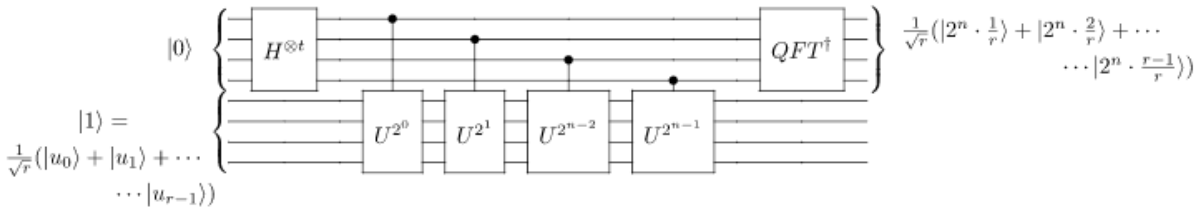
The quantum phase estimation subroutine takes a unitary operator U and estimates θ in θ in $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$. The algorithm writes the phase of U in the Fourier basis to t qubits in the counting register. Then, it uses the inverse QFT to convert from the Fourier basis to the computational basis, allowing for measurement. When a qubit is used to control the U-gate, the qubit turns proportionally to $e^{2i\pi\theta}$. The algorithm uses CU gates, or controlled-U gates, for repeating this rotation until the phase, θ, has been encoded as a number between 0 and 2^t in the Fourier basis. Then, the inverse QFT can be used for conversion back to the computational basis. This process is represented by the circuit below.



7.2 Shor's Algorithm and Period Finding

Quantum phase estimation is particularly useful for Shor's algorithm, which is one of the most famous and important algorithms in quantum computing. This algorithm is used for period finding. Period finding is important because integers can be factored efficiently by using efficient period finding algorithms. Integer factorization is a problem that is particularly of interest for mathematicians, computer scientists, and quantum computing specialists alike. The reason is that the best-known classical algorithm needs superpolynomial

time to factor the product of 2 primes. As a result, the effectiveness of currently popular cryptosystems, such as RSA, is largely contingent on the currently true assumption that factoring is virtually impossible for large enough integers. In period finding, we take a function $f(x) = a^x$ mod N, in which a and N are positive integers where a is less than N and the two do not have any common factors. The period is the smallest nonzero integer that satisfies the equation a^r mod N = 1. Shor's algorithm uses quantum phase estimation on the unitary operator $U|y\rangle = |aymodN\rangle$. Below is a representation of the circuit for Shor's algorithm.



8 Novel Advancements and Recent Approaches

Although quantum Fourier transforms have been applied for subroutines such as quantum phase estimation and Shor's algorithm for nearly decades, quantum Fourier transform research remains relevant and has been thriving in recent years. I discuss two different developments to provide a snapshot of this research: the application of QFTs for image processing and improvements in QFT at a theoretical level.

8.1 The Application of QFTs for Image Processing

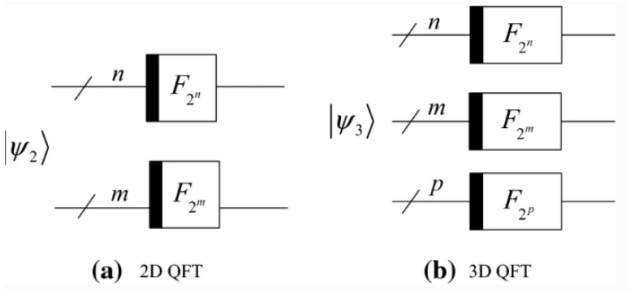
In 2018, Li et. al. published a paper, "The quantum Fourier transform based on quantum vision representation." Essentially, the investigated a different topic, quantum vision representation (QVR) and then proposed a 2-dimensional and 3-dimensional QFT based on QVR. First, they represented a natural image with size 2^n x 2^m as an angle matrix, in which $theta_{x,y}$ represents color details of the pixel on the coordinate (x, y).

$$\Lambda_{2^n,2^m} = \begin{bmatrix} \theta_{0,0} & \theta_{0,1} & \cdots & \theta_{0,2^m-1} \\ \theta_{1,0} & \theta_{1,1} & \cdots & \theta_{1,2^m-1} \\ \vdots & \vdots & \cdots & \vdots \\ \theta_{2^n-1,0} & \theta_{2^n-1,1} & \cdots & \theta_{2^n-1,2^m-1} \end{bmatrix}$$

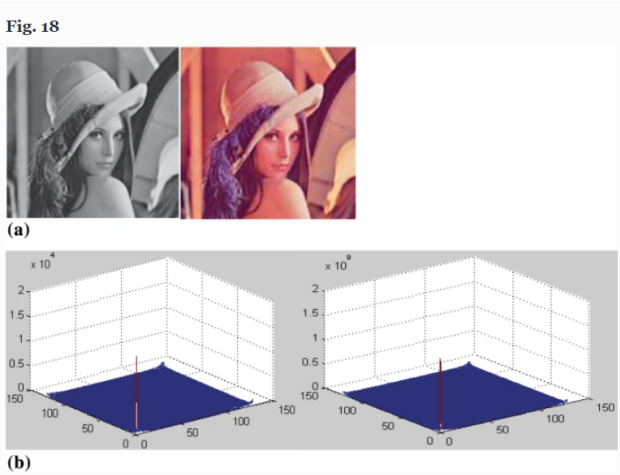
Similarly, for a video of 2^p frames of size 2^n x 2^n, they generated the following angle matrix.

$$A_{2^n,2^m,2^p} = \left(\Lambda^1_{2^n,2^m}, \Lambda^2_{2^n,2^m}, \cdots, \Lambda^{2^p}_{2^n,2^m}\right)$$

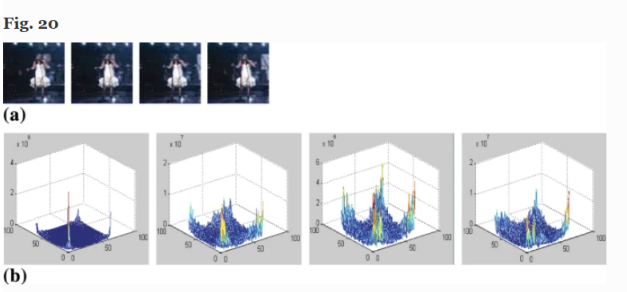
As for the actual implementation of the QFT, they developed a 2D QFT with complexity O(n^2 x m^2) for a 2^n x 2^m image; for videos, a 3D QFT with complexity O(n^2 x m^2 + p^2) was developed. It is important to note that the researchers found a significant benefit to using multidimensional QFTs over multidimensional DFTs. The multidimensional QFTs could be implemented by parallel running multiple 1D QFTs. On the other hand, the multidimensional DFTs must execute multiple 1D DFTs in sequence.



Their results were proven to be successful after testing on two images (grayscale and color) and two videos (grayscale and color).



The 2D QFT on two 128 × 128 images. **a** Original images; **b** Amplitude spectrums



The 3D QFT on a color video. **a** The four 64 × 64 frames; **b** Amplitude spectrums of the four frames

8.2 Improvements in QFT Theory

In 2020, Nam, Su, and Maslow published "Approximate quantum Fourier transform with O(nlog(n)) T gates." Due to the fact that many rotation gates in the QFT will have very small angles, the QFT can be implemented at an approximate level by removing rotation gates with angles below a certain threshold. In fact, an algorithmic accuracy of over 99.992% can be achieved by using an AQFT with approximately 53,000 CROT gates to factor 2048-digit numbers. Additionally, due to concerns with errors in quantum computing, large-scale quantum algorithms must use fault-tolerant computing, in which errors are suppressed by using multiple physical qubits to encode a single logical qubit. This can be achieved by using gates that are constructible fault tolerantly, such as the Clifford+T gates. In the past, the approach for implementing an n-qubit AQFT had been to remove small-angle CROT gates to bring down the gate count to O(nlog(n)) from O(n^2), along with replacing remaining O(n log(n)) CROT gates with Clifford+T implementations.

However, this circuit would have a T-count of O(n log^2(n)). The authors, however, were able to create a more efficient implementation with a T-count complexity of O(nlog(n)) by mapping controlled rotations into uncontrolled ones, inducing uncontrolled rotations that come in layers using integer adders and gate reusal, and uniting individual controlled rotations into n - 1 sets separated by Hadamard gates. As shown below, these strategies were largely proven to be effective, reducing the number of gates significantly.

	Our AQFT implementation			AQFT with controlled-z^a per[30] (Fig. 10)			Optimized AQFT[37]		
Circuit	n_q	**CNOT**	**T**	n_q	**CNOT**	**T**	n_q	**CNOT**	**T**
$AQFT_8$	25	390	303	9	336	1083	8	56	1821
$AQFT_{16}$	51	1798	1162	17	1404	6309	16	234	7815
$AQFT_{32}$	67	4654	2698	33	3900	19,261	32	650	22,683
$AQFT_{64}$	99	10,366	5770	65	8892	47,099	64	1482	54,269
$AQFT_{128}$	163	21,790	11,914	129	18,876	106,631	128	3146	123,333
$AQFT_{256}$	291	44,638	24,202	257	38,844	229,729	256	6474	267,007
$AQFT_{512}$	547	90,334	48,778	513	78,780	476,873	512	13,130	553,277
$AQFT_{1024}$	1059	181,726	97,930	1025	158,652	993,727	1024	26,442	1,148,497
$AQFT_{2048}$	2083	364,510	196,234	2049	318,396	2,084,983	2048	53,066	2,427,081
$AQFT_{4096}$	4131	730,078	392,842	4097	637,884	4,316,993	4096	106,314	4,993,035
Complexity	$O(n)$	$O(n\log(n))$	$O(n\log(n))$	$O(n)$	$O(n\log(n))$	$O(n\log^2(n))$	$O(n)$	$O(n\log(n))$	$O(n\log^2(n))$

As for future work, we see potential in laying out the author's circuit and methodology in restricted architectures with a constant SWAP overhead, so the increase in CNOT gate count due to SWAP operations is under control. In addition, the authors recommend optimizing depth by introducing additional gates that require more space but reduce depth.

9 Conclusion

It is not difficult to see why the quantum Fourier transform may be considered one of the most important subroutines in quantum computing. ultimately, its mathematical elegance rooted in the insights and equations by Euler and Fourier, strategic application of recursive techniques and quantum gates, crucial role in more complex quantum algorithms, potential for application in novel fields, and prospects for greater efficiency will continue to make the QFT a worthy concept for investigation.

10 References

1. Rieffel, Eleanor, and Wolfgang Polak. Quantum Computing: a Gentle Introduction. MIT Press, 2014.
2. The Qiskit Team. Learn Quantum Computation Using Qiskit, 16 Mar. 2021, qiskit.org/textbook/preface.html.
3. Coles, Patrick J. et al. "Quantum Algorithm Implementations for Beginners." ArXiv abs/1804.03719 (2018): n. pag.
4. Nam, Y., Su, Y. Maslov, D. Approximate quantum Fourier transform with O(n log(n)) T gates. npj Quantum Inf 6, 26 (2020). https://doi.org/10.1038/s41534-020-0257-5
5. Li, HS., Fan, P., Xia, Hy. et al. The quantum Fourier transform based on quantum vision representation. Quantum Inf Process 17, 333 (2018). https://doi.org/10.1007/s11128-018-2096-2