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,

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

and is defined by the expression

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

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).

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

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

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

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 Periodicity Estimation

3.1 Estimating Periodicity using QFT

Our main aim is to work through a particular problem in mind. Suppose we have the number $N = 21$ and our initial guess is $x = 9$. We apply Shor's algorithm to find the prime factors of r. Firstly, we choose our q such that:

$$2N^2 > q \geq N^2 \quad \Rightarrow \quad 441 \leq q < 882. \tag{10}$$

Here, $N = 21$, then $441 \leq q < 882$. We choose $L = 9$ such that $q = 512$. Since $m = 2L$, we get $m = 18$. Our initial state is of the form:

$$|\psi_0\rangle = |0\rangle^{\otimes 9} |0\rangle^{\otimes 18}. \tag{11}$$

Note that the input register must contain enough qubits to represent numbers as large as $q - 1$. In this case, $q - 1 = 511$. On the other hand, the output register must contain enough qubits to represent numbers as large as $N - 1$. In this case, that corresponds to up to 20, and so we need 5 qubits to represent the output. On the m-qubits, we perform the Hadamard gate:

$$|\psi_1\rangle = \frac{1}{\sqrt{512}} \sum_{a=0}^{511} |a\rangle \otimes |0\rangle^{\otimes 9}. \tag{12}$$

We now apply the Order-finding Unitary operator:

$$\begin{aligned} |\psi_2\rangle &= U_N^{(a)} \left(\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle \right) \otimes |0\rangle^{\otimes 9} \\ &= \frac{1}{\sqrt{512}} \sum_{a=0}^{511} |a\rangle \otimes \big(|0\rangle^{\otimes 9} |2^a \bmod 21\rangle \big) \end{aligned}$$

Which yields:

$$|\psi_2\rangle = \frac{1}{\sqrt{512}} \sum_{a=0}^{511} |a\rangle \otimes |2^a \bmod 21\rangle. \tag{13}$$

The periodicity of the function $f(a) = 2^a \bmod 21$ is 6. Evaluating the modulus yields output $1, 2, 4, 8, 16, 11,$. If we expand the state for some values of a, and then collect like terms corresponding to states in the second qubit, we get:

$$\begin{aligned}|\psi_2\rangle = \frac{1}{\sqrt{512}}&\left(|0\rangle + |6\rangle + |12\rangle + |18\rangle\right)\ldots|1\rangle\\ &+ \left(|1\rangle + |7\rangle + |13\rangle + |19\rangle\right)\ldots|2\rangle\\ &+ \left(|2\rangle + |8\rangle + |14\rangle + |20\rangle\right)\ldots|4\rangle\\ &+ \left(|3\rangle + |9\rangle + |15\rangle + |21\rangle\right)\ldots|8\rangle\\ &+ \left(|4\rangle + |10\rangle + |16\rangle + |22\rangle\right)\ldots|16\rangle\\ &+ \left(|5\rangle + |11\rangle + |17\rangle + |23\rangle\right)\ldots|11\rangle.\end{aligned}$$

We can see that the state in the second qubit is in the superposition of 6 different states, each of which corresponds to one of the possible periods of the function $f(a) = 2^a\, mod\, 21$. Applying the QFT on the first register yields of our state $|\psi_2\rangle$ gives:

$$|\psi_3\rangle = \frac{1}{512}\sum_{a=0}^{512}\sum_{y=0}^{512} e^{\frac{2\pi i a y}{512}}\,|y\rangle\,|2^a \mod 21\rangle \tag{14}$$

Suppose that after measurement, the state collapses to the states in which $x^a \mod N$ holds for some fixed value of z. For example, suppose that the state collapses to $z = |8\rangle$, then the corresponding values that y can take place is:

$$(|3\rangle + |9\rangle + |15\rangle + |21\rangle)...$$

The smallest a such that $f(a) = 8$ is 3. Therefore, $l = 3$. On the other hand, the value of $f(a) = 8$ cycles back after 6. Therefore, $k = 6$. Hence,

$$|\psi_4\rangle = \frac{1}{q}\sum_{a=0}^{q-1}\sum_{y=0}^{q-1} e^{\frac{2\pi i a y}{q}}\,|y\rangle\,|z\rangle$$

$$|\psi_4\rangle = \frac{1}{512}\sum_{a=0}^{511}\sum_{y=0}^{511} e^{\frac{2\pi i(3+6r)y}{512}}\,|y\rangle\,|8\rangle \tag{15}$$

The probability of obtaining some measurement $|y\rangle\,|8\rangle$ is given by:

$$P(y|z) = \left|\frac{1}{q}\sum_k e^{\frac{2\pi i(l+rk)y}{q}}\right|^2$$

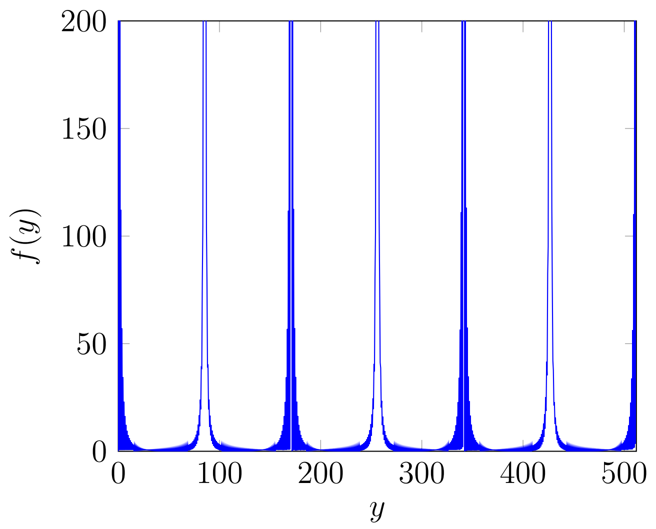
$$P(y|8) = \left|\frac{1}{512}\sum_k e^{\frac{2\pi i(3+6k)y}{512}}\right|^2 \tag{16}$$

where $k = \left\lfloor\frac{q-l-1}{r}\right\rfloor = 84$.

The term $|e^{2\pi i l/q}|^2$ would evaluate to 1. On the other hand, we can compute $|\frac{1}{q}\sum_k e^{2\pi i(rk)y/q}|^2$ using the geometric series to obtain: $\left|\frac{1}{q}\sum_k e^{2\pi i(rk)y/q}\right|^2 = \frac{1}{q^2}\left|\frac{e^{2\pi i(rky)/q}-1}{e^{2\pi i r y/q}-1}\right|^2$. Then we can simplify further using the identity $\left|(1-e^{i\theta})\right| = 2\left|\sin\left(\frac{\theta}{2}\right)\right|$. After plugging in the particulars, we obtain:

$$P(y|8) = \frac{1}{512^2}\left|\frac{\sin(504\pi y/512)}{\sin(6\pi y/512)}\right|, \tag{17}$$

If we plot $P(y|8)$ (non-normalized), we obtain spikes at 84, 168, 252, 336, 420 in the interval $y = [0, 512]$:



We would now like to put constraints for $\tilde{c}$ being the closest integer multiple to $\lfloor \frac{q-l-1}{r} \rfloor = \lfloor 84.66 \rfloor = 84$. We put the constraint that $\tilde{c}$ satisfy:

$$-\frac{1}{2} < \tilde{c} - \frac{q}{r}\lambda < \frac{1}{2}$$

$$-\frac{1}{2} < \tilde{c} - 84.66\lambda < \frac{1}{2} \tag{18}$$

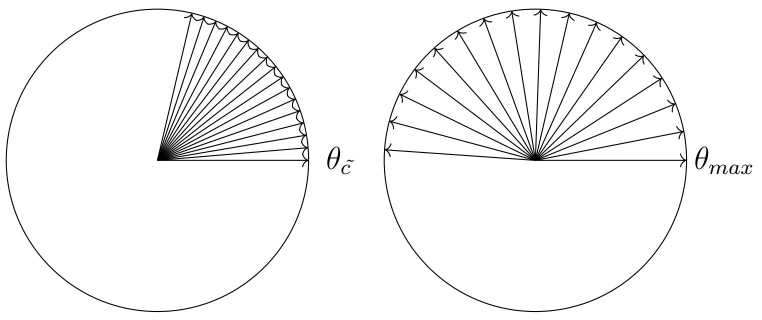
Where $\lambda = 1, 2, 3, \ldots, r$. Define:

$$\theta_{\tilde{c}} = \frac{2\pi r}{q}$$

Then $P(y|8)$ assumes the form:

$$P(y|8) = \left| \frac{1}{512} \sum_k e^{i\theta_{\tilde{c}} k y} \right|^2 \tag{19}$$

As we increase the value of $\theta_{\tilde{c}}$, the terms in the series rotate around the origin, and the sum of these terms forms a vector that also rotates. At some point, if we keep increasing $\theta_{\tilde{c}}$ some of the vectors will start to cancel each other out, resulting in a smaller total distance from the origin. Therefore, to minimize the probability, we need to find the largest allowed $\theta_{\tilde{c}}$, which corresponds to the minimum distance from the origin. We define the angle at which this occurs at θ_{max}. Our argument can be summarized in the illustration below:



Therefore, $P(y|8)$ is minimized for the maximum value of $\theta_{\tilde{c}}$. From equation (18), this occurs at:

$$\theta_{\text{max}} \leq \frac{\pi r}{2^9} = \frac{6\pi}{512}$$

For this value, from [1], we find:

$$P(y|8) \geq \frac{6}{512^2} \frac{1}{\sin^2(\frac{\pi}{2}\frac{6}{512})} \approx \frac{4}{\pi^2}\frac{1}{6}$$

Since there are r such values of $\tilde{c}$, the total probability of seeing one of them is:

$$P(y|8) \geq \frac{4}{\pi^2} = 0.405 \tag{20}$$

3.2 AQFT and Estimating Periodicity

Let us now introduce AQFT into the mix. From equation (7), we can introduce the difference between the terms of QFT and AQFT in terms of the correction function $\Delta(k, j)$.

$$\Delta(k,j) = \frac{2\pi}{2^L}\left(jk - \sum_{\substack{0\le a,b\le L-1 \\ L-m\le a+b\le L-1}}^{L-1} k_a j_b 2^{a+b}\right) \tag{21}$$

Our aim is to find maximum bound for $\Delta(k, j)$ for various values of m with $L = 9$. To begin, we make the assumption like in the paper [1] that $k_i j_i = 1$ for all i so that the terms in the AQFT is maximized. For any other value, we would have $0 < \Delta(k, j) \le \Delta_{\text{max}}(k, j)$:

$$\Delta_{\text{max}}(k,j) = \frac{2\pi}{2^9}\left(jk - \sum_{\substack{0\le a,b\le 8 \\ 9-m\le a+b\le 8}}^{8} 2^{a+b}\right) \tag{22}$$

For $m = 1$, we would have AQFT equal to the Hadamard gate and $m = 8$ the AQFT would equal the QFT. Rather than use the relation $\Delta_{\text{max}} = \frac{2\pi}{2^m}(L - m - 1 + 2^{m-L})$ as employed [1], we would aim to evaluate the sum $\sum^{8}_{\substack{0\le a,b\le 8 \\ 9-m\le a+b\le 8}} 2^{a+b}$ directly for various values of m and see how this changes the probability of obtaining $\tilde{c}$ from equation (20).

To give some idea about how we would go about this, for $m = 1$, the values of a, b that satisfy the condition $a+b = 8$ is the set $\{(8,0),(7,1),(6,2),(5,3),(4,4),(3,5),(2,6),(1,7),(0,8)\}$. Therefore, the total terms in the sum becomes $9(2^8) = 2304$. For $m = 2$, we obtain all the values corresponds to 2^8 in addition to the set $\{(0,7),(6,1),(5,2),(4,3),(3,4),(2,5),(1,6),(7,0)\}$ which gives $9(2^8) + 8(2^7) = 3328$. We would continue this for various values of m. The table of result we find for $S = \sum^{8}_{\substack{0\le a,b\le 8 \\ 9-m\le a+b\le 8}} 2^{a+b}$ for various values of m is,

Value of m	S
1	2304.
2	3328.
3	3776.
4	3968
5	4048
6	4080
7	4092
8	4096
9	4097

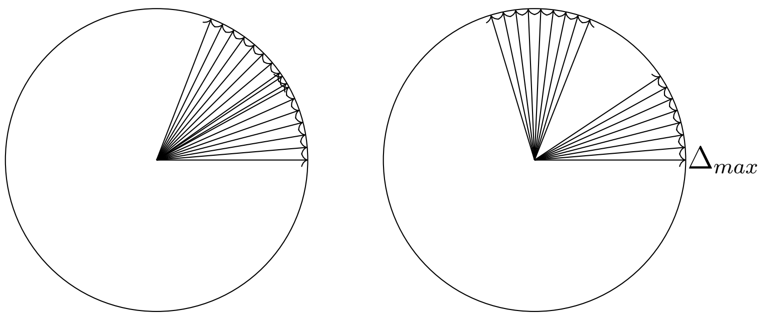
Table 1: Values of S for different values of m

The values of Δ_{max} assume the form:

Values of m	Δ_{max}
1	22.00
2	9.43
3	3.94
4	1.36
5	0.60
6	0.21
7	0.06
8	0.01
9	0.00

Table 2: Maximum deviation Δ_{max} for different values of m

For these various values of m, we can now introduce corrections to probability in equation (20). Once again, since we are interested in the lower bound for probability, and so we allow the vectors to fill one half of the circle such that $\theta_{max} = \frac{\pi r}{2^L}$. The effect of Δ_{max} can then be seen as a rotation on each vector by the maximum angle Δ_{max}.



The minimum of the probability is obtained when half of the vectors are rotated by $\Delta_{\max}$ as shown by the figure above. In this case, vectors in two areas of size $\Delta_{\max}$ cancel each other and all we have to do is calculate geometrical sums of the vectors in the two areas of size $\frac{\pi}{2} - \Delta_{\max}$. In an area of that size there are $\frac{q}{r}\left(\frac{1}{2} - \frac{\Delta_{\max}}{\pi}\right)$ vectors. Hence, the sum in equation (19) becomes modified and reads as thus,

$$\left| \sum_{j=0}^{\frac{q}{r}\left(\frac{1}{2}-\frac{\Delta_{\max}}{\pi}\right)-1} e^{i\frac{\pi r}{q}j} \right|^2 = \frac{\sin^2\left(\frac{1}{2}(\frac{\pi}{2} - \Delta_{\max})\right)}{\sin^2\left(\frac{\pi}{2}\frac{q}{r}\right)}$$

Such that,

$$Prob_A \geq \frac{2r}{q^2}\frac{\sin^2\left(\frac{1}{2}(\frac{\pi}{2} - \Delta_{\max})\right)}{\sin^2\left(\frac{\pi}{2}\frac{q}{r}\right)} \approx \frac{8}{\pi^2}\sin^2\left(\frac{1}{2}(\frac{\pi}{2} - \Delta_{\max})\right) \tag{23}$$

The expression (23) is useful in our analysis because it gives us the bound for $\Delta_{\max}$. From equation (23), we see that for $\Delta_{\max} = \frac{\pi}{2}$, we have $Prob_A = 0$. Therefore to avoid a non-zero probability $\Delta_{\max}$ must always be bounded by: $\Delta_{\max} < \frac{\pi}{2}$. For L = 9, we see that only for the following values of m do we have non-zero probability:

$$m > 3 \tag{24}$$

Therefore, for AQFT to give us a meaningful probability of the state $P(y|\vec{c})$, m must have a larger value than 3. For these values,

$$Prob_A > \frac{8}{\pi^2}\sin^2\left(\frac{\pi}{4}\frac{m}{L}\right) \tag{25}$$

From equation (25), these probabilities correspond to:

m	Minimum Probability to obtain $\tilde{c}$
8 (QFT equivalent)	0.4058
7	0.326
6	0.250
5	0.1801
4	0.11870

Table 3: Minimum Probability to obtain $\tilde{c}$ for different values of m

From the table, we see that the minimum probability of obtaining the current value of $\tilde{c}$ for these values of m remain significant enought that given enough reruns, AQFT is a viable alternative to QFT.

Let us now consider the gates costs. For QFT, the number of gates required (not including the number of swaps) is $\frac{(L(L+1))}{2}$. The major distinction between AQFT and QFT is that AQFT reduces the number of controlled rotations performed on each qubit. If k is the qubit that acts as control, then AQFT imposes the constraint $\theta_{jk} \equiv \frac{\pi}{2^{k-j}} < \frac{\pi}{2^m}$ as discussed in [1]

In that case, we need n Hadamard operations (just like in a normal QFT) and $\frac{(2n-m)(m-1)}{2}$ controlled rotations. For $L = 9$, the number of gates required for different values of m are as follows:

m	Number of Gates Required for $L = 9$
8 (QFT equivalent)	45
7	42
6	39
5	35
4	30

Table 4: Number of Gates Required for $L = 9$ for different values of m

Therefore, using $m = 4$ for $L = 9$ reduces the number of gates required to perform QFT by $\frac{1}{3}$ while still obtaining a meaningful value for $\tilde{c}$. This insight becomes increasingly more meaningful as we consider many real-world implementations of quantum systems which are sensitive to gate-related decoherence. Our analysis can then be carried out for any number of qubits L to choose a meaningful value of m that minimizes the number of gates required while also giving a big enough probability to obtain $\tilde{c}$

3.3 Qiskit Code Implementation

For Qiskit Implementation, we created a code that highlights how AQFT differs from QFT for a scalable Quantum circuit from L = 1 to L = 8 Qubits. The main insight that we utilized was putting the constraint on the number of rotation gates using m. Only controlled rotation gates that met the criterion $\theta_{jk} \equiv \frac{\pi}{2^{k-j}} < \frac{\pi}{2^m}$ were included in the circuit. Furthermore, we carried out the Shor's factoring on Qiskit for the case of N = 21 with the initial guess this time being 21. For this, the most challenging part was creating a function that carries modular exponentiation $2^x \mod 21$. Although the task of carrying out modular exponentiation classically is not a big issue and can be achieved by exponentiation by squaring approach, creating quantum circuit implementations for modular exponentiation is a much more difficult task. We can develop a trial-and-error analysis for a small-scale system like ours (512 qubits) rather than getting to the general case N. The general case is discussed in the paper [4]. On the other hand, a good source for implementing small-scale algorithms is [5]. The approach we utilized to implement our function was to take inspiration from the case $2^x \mod 15$ whose circuital design can be implemented from swaps $(0, 3), (3, 2), (2, 1)$ where 3 represents the leftmost qubit, and 0 represents the right-most qubit. We noticed that the change for the circuit $2^x \mod 21$ is that after 16, we have to account for the change 11. That is, we need somehow to implement the transformation $16 \Rightarrow 11$. By trial and luck method, we were able to find that a conditional flip achieves this aim. Finally, we generated the periodicity of $2^x \mod 21$, which was only achieved in this case by introducing a threshold value of 50, below which all counts were ignored. The reason we had to introduce this was that in our case, $\frac{q}{r}$ was not an exact integer, unlike algorithms on the net, which usually take the case $2^x \mod 15$ in which r divides q exactly.

4 Decoherence

4.1 Theoretical Background

Decoherence refers to the phenomenon experienced when quantum states interact with the environment and become correlated [6]. This is because quantum information spills into the environment, creating impure and/or mixed states, preventing them from interfering to produce accurate results. In general, quantum computation requires a precisely calibrated and controlled system evolution due to the probabilistic and seemingly random nature of quantum particles like photons. Even an infinitesimally small (by classical standards) interaction with the environment would non-unitarily evolve the pure input quantum state into a mixed state which can be illustrated through a reduced density operator as such [1]:

$$\rho = Tr_{enviournment}(\rho_{total}), \tag{26}$$

$Tr_{enviournment}$ is the trace over all the quantum states, and ρ_{total} is the combined density operator of the environment and the quantum computer.

Let us now 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:

$$(c_0 |1\rangle + c_1 |1\rangle) |a\rangle \longrightarrow c_0 |0\rangle |a_0\rangle + c_1 |1\rangle |a_1\rangle \, , \tag{27}$$

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

$$\rho_i j(0) = c_i(0) c_j^*(0) \longrightarrow \rho_i j(t) = c_i(t) c_j^*(t) \langle a_i(t) | a_j(t) \rangle, \tag{28}$$

where $i = 0$ and $j = 1$. These states become increasingly orthogonal as the quantum state continuously interacts with the surroundings; however, the coefficients c_i do not vary. Eventually, the off-diagonal elements will vanish due to the $\langle a_0(t)| \, |a_1(t)\rangle$ element, but the diagonal itself is not altered.

Another way to illustrate decoherence is imagining the environment as a bosonic heat bath. This introduces phase fluctuations to the qubit states and induces random phase fluctuations in the coefficients c_0 and c_1 as follows;

$$c_0 |0\rangle + c_1 |1\rangle \to c_0 e^{-i\phi} |0\rangle + c_1 e^{i\phi} |1\rangle \, . \tag{29}$$

The direction and the magnitude of each phase fluctuation ϕ are chosen at random through the Gaussian distribution;

$$P(\phi) d\phi = \frac{1}{\sqrt{2\pi\delta}} \exp\left[-\frac{1}{2} \left(\frac{\phi}{\rho} \right)^2 \right] d\phi. \tag{30}$$

In the equation above, δ signifies the strength of the coupling to the quantum states of the environment. The density matrix can be remodeled as $\rho_{ij} = \langle c_i c_j^* \rangle$ where the average value is taken over various phase fluctuations in a specified time period. Just like before, the off-diagonals of the density matrix tend to zero after an extended period of time.

4.2 AQFT and Decoherence

We now create a quantum network that consists of gates A and B. Gate A acts as a simple Hadamard gate, while gate B is a two-bit state which introduces a phase factor of $exp(i\theta_{jk})$ to the $|11\rangle$ state. Here, $\theta_{jk} = \frac{\pi}{2^{k-j}}$ and this allows us to model our decoherence. Note that we have not attached any decoherence-causing element to gate A as a single qubit operation for decoherence is computationally quick to carry out like the ion trap model [7] while conditional two-qubit operations would change out time scale.

We introduce the quality factor of Q, which represents the probability of hitting the closest possible integer multiple of $2^L/r$ when the register's state is measured after a quantum evolution and/or transformation. Without decoherence, we can assume $Q = 1$ for integer values of $2^L/r$. Now for the Quantum Fourier Transform, the quality factor is on the order of $4/\pi^2$. Still, for the Approximate Quantum Fourier Transform of degree 'm', our quality factor comes out on the order of $\frac{8}{\pi^2}(\frac{4m}{\pi L})$.

The quality factor Q acts as a function of δ and m, which showcases how strong the system is coupled with the environment. For $\delta > 0$, the highest value of Q is generated for $m < L$, which implies that using AQFT would save for resources than QFT.

This result is not very surprising when considering that AQFT uses fewer gates in the quantum network than the QFT and that introducing phase fluctuations in the B gate of AQFT produces less decoherence. Hence by decreasing our m we are able to mitigate the effects of decoherence at the expense of having approximation which reduces the value of Q. This "opportunity cost" between the two factors peaks for the maximum value of Q for $m \in [1, L]$. This proves that for cases of decoherence, it is computationally more efficient to use AQFT rather than QFT.

4.3 Factoring and Decoherence

In order to introduce decoherence to Shor's algorithm, we introduce the environment as an external system to the quantum network. The new input is as follows:

$$|\tilde{\psi_1}\rangle = \frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}|a,0\rangle \times |\epsilon\rangle\,, \tag{31}$$

where the degrees of freedom of the external system (environment) are represented by ϵ. At the beginning of the computation, the quantum state and environment are completely untangled or uncorrelated. But we know for the calculation of x^a mod N; there must be some interaction between the state and environment; this leads to the evolution as follows:

$$|\tilde{\psi_2}\rangle = \frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}|a, x^a \mathrm{mod} N\rangle \times |\epsilon_a\rangle\,. \tag{32}$$

The state is now partially correlated with the quantum state. If the quantum bits of the system are diagonal in the pointer basis of the environment, then decoherence has no effect when measuring the second label of $|\tilde{\psi_2}\rangle$ the computation of the qubits for the factoring algorithm. So for the purposes of this explanation, we mute the second label and take a look at the first label by tracing over the environment. The result is the reduced density matrix:

$$\rho_{red} = \frac{1}{q}\sum_{a=0}^{q-1}{}'\sum_{a=0}^{q-1}{}'[1-\beta_{aa'}]\,|a\rangle\,\langle a'|\,, \tag{33}$$

where $1 - \beta_{aa'} = |\langle\epsilon_a\rangle\,\epsilon_a'|^2$ gauges the degree with which the quantum state and the environment have become correlated. The impact of decoherence can be approximated using a different function: $1 - \beta_{aa'} \equiv \delta_{aa'} + (1 - \delta_{aa'})\beta$ where β is a constant. $\beta = 1$ would represent a state of full decoherence while $\beta = 0$ would represent a state of full coherence. In the limit of $\beta \to 1$ we can quantify β as the fractional loss of information to the environment as the state evolves. This can be written as:

$$\frac{S_f}{S_{max}} = 1 - (1-\beta)^2, \tag{34}$$

where S_f is the difference in entropy between the final and initial quantum computer state, while S_{max} is the system's entropy after it is completely decoherent. To clarify this, for $\beta \approx 0.5$, the probability of obtaining the correct and incorrect answer is almost the same. Finally, once $(1-\beta)^{-1} \sim \mathcal{O}(\exp\big[(\log N)^{1/3}\big])$, the quantum computer needs around $\exp\big[(\log N)^{1/3}\big]$ number of trials in order to factor N. This means the quantum computer uses just as many resources as a classical computer after a certain level of decoherence has been reached in the system. This phenomenon of decoherence has been the largest reason for skepticism in developing quantum computers, as the fragile nature of the system makes decoherence almost inevitable. However, due to the emergence of techniques like AQFT and quantum error correction, we have become more and more equipped to deal with decoherence while maintaining algorithmic efficiency.

4.4 Qiskit Implementation

Decoherence was implemented on the N = 21 and x = 2 Shor's factoring algorithm discussed in section 3.3. In order to model Decoherence over the system, the "qiskit_aer.noise" library was imported, and the "pauli_error" function was used to generate noise in the algorithm. The Pauli error was added to the controlled X-gates used in the algorithm to generate the unitary gates needed. The computational load put on the processor to stimulate decoherence was immense. To subvert this issue, the number of counting qubits was reduced to 5 in order to compile the data in a reasonable amount of time. A histogram was plotted from the factoring algorithm's counts with and without decoherence as shown in the figures below:

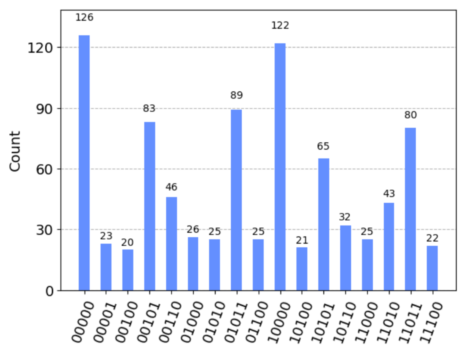


Figure 1: Histogram with Decoherence

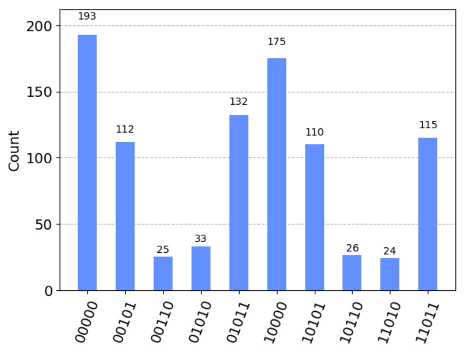


Figure 2: Histogram without Decoherence

A threshold of over 20 counts to remove redundant data due to the non-perfect division of 512 by 6. When comparing both figures, we can see 6 peaks for both plots that have significantly more counts than the rest. These represent the which would lead potentially result in the correct factors. However, on average, the plot containing decoherence has a lower count rate for the 6 main peaks than the plot without decoherence. This is due to the fact that the noise generated via decoherence spreads out the data over more possible phases, which implies that the factoring algorithm would, on average, need to run more times to reach the correct factors than if decoherence was not implemented. In a real quantum computer, decoherence is almost inevitable. AQFT is preferred in certain cases because it limits the amount of decoherence added in the system compared to QFT, which may lead to greater computational efficiency.

5 Quantum Addition

5.1 A basic addition circuit

In quantum computing, addition algorithms are typically 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. Such an algorithm makes use of the QFT and therefore the AQFT can also be implemented, as will be shown later.

A typical circuit for performing addition uses $3n$ qubits. Let us delve deeper into the methodology for addition $\mod (N)$. There are two reversible gates required: SUM and CARRY (shown in Figure 3).

The SUM subroutine adds the first two qubits to the third qubit in the following manner:

$$|i, j, k\rangle \Rightarrow |i \oplus j \oplus k\rangle \tag{35}$$

It is unitary and also the inverse of itself.

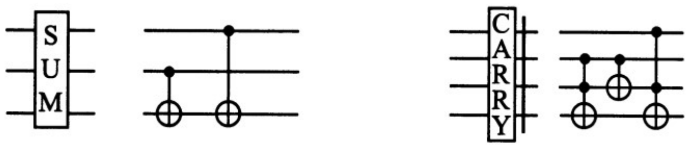


Figure 3: SUM and CARRY subroutines.

CARRY is a subroutine also used in the addition circuit and is usually read left to right. Given the four input qubits are called c, a, b, d respectively, then CARRY operates in the following way.

$$|c, a, b, d\rangle \Rightarrow |c, a, a \oplus b, d \oplus ab \oplus ac \oplus bc\rangle \tag{36}$$

However, for the addition, we will also have to perform CARRY in the opposite direction, from right to left. Thus, the position of the vertical bar in the diagram is put on the other side. We call this function RCARRY which works as:

$$|u, x, y, v\rangle \Rightarrow |u, x, v \oplus uy \oplus xy \oplus x\rangle \tag{37}$$

This is the inverse of the CARRY function; both the CARRY and RCARRY operators are unitary.
For performing the addition of two 3-bit numbers, the circuit in Figure 4 is employed. We have here as input $a_3a_2a_1$ and $b_3b_2b_1$ and the $|0\rangle$ as carry qubits. On the output end we get $a_3a_2a_1$ and $(a + b)$. The carry qubits start out as $|0\rangle$ and are also $|0\rangle$ after the circuit has been run. The reason they are zero is because the they serve as carry qubits during the operation and are disentangled from the a and b qubits. At the end of the third CARRY, the carry qubits 'carry' appropriate information, the a_i qubits are unchanged and the b_i qubits are in the state $(a_i \oplus b_i)$. Then, after applying XOR restores b_3 to its original state so that SUM can subsequently applies the $\mod (2)$ sum onto a_i and b_i and store the result in the b_i qubits. Having then used the carried information, the reverse carry returns the qubit back to the input state $|0\rangle$ without affecting the other qubits, such that the next significant sum bit can be put into the b_i qubits. Note that the last carry qubit is not restored to its original state but is not a new state of $|(a + b)_4\rangle$. Because the operators we have used are invertible, the adder circuit is completely invertible. Moreover, the adder can be run backwards to perform an operation that is somewhat a subtractor [9]. Because we are using carry qubits, and also have a and b as input qubits, this circuit uses $3n$ qubits in total.

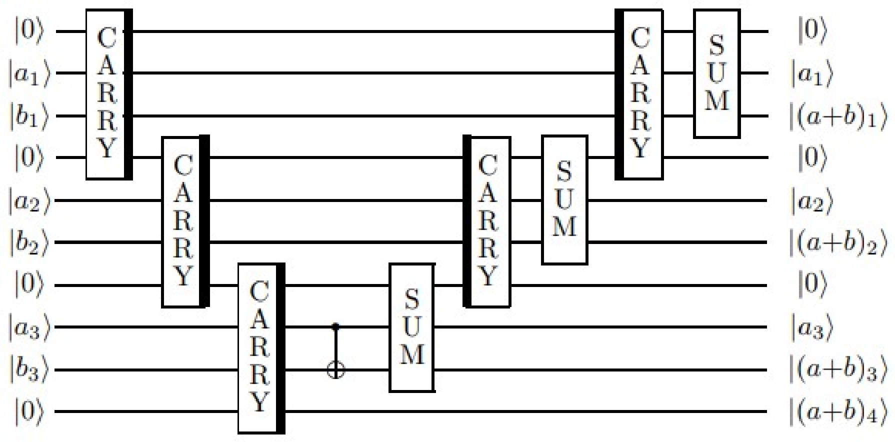


Figure 4: Circuit for addition.

5.2 Quantum Addition using QFT and thus AQFT

We now move on to finding ways to reduce the number of qubits used for the addition. The circuit in Figure 5 uses the QFT to implement addition [10]. It starts by performing the QFT on a_i. A series of rotators that are dependent on b_i being in the state $|1\rangle$ are applied onto the transformed a_i states, which we will call $|\phi(a)\rangle$. To add a phase shift to the complex exponentials in $|\phi(a)\rangle$, a circuit similar to the QFT is implemented, which uses a number of controlled phase gates, but no Hadamard gates. So $|\phi_{n-1}(a)\rangle$ is transformed to $|\phi_{n-1}(a+b)\rangle$ similarly to $|\phi(a)\rangle$. This is repeated a total of n times. Lastly, we apply the inverse QFT to $|\phi(a+b\rangle$ to acquire the sum $a+b$.

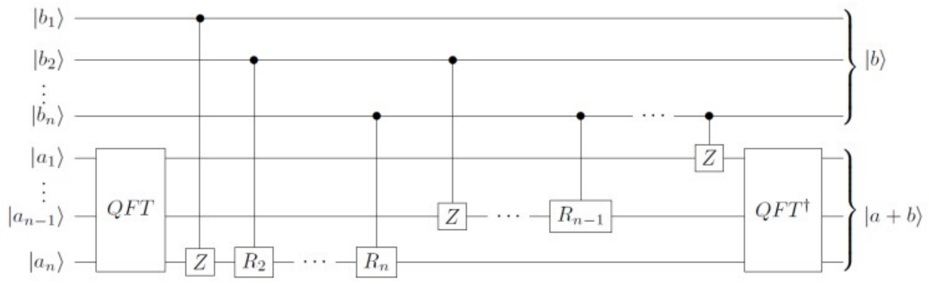


Figure 5: Adder circuit that uses the QFT.

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.

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. If we replace the QFT and inverse QFT in Figure 5 and the number of bits we have is compatible with the accuracy of the AQFT, then we can create an adder circuit that is significantly more efficient.

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 eliminates rotations below a certain threshold, quantum addition can be completed in $\log(2n)$ time intervals [3].

5.3 Implementation and Algorithms

Let us now come to actually simulating the algorithm and getting results that are significant. To do this, we use the Qiskit library and use functions to simulate a Hadamard gate. This Hadamard gate is then run on all the qubits in a given input register; the number of times the Hadamard is applied is parameterized used the AQFT. A simplified version of this is what we have tried to implement in the addition part of our code. This code can be run for different inputs and provides a corresponding output. A more rigorous implementation of the circuit and algorithm in Figure 5 is given in [10], with the code being in C. Although this does not use the Qiskit library, which is only available in Python, it is a far more beautiful implementation. However, by making use of the ease that Qiskit functions provide when simulating quantum circuits, multiple adder circuits have been written in [8], one of which implements the AQFT circuit as well.

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, quant-ph/0201067v1.

[3] Thomas G. Draper, ”Addition on a Quantum Computer”, arXiv, 2000, quant-ph/0008033.

[4] Beauregard, Stephane. ”Circuit for Shor's algorithm using 2n+ 3 qubits.” arXiv preprint quant-ph/0205095 (2002).

[5] Gamel, Omar, and Daniel FV James. ”Simplified Factoring Algorithms for Validating Small-Scale Quantum Information Processing Technologies.” arXiv preprint arXiv:1310.6446 (2013).

[6] Chuang, Isaac, et al. ”Quantum computers, factoring, and decoherence”. Science, vol. 270, no. 5242, pp. 1633-1635, 1995.

[7] J. I. Cirac and P. Zoller, “Quantum computations with cold trapped ions,” Physical Review Letters, vol. 74.20, no. 4091, 1995.

[8] Suau, Adrien, CERFACS, GitHub Repository, 2018, `https://github.com/nelimee/quantum-tools/blob/b9d19c6a1bdcb4e30316f152d9c441fb63ace6ec/gates/add.py`

[9] A. Pittenger, “An introduction to quantum computing algorithms”, Birkhauser: 2000.

[10] Van der Lans, Mike, ”Quantum Algorithms and their Implementation on Quantum Computer Simulators”, Thesis, Delft University of Technology, `http://resolver.tudelft.nl/uuid:f05164dd-b853-41a1-a9e9-394cb7a1105e` (2018)