The Approximate Quantum Fourier Transform (AQFT) and its Applications

Hania Shahid (24100044)
Rehan Ahmad (24100219)
Sharjeel Ahmad (24100083)

Abstract

This paper aims to showcase the mathematical apparatus behind Approximate Quantum Fourier Transforms (AQFT) and their formulation. After establishing a solid mathematical theory, we will then go on to show the strength of AQFT in various quantum computing algorithms like periodicity estimation and quantum addition, in terms of its resource efficiency when stacked up against mainstream computational methods.

1 Introduction

The quantum Fourier transform (QFT) plays a critical role in various quantum algorithms, including Shor's factoring algorithm. However, a significant challenge with implementing QFT is the need for a large number of qubits, which can increase the likelihood of decoherence. Decoherence, which arises from the entanglement of input states with the environment, can adversely affect the accuracy and reliability of quantum computations. To address this issue, the approximate quantum Fourier transform (AQFT) has been developed, which reduces the number of gates required for the algorithm, thereby minimizing the impact of gate-related decoherence

In this context, we will explore the impact of using QFT and AQFT in the Shor's algorithm. Specifically, we will perform the algorithm with N = 21 and x = 9 using both QFT and AQFT and compare the results. Additionally, we will analyze how decoherence can affect the outcomes of both implementations. It is worth noting that AQFT can be more efficient and robust than QFT in certain scenarios where input data may be subject to small variations or noise. Furthermore, by reducing the number of gates needed, AQFT can enable more efficient non-classical algorithms, leading to more powerful computations. Through our analysis, we hope to provide insights into the trade-offs between QFT and AQFT and how they can impact the practical implementation of quantum algorithms.

For the case of quantum addition, the AQFT significantly reduces the run-time need to perform the algorithm. Since there are much fewer gates needed, a more efficient non-classical algorithm can be implemented, making the computation more powerful.

2 Mathematical Formulation of AQFT

2.1 Quantum Fourier Transform

The discrete Fourier transform (DFT) converts a finite sequence of equally-spaced samples of a function into a same-length sequence of equally-spaced samples of the discrete-time Fourier transform (DTFT), which is a complex-valued function of frequency. The DFT is a unitary transformation on a s-dimensional vector,

$$f(0), f(1), f(2), \dots, f(s-1)$$

and is defined by the expression

$$\tilde{f}(c) = \frac{1}{\sqrt{s}} \sum_{a=0}^{s-1} \exp(2\pi iac/s) f(a) \tag{1}$$

where $f(a)$ and $\tilde{f}(c)$ are usually complex numbers. s is assumed to be a power of 2, i.e. $s = 2^L$ where L is some integer. We usually use powers of 2 when binary coding, for the sake of convenience. If a and c are L-bit integers, they can be represented as:

$$a = \sum_{i=0}^{L-1} a_i 2^i; \; c = \sum_{i=0}^{L-1} c_i 2^i \tag{2}$$

The quantum Fourier transform is simply the DFT but in terms of qubits instead of bits. It is precisely the same function and can be written as

$$|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \exp\left(\frac{2\pi i}{N} kj\right) |k\rangle \tag{3}$$

where $N = 2^n$ where n denotes the number of qubits we are dealing with.

The product ac in (1) can be be expanded using the binary notation in (2). Let us first expand ac and then investigate the consequences in (1).

$$\begin{aligned} ac = (a_0c_0) + 2(a_0c_1 + a_1c_0) + 2^2(a_0c_2 + a_1c_1 + a_2c_0) + \dots \\ +2^{L-1}(a_0c_{L-1} + \dots + a_{L-1}c_0) + \mathcal{O}(2^L) \end{aligned} \tag{4}$$

Note that powers of a_1c_1 are not stacked together for the term $(a_0c_1 + a_1c_0)2$ because in general both a_1 and c_1 would have 2 multiplied with them such their product would have the power 2^2. Then, in terms of the binary representation, the $\exp(2\pi iac/2L)$ in the R.H.S. of equation 1 becomes

$$\begin{aligned} \exp(2\pi iac/2L) = \exp\bigl(2\pi i(a_0c_0)/2^L\bigr) \exp\bigl(2\pi i(a_0c_1 + a_1c_0)/2^{L-1}\bigr) \dots \\ \times \exp(2\pi i(a_0c_{L-1} + \dots + a_{L-1}c_0)/2). \end{aligned} \tag{5}$$

We can then write the full expression to be

$$|c\rangle = \frac{1}{\sqrt{2^L}} \sum_{a=0}^{L-1} |a\rangle \exp\left(\frac{2\pi i}{2^L} \sum_{j,k=0}^{L-1} a_j c_k 2^{j+k}\right) \tag{6}$$

2.2 How the QFT is approximated

Notice that from the right hand of the expression in (4) the arguments in the exponential become smaller and smaller. In the approximate Fourier transform parameterised by an integer m, the $L - m$ smallest terms are neglected. In all the remaining terms the arguments are then multiples of $\frac{2\pi}{2m}$.The 2^mth root of unity becomes the basic element of the approximate Fourier transform as opposed to 2^nth root of unity which used in the ordinary Fourier transform. (The ordinary Fourier transform is obtained for $m = L$; when $m = 1$ we obtain the Hadamard transform, for which all terms but the last one is dropped.)

Keeping the above discussion in mind, the Fourier transform in (6) becomes:

$$|c\rangle = \frac{1}{\sqrt{2^L}} \sum_{a=0}^{L-1} |a\rangle \exp\left(\frac{2\pi i}{2^L} \sum_{j,k=0}^{L-1} a_j c_k 2^{j+k}\right) \tag{7}$$

Whenever $j + k \geq N$ we have terms that are of higher order than $N - 1$ which do not contribute to the transform. Therefore, equation 7 becomes

$$|c\rangle = \frac{1}{\sqrt{2^L}} \sum_{a=0}^{L-1} |a\rangle \exp\left(\frac{2\pi i}{2^L} \sum_{\substack{0\leq j,k\leq L-1 \\ 0\leq j+k\leq L-1}}^{L-1} a_j c_k 2^{j+k}\right) \tag{8}$$

The condition $j + k \leq L - 1$ is just saying that the power of 2 should not exceed $L - 1$. On the other hand, a and b should always lie between 0 and $L - 1$ which is once again encapsulated by the above equation. Both these conditions *must* be satisfied so that we stay consistent with the expansion in (4).

Let us now parameterize equation 8 by an integer m which lies in the interval $0 < m < L$ such that we would ignore the lower order powers in equation 4, only taking into account powers which are greater than $L - m$. With that, equation 8 becomes:

$$|c\rangle = \frac{1}{\sqrt{2^L}} \sum_{a=0}^{L-1} |a\rangle \exp\left(\frac{2\pi i}{2^L} \sum_{\substack{0 \leq j,k \leq L-1 \\ L-m \leq j+k \leq L-1}}^{L-1} a_j c_k 2^{j+k} \right) \tag{9}$$

Note that what changed from equation 8 to equation 9 is the lower bound of the sum within the exponential. Moreover, when $L = m$, equation 9 reduces to the ordinary QFT.

In equation 9, since $L - m \leq j + k$, the argument of exponential is some multiple of $\frac{2\pi i 2^(L-m)}{2^L} = \frac{2\pi i}{2^m}$. The argument of the exponential in AQFT differs from that of QFT by

$$\frac{2\pi i}{2^L} \sum_{j+k<L-m} a_j c_k 2^{j+k}$$

The execution time of the AQFT grows with $\sim Lm$.

3 Applications

3.1 Estimating Periodicity for N = 21, x = 9

The quantum Fourier transform is a powerful method for unveiling periodicities. Any periodicity in probability amplitudes describing a quantum state of a register on a computational basis can be estimated via the Quantum Fourier Transform, proceeded by a measurement of the register. Reading the qubits of the register in the reversed order gives us the result. To explain, let's take a look at Shor's algorithm, which can be written as:

$$|\Psi\rangle = \frac{1}{\sqrt{N}} \sum_{a=0}^{2^L-1} f(a) |a\rangle ,$$

Where N serves as the normalization factor and $f(a)$ is the state of a quantum register of size L in which the probability amplitudes f(a) occur with a periodicity of r and an offset of l. Assuming we don't know the offset, measurements performed on the state will not yield r or its integer multiples. Furthermore, by performing QFT on the register and then measuring the state, we get a number $\bar{c}$ whose probability does not equal unity because of the register's finite size. On the other hand, using the AQFT, we get a probability that satisfies

$$\text{Prob}_A \geq \frac{8}{\pi^2} \sin^2\left(\frac{m\pi}{4L}\right).$$

In cases where QFT forms a part of a randomized algorithm, the computation must be repeated a few times to highlight the correct answer. For such circumstances, the AQFT is polynomially less efficient. If we consider Shor's quantum factoring algorithm and replace the AQFT with the QFT. In order to obtain a correct factor with a prescribed probability of success, we have to repeat the computation several times. Let k and k' be the number of times we need to run QFT and AQFT, respectively, to get at least one correct result. Their ratio gives us:

$$\frac{k'}{k} = \frac{\log\left(1 - \frac{4}{\pi^2}\right)}{\log(1 - H)} < C\left(\frac{L}{m}\right)^3$$

where $H = \frac{8}{\pi^2} \sin^2(\frac{4m}{L\pi})$ and C is an upper bound found through graphical means. This shows that for the quantum factoring algorithm, AQFT has a shorter runtime.

3.2 Decoherence

Let us take a look at a simple mathematical model of decoherence. If we assume that the environment effectively acts as a measuring apparatus, a single qubit in state $c_0 |0\rangle + c_1 |1\rangle$ evolves together with the environment as

$$(c_0 |1\rangle + c_1 |1\rangle) |a\rangle \longrightarrow c_0 |0\rangle |a_0\rangle + c_1 |1\rangle |a_1\rangle \tag{10}$$

where states $|a\rangle$,$|a_0\rangle$, and $|a_1\rangle$ are the states of the environment; $|a_0\rangle$, and $|a_1\rangle$ cannot be assumed to be orthogonal, as they usually aren't. The elements of the density matrix evolve as

$$\rho_i j(0) = c_i(0)c_j^*(0) \longrightarrow \rho_i j(t) = c_i(t)c_j^*(t)\langle a_i(t)|a_j(t)\rangle;\ i, j = 0, 1 \tag{11}$$

Another way to think about this is if the environment is regarded as a bosonic heat bath, which introduces phase fluctuations to the qubit states, such that it induces random phase fluctuations in the coefficients c_0 and c_1 as following

$$c_0 |0\rangle + c_1 |1\rangle \to c_0 e^{-i\phi} |0\rangle + c_1 e^{i\phi} |1\rangle \tag{12}$$

The direction and the magnitude of each phase fluctuation ϕ is chosen randomly following the Gaussian distribution

$$P(\phi)d\phi = \frac{1}{\sqrt{2\pi}\delta} \exp\left[-\frac{1}{2}\left(\frac{\phi}{\rho}\right)^2\right] d\phi \tag{13}$$

where the distribution width δ defines the strength of the coupling to the quantum states of the environment.

If we evaluate the performance of the AQFT by introducing a quality factor, Q, which measures the likelihood of obtaining an integer that is closest to any integer multiple of $2^L/r$ when the transformed register state is measured. In a decoherence-free environment, we achieve $Q = 1$ for integer values of $2^L/r$, and for a randomly selected r, the quality factor for the QFT is approximately $4/\pi^2$, while the AQFT of degree m has a quality factor of approximately $8\pi^2 \sin^2(4\pi mL)$.

For $\delta > 0$, a maximum for Q is obtained for $m < L$. Thus, for the case of decoherence, the AQFT is better than QFT.

3.3 Quantum Addition

In the context of quantum computing, addition algorithms have typically been based on classical methods but modified for reversible computation. While faster quantum addition algorithms have been developed using carry-save techniques, they still follow a classical approach. However, it's possible that the most efficient addition algorithm for a quantum computer may differ significantly from classical methods.

To perform quantum addition, a series of conditional rotations that are mutually commutative are utilized. This structure is similar to the quantum Fourier transform, but with the difference that the rotations are dependent on n external bits. This feature is particularly useful when adding classical data to quantum data. The addition is performed according to Figure 1.
The gates used in 1 are conditional rotation gates. The conditional rotation gate performs a phase rotation between two qubits conditioned on their superposition (visualized in Figure 2).

Due to the strong similarity between the implementation of quantum addition and the QFT, it's not unexpected to find a more effective approximate implementation of quantum addition that utilizes the same methodology as the AQFT.

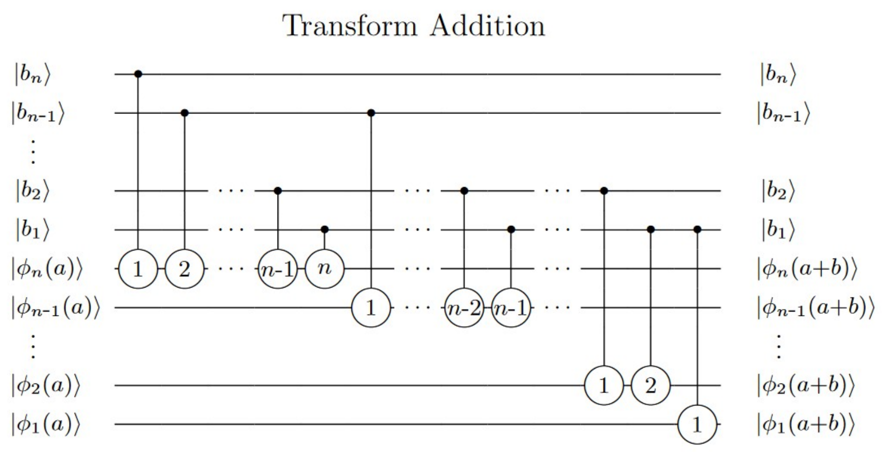

Figure 1: Logic Circuit to perform quantum addition using AQFT

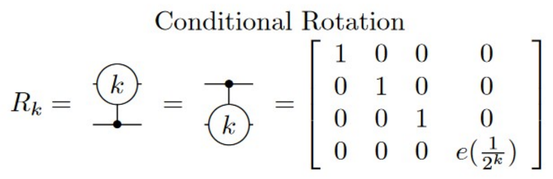
Conditional Rotation
$R_k =$
k
=
k
=
$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e(\frac{1}{2^k}) \end{bmatrix}$

Figure 2: Conditional rotation gate.

The main distinction between quantum addition and the quantum Fourier transform (QFT) lies in the fact that all operations in quantum addition commute with one another, while the Hadamard transforms used in the QFT require a specific order of operations. As a result, a quantum computer with the ability to execute multiple independent gate operations concurrently will have a proportionally faster runtime.

Specifically, if a quantum computer can perform n^2 independent 2-qubit gate operations concurrently, quantum addition can be completed in roughly $n + 1$ time intervals. However, if the AQFT method is used to eliminate rotations below a certain threshold, quantum addition can be completed in $\log(2n)$ time intervals.

References

[1] Barenco, Adriano, et al. “Approximate Quantum Fourier Transform and Decoherence.” Physical Review A, vol. 54, no. 1, 1996, pp. 139–146., https://doi.org/10.1103/physreva.54.139.

[2] D. Coppersmith, ”An approximate Fourier transform useful in quantum factoring”, 2002, IBM Research Report, ArXiv, https://arxiv.org/pdf/quant-ph/0201067v1.pdf.

[3] “Discrete Fourier Transform.” Wikipedia, Wikimedia Foundation, 3 Apr. 2023, https://en.wikipedia.org/wiki/Discrete_Fourier_transform.

[4] Thomas G. Draper, ”Addition on a Quantum Computer”, arXiv, 2000, https://doi.org/10.48550/arXiv.quant-ph/0008033