National Institute of Science Education & Research

Quantum Fourier Transform and Quantum Phase Estimation

Quantum Information Quantum Computation (P471) Report

School of Physical Sciences

Done by :
Varun Subudhi (201188)
Mrinali Mohanty (2011091)
Hudha P M (2011071)

Under the Guidance of :
Dr. Anamitra Mukherjee
Associate Professor, School of Physical Sciences, NISER

Abstract

Quantum Fourier Transform (QFT) and Quantum Phase Estimation (QPE) are fundamental algorithms in quantum computing, providing powerful tools for analyzing and manipulating quantum states. In this report, we explore the theory and implementation of these transformative techniques using Qiskit, a leading open-source quantum computing framework. Beginning with an investigation into Gaussian state preparation, we discretize a Gaussian function and encode it into a quantum state, followed by the application of the QFT to obtain its Fourier transform. We address challenges such as gate availability and measurement integration, ensuring a seamless implementation. Furthermore, we extend our inquiry to Quantum Phase Estimation, a pivotal algorithm for determining eigenvalues of unitary operators. Through Qiskit, we demonstrate the practical application of QPE in quantum computing, showcasing its significance in quantum algorithms. By combining theoretical insights with hands-on implementation, this report provides a comprehensive understanding of QFT and QPE, along with their real-world implications in quantum computing.

Keywords : Quantum Fourier Transform (QFT), Quantum Phase Estimation (QPE), Quantum computing, Qiskit, Gaussian state preparation, Discretization, Fourier transform, Gate availability, Measurement integration, Unitary operators, Eigenvalues, Quantum algorithms.

Contents

I Introduction

Quantum Fourier Transform (QFT) and Quantum Phase Estimation (QPE) are indispensable components of quantum computing, offering unprecedented capabilities that classical algorithms cannot match. One of the key reasons we need these algorithms is their remarkable efficiency compared to classical counterparts. QFT, for instance, lies at the heart of Shor's quantum factoring algorithm, enabling exponential speedup in factoring large integers – a task that poses a significant challenge for classical computers. While classical factoring algorithms, such as the general number field sieve, have a worst-case time complexity of $O(exp(\frac{64}{9})^{1/3}(logN)^{1/3}(loglogN)^{2/3})$ [1], Shor's algorithm with QFT achieves polynomial time complexity, specifically $O((logN)^3)$ for an N-bit integer.

Similarly, QPE provides an efficient means of estimating the eigenvalues of unitary operators, a fundamental problem in quantum computing, allowing for exponential speedup in various quantum algorithms. Classical methods for eigenvalue estimation, such as iterative algorithms like the power method or QR iteration, typically have a worst-case time complexity of $O(n^3)$, where n is the dimension of the matrix. In contrast, QPE offers an exponential speedup by leveraging quantum parallelism and interference effects. By encoding the eigenvalue problem into a quantum algorithm, QPE can efficiently estimate eigenvalues with high precision, achieving polynomial time complexity, specifically $O(poly(logN))$ for an N-bit precision.

Moreover, QFT and QPE leverage unique quantum properties such as superposition and entanglement to achieve computational advantages over classical algorithms. These properties enable quantum computers to perform calculations in parallel and exploit interference effects, leading to exponential speedup for certain tasks. The significance of QFT and QPE extends beyond theoretical considerations, as they find wide-ranging applications in quantum computing and related fields. QPE, for instance, is employed in quantum chemistry for simulating molecular structures and reactions, in quantum cryptography for developing secure communication protocols, and in quantum machine learning for optimizing algorithms and solving optimization problems. Similarly, QFT serves as a foundational building block for many other quantum algorithms, demonstrating its versatility and importance in the quantum computing landscape.

1 History

The Quantum Fourier Transform (QFT) and Quantum Phase Estimation (QPE) algorithms have been instrumental in the development of quantum computing. In 1984, physicist Peter Shor introduced the concept of QFT in his groundbreaking paper "Algorithms for Quantum Computation: Discrete Logarithms and Factoring." This work laid the foundation for understanding how quantum computers could tackle complex mathematical problems, such as integer factorization, which are difficult for classical computers. Shor's subsequent quantum factoring algorithm, published in 1994, demonstrated the critical role of QFT in quantum algorithms, particularly in the context of cryptography and security.

Around the same time, in 1995, Lov Grover proposed QPE in his paper "Quantum

Mechanics Helps in Searching for a Needle in a Haystack." QPE emerged as a powerful algorithm for estimating the eigenvalues of unitary operators, a fundamental problem in quantum computing. Over the years, QPE has found applications in various fields, including quantum simulation, chemistry, and cryptography.

Throughout the late 1990s and early 2000s, further research refined the theoretical underpinnings of QFT and QPE, solidifying their status as fundamental quantum algorithms. Cleve, Ekert, Macchiavello, and Mosca's 1999 paper "Quantum algorithms revisited" provided a comprehensive analysis of QPE, contributing to its widespread recognition in the quantum computing community. Additionally, advancements in quantum computing frameworks, such as the development of Qiskit, have facilitated the practical implementation and experimentation of QFT and QPE.

As quantum computing continues to progress, QFT and QPE remain essential components of quantum algorithms, offering unique capabilities for solving complex problems efficiently. Their historical significance underscores the transformative potential of quantum computing in addressing real-world challenges across diverse domains.

II Theory and Working Principles

1 What is Fourier Transform?

The Fourier transform is a fundamental mathematical tool used in various fields such as signal processing, engineering, physics, and mathematics. Named after the French mathematician Joseph Fourier, it provides a way to analyze functions and signals in terms of their frequency components. This transformative technique plays a pivotal role in understanding the behavior of complex systems and extracting meaningful information from data.
At its core, the Fourier transform decomposes a function or signal into sinusoidal components of different frequencies. Mathematically, it expresses a function $f(t)$ in terms of its frequency spectrum $F(\omega)$, where ω represents the angular frequency. This transformation is accomplished using complex exponential functions $e^{i\omega t}$, which oscillate at various frequencies. By integrating the product of the signal and these complex exponentials over all time, the Fourier transform yields the amplitude and phase of each frequency component present in the signal.

1.1 Types of Fourier Transforms:

There are several variations of the Fourier transform tailored to different contexts:

- **Continuous Fourier Transform (CFT):** This is the classical form of the Fourier transform, applicable to continuous, time-domain signals. It converts a function of time into a function of frequency.
- **Discrete Fourier Transform (DFT):** The DFT is used for discrete, sampled signals. It computes the frequency spectrum of a finite sequence of data points. The Fast Fourier Transform (FFT) algorithm efficiently computes the DFT, making it widely used in digital signal processing.

- **Fourier Series:** Fourier series represents periodic functions as a sum of sine and cosine functions with different frequencies. It is particularly useful for analyzing periodic phenomena.
- **Short-Time Fourier Transform (STFT):** STFT provides a time-varying frequency analysis of a signal by computing the Fourier transform over short, overlapping windows. This is useful for analyzing signals that vary in frequency over time, such as non-stationary signals.

1.2 Applications:

The applications of Fourier transforms are extensive and diverse:

- **Signal Processing:** Fourier analysis is indispensable in signal processing tasks such as filtering, modulation, and spectral analysis. It allows engineers to extract relevant information from signals and manipulate them effectively.
- **Communications:** In telecommunications, Fourier transforms are used in modulating and demodulating signals, as well as in coding and decoding schemes.
- **Image Processing:** Fourier transforms play a crucial role in image analysis and manipulation. They are used in tasks such as image compression, enhancement, and feature extraction.
- **Physics and Engineering:** Fourier analysis is widely applied in physics and engineering disciplines for solving differential equations, analyzing vibrations and oscillations, and understanding the behavior of dynamical systems.
- **Mathematics:** Fourier transforms have deep connections to various areas of mathematics, including harmonic analysis, functional analysis, and partial differential equations.

Fourier transforms are a versatile and powerful mathematical tool with a broad range of applications. Their ability to decompose signals into their frequency components provides valuable insights and enables sophisticated analysis and manipulation of data. From signal processing and communications to image analysis and beyond, Fourier transforms continue to be essential in advancing scientific and technological frontiers. Their impact extends far beyond their mathematical origins, shaping our understanding of the world and driving innovation across multiple disciplines.

1.3 Implementation of Fourier transforms

Continuous Fourier Transform :
The CFT represents a signal as a continuous spectrum of frequencies, obtained by integrating the signal against complex exponential functions. Mathematically, the CFT is defined as an integral over all time of the signal multiplied by a complex exponential:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \tag{1}$$

While the CFT provides a powerful theoretical framework, its computational implementation is limited due to the need for integration over an infinite domain.

Discrete Fourier Transform :
This brings us to DFT which is a discrete version of the CFT, developed to analyze finite-length discrete signals. It's defined by summing up the product of signal samples with complex exponential functions at discrete frequency points as:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-i2\pi kn/N} dt \tag{2}$$

The DFT provides a way to perform Fourier analysis on digital signals, making it suitable for computation on computers. but has a time complexity of $O(N^2)$, where N is the number of samples in the signal, making it impractical for large datasets.
We can take an example of DFT by considering a set of N time values [2]:

$$t_k = \frac{kT}{N}, \text{ where, } k = 0, 1.., N-1$$

The reciprocal space, ω space, can be represented by:

$$\omega_p = \frac{2\pi p}{T}, \text{ where, } p == 0, 1.., N-1$$

So, the function of time in discrete space undergoes Fourier transform to give the function of frequency as shown:

$$F(\omega_p) = \frac{1}{N} \sum_{0}^{N-1} f(t_k) e^{i\omega_p t_k} \tag{3}$$

And the inverse Fourier transform is done by:

$$f(t_k) = \sum_{0}^{N-1} g(\omega_p) e^{-i\omega_p t_k} \tag{4}$$

$f(t_k)$ and $F(\omega_p)$ are Fourier transforms of each other.

Fast Fourier Transform :
The FFT is an algorithmic approach introduced to compute the DFT more efficiently. The Cooley-Tukey algorithm, published in 1965, is one of the most famous FFT algorithms. It employs a divide-and-conquer strategy, recursively breaking down the DFT calculation into smaller sub-problems. By exploiting symmetries and redundancies in the computation of the DFT, the FFT reduces the number of arithmetic operations from $O(N^2)$ to $O(NlogN)$, where N is the number of samples. The FFT revolutionized signal processing and made real-time analysis feasible for a wide range of applications, including audio processing, image processing, and telecommunications.

The Cooley-Tukey FFT is just one of many FFT algorithms. Various other FFT algorithms have been developed, each optimized for different scenarios. Radix-2 FFT algorithms are well-suited for signals with a length that is a power of 2, while mixed-radix

FFT algorithms handle arbitrary lengths efficiently. Prime factor FFT algorithms are designed to efficiently compute the DFT for lengths that are prime numbers or products of small prime factors. These algorithms incorporate optimizations such as butterfly operations, twiddle factor reordering, and memory access patterns to further enhance efficiency.

FFT algorithms have been extensively optimized and parallelized for various hardware architectures, including CPUs, GPUs, FPGAs, and DSPs. Highly optimized FFT libraries, such as FFTW (Fastest Fourier Transform in the West), provide efficient implementations for a wide range of platforms. These implementations leverage platform-specific optimizations, such as SIMD instructions, multithreading, and GPU parallelism, to achieve high performance.

1.4 Fourier Transform of a Sample Gaussian Function

A Gaussian function, also known as a Gaussian distribution or bell curve, is a mathematical function that represents a probability distribution. It is characterized by its bell-shaped curve, with a peak at the mean value and symmetrical tails on either side. They are widely utilized across disciplines for their efficacy in modeling real-world data distributions, owing to their simplicity and versatility. A Gaussian function has the general form as the following:

$$f(t) = \alpha e^{-(t-\mu)^2/2\sigma^2} \tag{5}$$

where α, μ, and σ are arbitrary constants representing the height of the curve's peak, the peak's position (mean/average value), and the distribution's width (standard deviation, which measures the spread of the distribution), respectively.
From the above equation, its Fourier transform is:

$$g(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{i\omega t} dt = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \alpha e^{-(t-\mu)^2/2\sigma^2} e^{i\omega t} dt \tag{6}$$

$$\text{Let } \frac{t-\mu}{\sqrt{2}\sigma} = y \implies dy = \frac{dx}{\sqrt{2}\sigma}$$

$$g(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \alpha e^{-y^2} e^{i\omega(\sqrt{2}y\sigma+\mu)} dy\sqrt{2}\sigma$$

$$g(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \alpha e^{-(y^2 - i\omega\sqrt{2}y\sigma)} e^{i\omega\mu} dy\sqrt{2}\sigma$$

$$\therefore y^2 - \frac{i\omega y\sigma}{\sqrt{2}} = y^2 - \frac{i\omega y\sigma}{\sqrt{2}} - \frac{\omega^2\sigma^2}{2} + \frac{\omega^2\sigma^2}{2} = \left(y - \frac{i\omega\sigma}{\sqrt{2}}\right)^2$$

$$g(\omega) = \frac{\sqrt{2}\sigma\alpha}{\sqrt{2\pi}} e^{i\omega\mu} e^{-\omega^2\sigma^2/2} \int_{-\infty}^{\infty} e^{-(y - i\omega y\sigma/\sqrt{2})^2} dy$$

$$\text{Let } y - \frac{i\omega y\sigma}{\sqrt{2}} = z \implies dy = dz$$

$$g(\omega) = \frac{\sigma\alpha}{\sqrt{\pi}} e^{i\omega\mu} e^{-\omega^2\sigma^2/2} \int_{-\infty}^{\infty} e^{-z^2} dz$$
$$g(\omega) = \frac{\sigma\alpha}{\sqrt{\pi}} e^{i\omega\mu} e^{-\omega^2\sigma^2/2} \sqrt{\pi}$$
$$g(\omega) = \sigma\alpha e^{i\omega\mu} e^{-\omega^2\sigma^2/2}$$

The Fourier transform of a Gaussian function yields another Gaussian function, with its peak positioned at the origin in the frequency domain. An oscillatory phase factor $e^{i\omega\mu}$ signifies the peak's displacement from the origin in position space. The pulse width (standard deviation) of the transformed Gaussian is inversely related to the pulse width of the original Gaussian. The amplitude of the transformed Gaussian is adjusted by a scaling factor compared to the original Gaussian, maintaining the preservation of the area under the curve during the transformation.

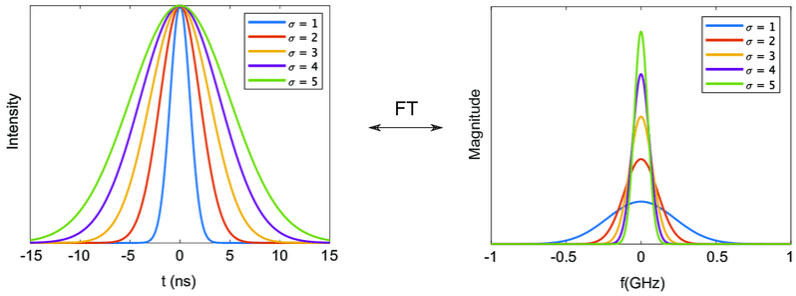

Figure II.1: Fourier transforms of various Gaussian distributions frequencies of varying widths. [3]

2 Quantum Fourier Transform (QFT)

The Fourier transform occurs in many different versions throughout classical computing, in areas ranging from signal processing to data compression to complexity theory. It operates on classical bits, utilizing arithmetic and logical operations to transform input data from the time domain to the frequency domain.

Quantum computing utilizes quantum bits, or qubits, which can exist in superpositions of states and be entangled with each other. Quantum algorithms leverage principles such as superposition, interference, and entanglement to perform computations in ways that classical computers cannot replicate efficiently.

The Quantum Fourier Transform is a quantum analog of the classical Fourier Transform, designed to operate on quantum state. It has advantages over the Fast Fourier Transform (FFT) in certain situations because it operates on qubits, which can hold multiple states simultaneously (superposition), enabling quantum computers to perform numerous computations in parallel and potentially achieve exponential speedup for certain problems. Additionally, qubits can be entangled, allowing for highly correlated operations across distant qubits. Quantum algorithms leverage superposition to process vast amounts of data simultaneously, providing computational advantages over classical approaches. Moreover, the state space of a quantum computer grows exponentially with the

number of qubits, offering the potential for exponentially greater computational power compared to classical systems.

The QFT transforms a quantum state representing amplitudes of different basis states into a state representing the frequencies of those basis states. Mathematically, the QFT is defined by a unitary transformation that maps the amplitudes of the input quantum state to the Fourier amplitudes of the output state[4].

Implementing the QFT on quantum hardware poses significant challenges due to requirements such as maintaining coherence, mitigating errors, and orchestrating quantum gates. Researchers have developed various techniques to implement the QFT efficiently on different types of quantum hardware, including superconducting qubits, trapped ions, and photonic systems. Quantum computing platforms and programming languages, such as Qiskit, Quirk, and Cirq, provide tools and frameworks for simulating and executing quantum algorithms, including the QFT.

2.1 Mathematical Description

The discrete Fourier transform (DFT) acts on a vector $x_0, ..., x_{N-1}$ and maps it to the vector $y_0, ..., y_{N-1}$ according to the formula [5]:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i \frac{jk}{N}} \tag{7}$$

We note here that only the amplitudes of the state were affected by this transformation. Similarly, the quantum Fourier transform acts on a quantum state $|X\rangle = \sum_{j=0}^{N-1} x_j |j\rangle$ and maps it to the quantum state $|Y\rangle = \sum_{k=0}^{N-1} y_k |k\rangle$ according to the above formula which can also be expressed as the map:

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i \frac{jk}{N}} |k\rangle \tag{8}$$

which can be represented by the unitary matrix:

$$U_{QFT} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} e^{2\pi i \frac{jk}{N}} |k\rangle\langle j| \tag{9}$$

Now, we know the quantum Fourier transform (QFT) transforms between two bases, the computational (Z) basis and the Fourier basis. The H-gate is the single-qubit QFT, and it transforms between the Z-basis states $|0\rangle$ and $|1\rangle$ to the X-basis states $|+\rangle$ and $|-\rangle$. In the same way, all multi-qubit states in the computational basis have corresponding states in the Fourier basis. The QFT is simply the function that transforms between these bases.

$$|\text{Computational Basis States}\rangle \rightarrow |\text{Fourier Basis States}\rangle$$

$$\text{QFT}|x\rangle = |\tilde{x}\rangle$$

The tilde ($\sim$) is used to denote the states in Fourier basis.

2.2 QFT for N-qubits

Now, let's see how the quantum Fourier transform looks like for larger N. Let's derive a transformation for $N = 2^n$, QFT_N acting on the state $|x\rangle = |x_1...x_n\rangle$ where x_1 is the most significant bit [6].

$$QFT_N|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i xy/2^n}|y\rangle \text{ where, } N = 2^n$$
$$= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i \sum_{k=1}^{n} y_k x/2^n}|y_1...y_n\rangle$$

rewriting in fractional binary notation $y = y_1...y_n/2^n = \sum_{k=1}^{n} y_k/2^k$

$$QFT_N|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \Pi_{k=1}^{n} e^{2\pi i y_k x/2^n}|y_1...y_n\rangle$$

by expanding the exponential of a sum to a product of exponentials.

$$QFT_N|x\rangle = \frac{1}{\sqrt{N}} \bigotimes_{k=1}^{n} (|0\rangle + e^{2\pi i x/2^k}|1\rangle)$$

after rearranging the sum and products, and expanding $\sum_{y=0}^{N-1} = \sum_{y_1=0}^{1} \cdots \sum_{y_n=0}^{1}$

$$QFT_N|x\rangle = \frac{1}{\sqrt{N}}(|0\rangle + e^{2\pi i x/2}|1\rangle) \otimes (|0\rangle + e^{2\pi i x/2^2}|1\rangle) \otimes ... \otimes (|0\rangle + e^{2\pi i x/2^{n-1}}|1\rangle) \otimes (|0\rangle + e^{2\pi i x/2^n}|1\rangle)$$

2.3 Form of QFT Circuit

The example above demonstrates a very useful form of the QFT for $N = 2^n$ Note that only the last qubit depends on the values of all the other input qubits and each further bit depends less and less on the input qubits. This becomes important in physical implementations of the QFT, where nearest-neighbor couplings are easier to achieve than distant couplings between qubits.

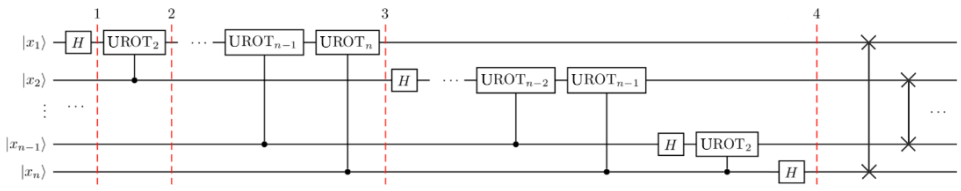

Figure II.2: A Circuit that implements an n-qubit quantum Fourier transform. [6]

Additionally, as the QFT circuit becomes large, an increasing amount of time is spent doing increasingly slight rotations. It turns out that we can ignore rotations below a certain threshold and still get decent results, which is known as the approximate QFT. This is also important in physical implementations, as reducing the number of operations can greatly reduce decoherence and potential gate errors.

3 Quantum Phase Estimation (QPE)

Quantum Phase Estimation (QPE) is a quantum algorithm that aims to estimate the eigenvalues of unitary operators, which represent the phases acquired by quantum states under the action of these operators[4].

3.1 Working Principle

1. Phase estimation typically starts with an eigenvector of a unitary operator U and a corresponding eigenvalue. Our algorithm takes a unitary operator U and an eigenvector $|\psi\rangle$ as input, where U acts on $|\psi\rangle$ as $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$, where θ is the eigenvalue to be estimated. $|\psi\rangle$ is in one set of qubit registers. An additional set of n qubits form the counting register on which we will store the value $2^n\theta$. [5]

$$|\psi_0\rangle = |0\rangle^{\otimes n}|\psi\rangle$$

Now, we apply a n-bit Hadamard gate operation $H^{\otimes n}$ on the counting register.

$$|\psi_1\rangle = \frac{1}{2^{n/2}}(|0\rangle + |1\rangle)^{\otimes n}|\psi\rangle$$

2. A quantum circuit is constructed that prepares an ancillary qubit in a superposition of states, representing different estimates of the phase θ. This is achieved by applying a series of controlled-unitary operations, with the number of controls increasing in each step.

$$U^{2^j}|\psi\rangle = U^{2^j-1}U|\psi\rangle = U^{2^j-1}e^{2\pi i\theta}|\psi\rangle = ... = e^{2\pi 2^j\theta}|\psi\rangle$$

Applying all the n controlled operations CU^{2^j} with $0 \leqslant j \leqslant n-1$, and using the relation $|0\rangle\otimes|\psi\rangle + |1\rangle\otimes e^{2\pi i\theta}|\psi\rangle = (|0\rangle + e^{2\pi i\theta}|1\rangle)\otimes|\psi\rangle$,

$$\begin{aligned}|\psi_2\rangle &= \frac{1}{2^{n/2}}\left(|0\rangle + e^{2\pi i\theta 2^{n-1}}|1\rangle\right)\otimes ...\otimes\left(|0\rangle + e^{2\pi i\theta 2^1}|1\rangle\right)\otimes\left(|0\rangle + e^{2\pi i\theta 2^0}|1\rangle\right)\\ &= \frac{1}{2^{n/2}}\sum_{k=0}^{2^n-1}|k\rangle\otimes|\psi\rangle\end{aligned}$$

where k denotes the integer representation of n-bit binary numbers.

3. The phase θ is encoded into the state of the ancillary qubit through a phenomenon known as phase kickback. This happens when the eigenvalue θ "kicks back" to the ancillary qubit during the controlled-unitary operations.

4. QPE applies an inverse Quantum Fourier Transform to the ancillary qubits. This transforms the phase-encoded state into a superposition of computational basis states, where the probability amplitudes correspond to estimates of the phase θ.

$$QFT_N|x\rangle = \frac{1}{\sqrt{N}}(|0\rangle+e^{2\pi ix/2}|1\rangle)\otimes(|0\rangle+e^{2\pi ix/2^2}|1\rangle)\otimes...\otimes(|0\rangle+e^{2\pi ix/2^{n-1}}|1\rangle)\otimes(|0\rangle+e^{2\pi ix/2^n}|1\rangle)$$

Replacing x by $2^n\theta$ in the above expression gives exactly the expression derived in step 2 above. Therefore, to recover the state $|2^n\theta\rangle$, apply an inverse Fourier transform on the auxiliary register. Doing so, we find:

$$|\psi_3\rangle = \frac{1}{2^{n/2}}\sum_{k=0}^{2^n-1}e^{2\pi i\theta k}|k\rangle\otimes|\psi\rangle \xrightarrow{QFT_n^{-1}} \frac{1}{2^n}\sum_{x,k=0}^{2^n-1}e^{-2\pi ik(x-2^n\theta)/N}|x\rangle\otimes|\psi\rangle$$

5. Finally, the ancillary qubits representing the phase information are measured, yielding an estimate of the phase θ with high probability. The above expression peaks near $x = 2^n\theta$. For the case when $2^n\theta$ is an integer, measuring in the computational basis gives the phase in the auxiliary register with high probability:

$$|\psi_4\rangle = |2^n\theta\rangle \otimes |\psi\rangle$$

The measurement outcomes provide an estimate of the phase, typically in the form of a binary fraction.

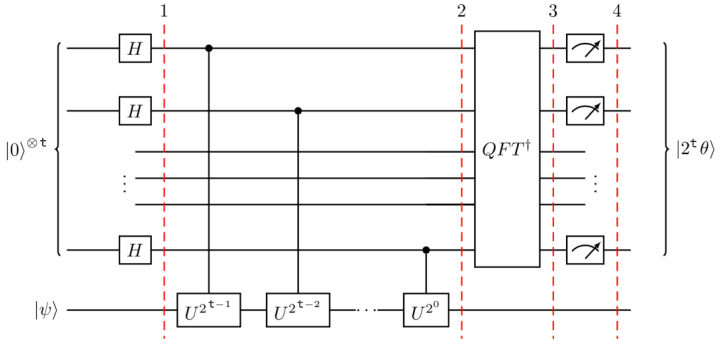

Figure II.3: A general quantum circuit for quantum phase estimation with the top register contains t 'counting' qubits, and the bottom contains qubits in the state $|\psi\rangle$ [5]

Quantum Phase Estimation (QPE) is a crucial algorithm in quantum computing with diverse applications. In Shor's algorithm, QPE efficiently determines the period of a function, facilitating the factorization of large composite numbers, a task considered challenging for classical computers. Additionally, QPE finds utility in quantum chemistry, where it estimates the energies of molecular systems, aiding research in quantum chemistry and materials science. Furthermore, QPE enables quantum simulation by estimating the eigenvalues of Hamiltonian operators, allowing for the study of complex quantum phenomena and the simulation of quantum systems' time evolution.

III Numerical Procedures and Computational Setup

1 Basics of Quantum Computing

Qubits or quantum bits are the fundamental building blocks of quantum computing. Unlike classical bits, which can exist in one of two states (0 or 1), qubits can exist as a superposition of states $|0\rangle$ and $|1\rangle$, where these states can be represented in the matrix form as:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}; |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix};$$

Quantum gates are the basic building blocks of quantum circuits, (similar to classical logic gates in classical digital circuits) which operate on qubits, the quantum equivalent of classical bits, and manipulate them according to the laws of quantum mechanics. These gates perform various operations on qubits, such as changing their state, creating superpositions, and entangling qubits. Some common types of quantum gates which are relevant to our topic are listed as below:

- Hadamard gate: It converts $|0\rangle$ and $|1\rangle$ states to $|+\rangle$ and $|-\rangle$ states, which are equal superposition of both $|0\rangle$ and $|1\rangle$ states.

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

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle \text{ and } H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle$$

- X-gate: This is also known as the Pauli-X gate, which rotates the state by 180^0 about the x-axis of the Bloch sphere.

$$X = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$$

$$X|0\rangle = |1\rangle \text{ and } X|1\rangle = |0\rangle$$

- $UROT_k$ Gate: It is also known as the Uniformly Controlled R(θ) gate, which performs a rotation by an angle θ around the Z-axis for each qubit in a quantum register, controlled by another set of qubits.

$$U = \begin{pmatrix}1 & 0\\ 0 & e^{2\pi i/2^k}\end{pmatrix}$$

$$UROT_k|0\rangle = |0\rangle \text{ and } UROT_k|1\rangle = e^{2\pi i/2^k}|1\rangle$$

2 Qiskit Implementation

2.1 Qiskit

Qiskit is an open-source quantum computing software development framework created by IBM. It provides tools for working with quantum circuits, algorithms, and simulators, as well as access to real quantum hardware through the IBM Quantum Experience platform. It allows users to design quantum circuits using a high-level programming language based on Python. Users can specify quantum gates, qubit operations, and measurements to create quantum algorithms. It provides built-in simulators that allow users to simulate the behavior of quantum circuits with tools for visualizing quantum circuits, including circuit diagrams and state vectors on classical computers. These simulators are useful for testing and debugging quantum algorithms before running them on real quantum hardware. Through the IBM Quantum Experience platform, users can access real quantum hardware provided by IBM. This includes a variety of quantum processors with different numbers of qubits and error rates.

3 Execution of Algorithms

(Circuits with their Executions and Outputs can be accessed in the GitHub links.)
Link to the Repository:
`https://github.com/varun-subudhi/QIQC-Project-P471`

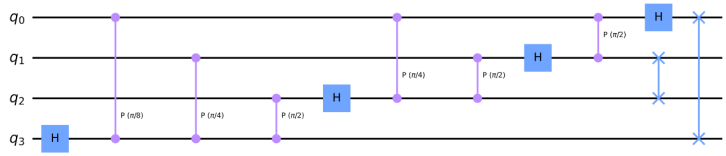

Figure III.1: A general quantum circuit for quantum Fourier transform using 4 qubits.

3.1 Quantum Fourier Transform (QFT)

`https://github.com/varun-subudhi/QIQC-Project-P471/blob/main/QFT_Implementation.ipynb`

3.2 Quantum Phase Estimation (QPE)

`https://github.com/varun-subudhi/QIQC-Project-P471/blob/main/QPE_Implementation.ipynb`

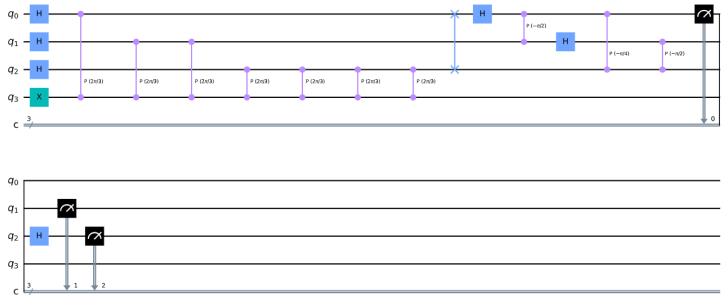

Figure III.2: A general quantum circuit for quantum phase estimation using 4 qubits.

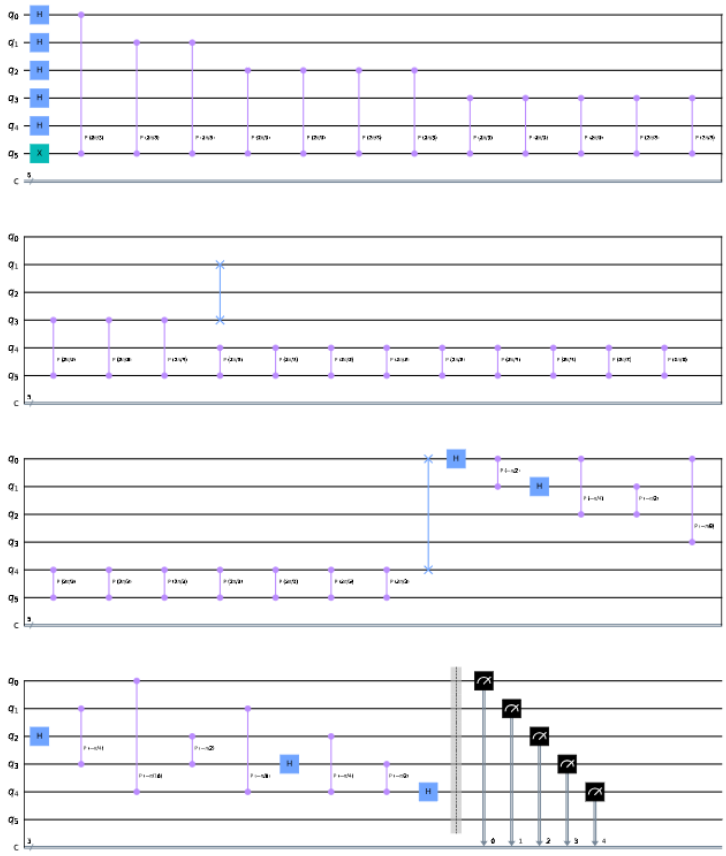

Figure III.3: A general quantum circuit for quantum phase estimation using 6 qubits.

IV Results & Discussion

- In this project, we were successful in implementing and executing the quantum Fourier transform (QFT) and quantum phase estimation (QPE) algorithms and building quantum circuits using Qiskit to simulate it. To ensure the accuracy of our circuits, we conducted comprehensive tests.

- For the QFT circuit, we provided a Gaussian function as input and observed that the output remained a Gaussian function. Remarkably, even when we altered the mean position, the peak of the Fourier-transformed Gaussian consistently remained at zero. Moreover, as we increased the standard deviation of the input Gaussian, we noted a proportional increase in the amplitude of the Fourier-transformed Gaussian. This was in agreement with the calculations done analytically.

- When an input state was given to the QPE circuit, the global phase of the state was determined from the probability distribution. As we increase the number of qubits, we observed an increased precision in phase estimation. Overall, these tests validated the accuracy of our circuits and underscored the efficacy of Qiskit for constructing and verifying quantum algorithms.

V Future Scope

- Qiskit's simulators offer a cost-effective solution for simulating quantum circuits, aiding algorithm development and testing without needing quantum hardware. These simulators enable scalable exploration of circuits with various qubits and gates, ensuring flexibility and speed for efficient experimentation in quantum computing. IBM Q provides access to real quantum hardware via the IBM Quantum Experience platform, allowing users to experiment with actual qubits and gain insights into quantum system behavior. While quantum hardware computations are slower and have limited qubit availability compared to simulations, they are crucial for validating algorithms in real-world conditions, providing valuable feedback for refinement.

- Implementing the Quantum Fourier Transform (QFT) on various functions and verifying classically allows for benchmarking the speed of calculations, providing insights into the efficiency of quantum algorithms in different scenarios. Comparing the performance of QFT implementations across different functions helps evaluate the effectiveness and applicability of quantum computing in solving specific problems, aiding in algorithm optimization and refinement.

- Exploring the quantum counterparts of transforms such as the Laplace transform and Chirp-Z transform provides an avenue for assessing their speed and effectiveness in quantum computing. By benchmarking these quantum transformations against their classical counterparts, we gain insights into their computational efficiency and applicability across various domains. This comparative analysis helps identify the strengths and limitations of quantum transforms, facilitating the optimization and adaptation of quantum algorithms for specific tasks.

- The quantum phase estimation algorithm may appear limited, since we have to know to perform the controlled- operations on our quantum computer. However, it is possible to create circuits for which we don't know, and for which learning theta can tell us something very useful setting a fundamental subroutine for the famous Shor's algorithm for identifying the period of a function relevant to integer factorization.

Acknowledgement

I would like to acknowledge and express my gratitude to Dr. Anamitra Mukherjee, our professor at NISER, Bhubaneswar, whose guidance and inputs played a crucial role throughout the project. I want to thank my project teammates Mrinali Mohanty and Hudha P M, and my friends for their insightful discussions and feedback, which greatly enriched the content of this report.

References

[1] Number field sieve. https://mathworld.wolfram.com/NumberFieldSieve.html.

[2] George Arfken. *Mathematical Methods for Physicists*. Academic Press, Inc., San Diego, third edition, 1985.

[3] Sample gaussian ft. https://www.researchgate.net/figure/Left-Several-Gaussian-functions-with-different-s-values-Right-Spectrum-magnitude-c fig3_339376506.

[4] Giuliano Benenti, Giulio Casati, Davide Rossini, and Giuliano Strini. *Principles of Quantum Computation and Information*. 2 edition.

[5] Qpe. https://www.youtube.com/watch?v=5kcoaanYyZw&ab_channel=Qiskit.

[6] Qft. https://www.youtube.com/watch?v=pq2jkfJlLmY&ab_channel=Qiskit.