**Project Report**

Quantum Information Quantum Computation (P471)

# Quantum Fourier Transform and Quantum Phase Estimation


Submitted by

**Hudha PM**

2011071

Under the Guidance of

**Dr. Anamitra Mukherjee**

Associate Professor, School of Physical Sciences

National Institute of Science Education and Research, Bhubaneswar

# Abstract

Quantum Fourier Transform (QFT) and Quantum Phase Estimation (QPE) are fundamental algorithms in quantum computing that provide powerful tools for analyzing and manipulating quantum states. This report explores the theory and implementation of these transformation techniques using Qiskit, a leading open-source quantum computing framework. Starting with the study of Gaussian state preparation, we discretize the Gaussian function and encode it into a quantum state, followed by applying QFT to obtain the Fourier transform.We extend our study to quantum phase estimation, a basic algorithm for determining the eigenvalues of unitary operators. Using Qiskit, we demonstrate a practical implementation of QPE in quantum computing and its importance in quantum algorithms. Combining theoretical insights with practical application, this report provides a comprehensive overview of QFT and QPE and their real-world implications for 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 with their remarkable efficiency compared to classical counterparts. QFT and QPE, lies at the heart of Shor's quantum factoring algorithm, enabling exponential speedup in factoring large integers. While classical factoring algorithms, such as the general number field sieve, have a worst-case time complexity of $O(exp(\frac{64}{9})^{1/3}(logN)^{1/3}(loglogN)^{2/3})$ [**?**], Shor's algorithm with QFT achieves polynomial time complexity, specifically $O((logN)^3)$ for an N-bit integer.
Quantum phase estimation (QPE) enables the estimation of the phase of an eigenvalue of a unitary operator to an arbitrary number of bits of precision which is 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. Thus these a crucial subroutine that are often called by other algorithms.

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.

The Quantum Fourier Transform (QFT) and Quantum Phase Estimation (QPE) algorithms have played pivotal roles 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 address complex mathematical problems, such as integer factorization, which pose challenges for classical computers. Shor's subsequent quantum factoring algorithm, published in 1994, showcased the crucial role of QFT in quantum algorithms, particularly in cryptography and security contexts.

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 potent algorithm for estimating the eigenvalues of unitary operators, a fundamental challenge in quantum computing. Over time, 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 foundations of QFT and QPE, cementing their status as fundamental quantum algorithms. The 1999 paper "Quantum algorithms revisited" by Cleve, Ekert, Macchiavello, and Mosca provided a comprehensive analysis of QPE, contributing to its widespread recognition within the quantum computing community. Moreover, 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 advance, QFT and QPE remain indispensable components of quantum algorithms, offering unique capabilities for efficiently solving complex problems. Their historical significance underscores the transformative potential of quantum computing in addressing real-world challenges across diverse domains.

# II Theory

## 2.1 Fourier Transform

The concept of the Fourier transform is fundamental in mathematics, physics, and engineering, providing a powerful tool to analyze functions and signals in different domains. The Fourier transform is an integral transform that takes a function as input and outputs another function representing the frequencies present in the original function. It is a complex-valued function of frequency that describes the extent to which various frequencies are present in the input function. The Fourier transform allows for the decomposition of a function into its constituent frequencies and their amplitudes, enabling the analysis of signals in the frequency domain. Used in various fields, such as signal processing, engineering, physics, and mathematics, 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 essence, the Fourier transform breaks down a function or signal into sinusoidal elements of varying frequencies. It expands upon the notion of Fourier series to encompass non-periodic functions, enabling functions to be depicted as a combination of basic sinusoids. In mathematical terms, it represents a function $f(t)$ in relation to its frequency spectrum $F(\omega)$, where $\omega$ denotes angular frequency. This transformation is achieved through complex exponential functions $e^{i\omega t}$, which oscillate at different frequencies. By integrating the signal's product with these complex exponentials across all time, the Fourier transform reveals the magnitude and phase of each frequency constituent within the signal.

### 2.1.1 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.

Fourier Transform's significance lies in its ability to transform complex problems into simpler algebraic operations gives it extensive and diverse applications thereby making it a cornerstone tool in various disciplines for analyzing data, solving equations, and understanding the fundamental properties of signals and functions. 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. In telecommunications, Fourier transforms are used in modulating and demodulating signals, as well as in coding and decoding schemes. Fourier transforms play a crucial role in image analysis and manipulation. They are used in tasks such as image compression, enhancement, and feature extraction. 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. 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.

### 2.1.2 Implementation of Fourier transforms

**Continuous Fourier Transform :**

**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:

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

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.

**Fast Fourier Transform :**

The Fast Fourier Transform (FFT) is an algorithm that efficiently computes the Discrete Fourier Transform (DFT) of a sequence or its inverse. It is a crucial tool in signal processing, converting signals from their original domain (such as time or space) to the frequency domain and vice versa. The FFT reduces the complexity of computing the DFT making it significantly faster, especially for large datasets. The FFT algorithm is based on factorizing the DFT matrix into sparse factors, enabling rapid computation of transformations. There are various FFT algorithms, including those based on different mathematical theories, from simple arithmetic to group theory and number theory, providing flexibility and efficiency in different applications

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. It has been described as "the most important numerical algorithm of our lifetime" and is considered one of the top 10 algorithms of the 20th century. The Cooley-Tukey FFT is just one of many FFT algorithms. Various other FFT algorithms have been developed, each optimized for different scenarios.

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.

### 2.1.3 Fourier Transform of a Sample Gaussian Function

A Gaussian function characterized by its bell-shaped curve with a peak at the mean value and symmetrical tails on either side is a mathematical function commonly used to represent the probability density function of a normally distributed random variable. 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 :

Gaussian function has the following form:

$$f(t) = ae^{-(t-b)^2/2c^2} \tag{2}$$

where a, b, and c are arbitrary constants representing the height of the curve's peak, the peak's position, and the distribution's width (standard deviation), respectively.
Its Fourier transform can be estimated as:

$$g(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t)e^{i\omega t}dt = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} ae^{-(t-b)^2/2c^2}e^{i\omega t}dt \tag{3}$$

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

$$g(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} a e^{-y^2} e^{i\omega(\sqrt{2}yc+b)} dy \sqrt{2}c$$

$$g(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} a e^{-(y^2 - i\omega\sqrt{2}yc)} e^{i\omega b} dy \sqrt{2}c$$

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

$$g(\omega) = \frac{\sqrt{2}ca}{\sqrt{2\pi}} e^{i\omega b} e^{-\omega^2 c^2/2} \int_{-\infty}^{\infty} e^{-(y - i\omega yc/\sqrt{2})^2} dy$$

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

$$g(\omega) = \frac{ca}{\sqrt{\pi}} e^{i\omega b} e^{-\omega^2 c^2/2} \int_{-\infty}^{\infty} e^{-z^2} dz$$

$$g(\omega) = \frac{ca}{\sqrt{\pi}} e^{i\omega b} e^{-\omega^2 c^2/2} \sqrt{\pi}$$

$$g(\omega) = cae^{i\omega b} e^{-\omega^2 c^2/2}$$

where g($\omega$)=FT(f(t))

Thus Fourier transform of a Gaussian function (f(t) yields another Gaussian function g($\omega$) 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 standard deviation of the transformed Gaussian is inversely related to the standard deviation of the original Gaussian. The amplitude of the transformed Gaussian is adjusted by a scaling factor compared to the original Gaussian, maintaining the area under the curve during the transformation. Thus the transformation showcases that the Gaussian function in the time domain is transformed into a Gaussian function in the frequency domain, with the width of the Gaussian in the frequency domain inversely proportional to the width of the Gaussian in the time domain. This property is fundamental in signal processing and engineering applications, where Gaussian functions play a significant role due to their unique properties and the simplicity of their Fourier transforms.

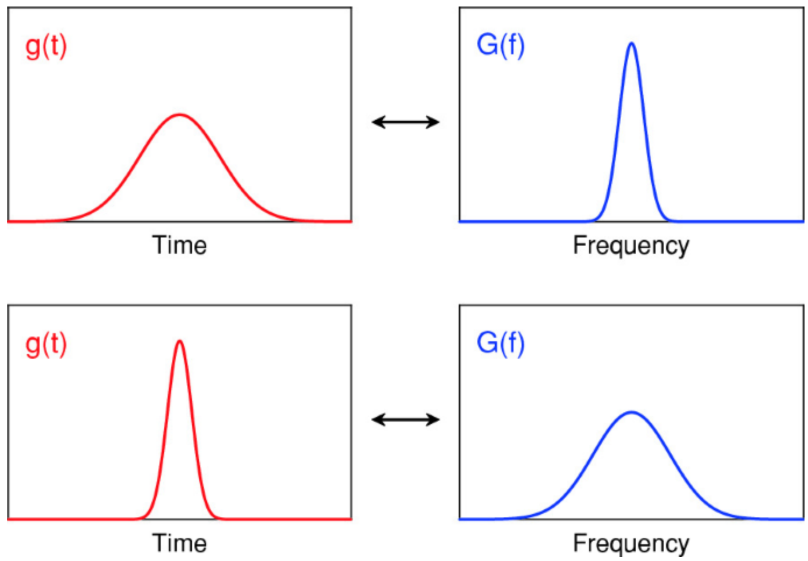


Figure 1: Fourier transform of Gaussian function [1]

## 2.2 Basics of Quantum Computing

Quantum computing leverages quantum mechanical phenomena of superposition and entanglement. The fundamental unit of information in quantum computing is the qubit (quantum bit). Unlike classical bits that can only be in a state of 0 or 1, qubits can exist in a superposition of states $|0\rangle$ and $|1\rangle$ simultaneously. Entanglement creates a strong correlation between multiple qubits. These properties enable quantum computers to process information in a fundamentally different way compared to classical computers and perform certain computations exponentially faster than classical computers for specific problems. Thus Quantum algorithms are designed to take advantage of quantum phenomena to solve problems more efficiently. The basis states $|0\rangle . |1\rangle$ 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 programming involves manipulating the quantum states of qubits to perform computations. This is done using quantum gates, which are the quantum analoge of classical logic gates 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.

In summary, quantum computing aims to harness quantum phenomena to perform computations in a fundamentally different way than classical computers, with the potential for exponential speedups for certain problems.

## 2.3 Quantum Fourier Transform (QFT)

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.

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. The quantum Fourier transform (QFT) is the quantum implementation of the discrete Fourier transform over the amplitudes of a wavefunction. It is part of many quantum algorithms, most notably Shor's factoring algorithm and quantum phase estimation.

### 2.3.1 Mathematical Description

The discrete Fourier transform acting on a vector $x_0, ..., x_{N-1}$ is according to the formula:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i \frac{jk}{N}} \tag{4}$$

Similarly, in the quantum Fourier transform, defined on a quantum register of n qubits (N = 2n), the mapping is according to :

$$|j\rangle \to \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i \frac{jk}{N}} |k\rangle \tag{5}$$

or the unitary matrix:

$$U_{QFT} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} e^{2\pi i \frac{jk}{N}} |k\rangle \langle j| \tag{6}$$

This transforms a quantum state $|X\rangle = \sum_{j=0}^{N-1} f(j)\,|j\rangle$ it to the quantum state $|Y\rangle = \sum_{k=0}^{N-1} \tilde{f}(k)\,|k\rangle$ where the coeffcients $f(k)$ are the discrete Fourier transform of the coeffcients $f(j)$.

Constructing a quantum circuit will take us from the computation basis to a the fourier basis.

$$|\text{Computational Basis States}\rangle \to |\text{Fourier Basis States}\rangle$$
$$\text{QFT}\,|x\rangle = |\tilde{x}\rangle$$

### 2.3.2 Quantum circuit for N-qubits Fourier Transform

The quantum Fourier transform for $N = 2^n$, acting on the state $|x\rangle = |x_1...x_n\rangle$ in binomial representation with $x_1$ as the most significant bit,

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

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 $\sum_{y=0}^{N-1} = \sum_{y_1=0}^{1} \cdots \sum_{y_n=0}^{1}$

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

In this product representation which is factorized, the corresponding quantum state is not entangled. The product representation makes it easy to construct a quantum circuit that computes the quantum Fourier transform efficiently.

The circuit implementation requires the quantum gates:

- 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.

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

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

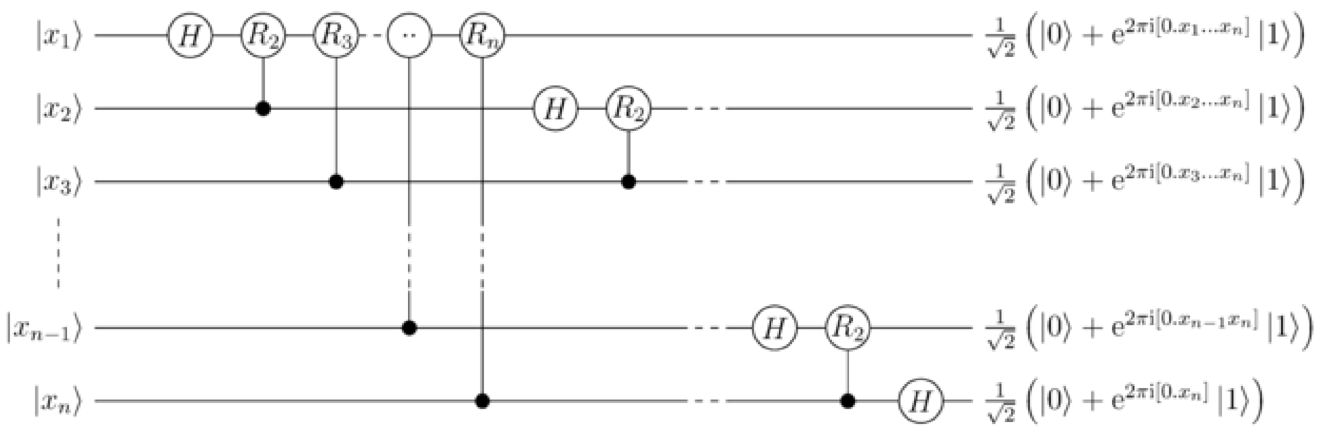


Figure 2: A circuit implementing the quantum Fourier transform [2]

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

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

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

using these gates, the circuit can be executed as in Fig 2

The output state of this circuit coincides with the above product representation, except for the fact that the order of the qubits is reversed. The correct order can be obtained by means of O(n) SWAP gates.

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.

## 2.4 Quantum Phase Estimation

The quantum phase estimation algorithm is a quantum algorithm to estimate the phase corresponding to an eigenvalue of a given unitary operator. Because the eigenvalues of a unitary operator always have unit modulus, they are characterized by their phase, and therefore the algorithm can be equivalently described as retrieving either the phase or the eigenvalue itself.

The process typically begins with an eigenvector of a unitary operator U along with its 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 $\theta$ is the eigenvalue to be estimated.A quantum circuit is constructed to prepare an ancillary qubit in a superposition of states, reflecting varied estimations of the phase $\theta$. This is accomplished through a sequence of controlled-unitary operations, with the number of controls increasing incrementally. The phase $\theta$ is embedded into the ancillary qubit's state through phase kickback, where the eigenvalue $\theta$ is "kicked back" to the ancillary qubit during the controlled-unitary operations. QPE applies an inverse Quantum Fourier Transform to the ancillary qubits, transitioning the phase-encoded state into a superposition of computational basis states, where the probability amplitudes signify $\theta$ estimates.

Ultimately, the ancillary qubits embodying the phase information are measured, resulting in a high-probability estimate of the phase $\theta$. The measurement outcomes furnish a phase approximation, usually presented as a binary fraction.

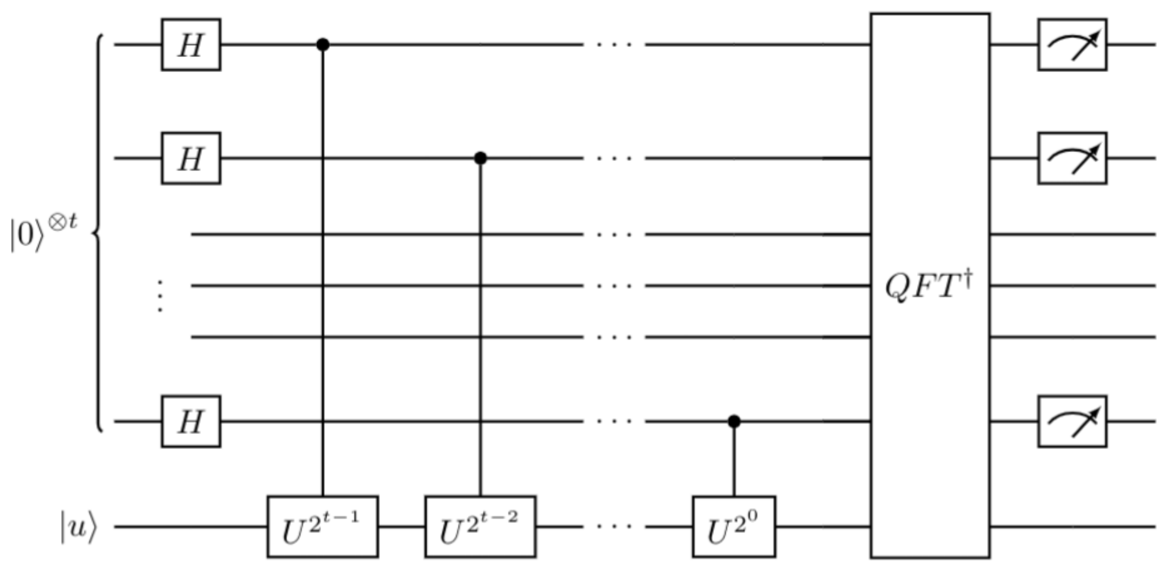


Figure 3: A general quantum circuit for quantum phase estimation [3]

The quantum circuit solving this problem is shown in Fig 3.

- The input $|u\rangle$is in one set of qubit registers along with additional set of t qubits that form the counting register on which we will store the value $2^t\phi$.

$$|\psi_0\rangle = |0\rangle^{\otimes t}|u\rangle$$

- A t-bit Hadamard gate operation$H^{\otimes t}$ on the counting register.

$$|\psi_1\rangle = \frac{1}{\sqrt{2^t}}(|0\rangle + |1\rangle)^{\otimes t}|u\rangle$$

- applying a series of controlled-unitary operations, with the number of controls increasing in each step for the ancillary qubit in a superposition of state,

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

  where $k$ denotes the integer representation of t-bit binary numbers.Phase $\theta$ is encoded into the state of the ancillary qubit.

- Inverse Quantum Fourier Transform on the auxiliary register to recover the state $|2^n\theta\rangle$.

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

- measuring the ancillary qubits representing the phase information to yield an estimate of the phase $\theta$ with high probability. The above expression peaks near $x = 2^t\theta$. For the case when $2^t\theta$ is an integer, measuring in the computational basis gives the phase in the auxiliary register with high probability:

$$|\psi_4\rangle = \left|2^t\theta\right\rangle \otimes |\psi\rangle$$

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

Quantum Phase Estimation (QPE) is a crucial algorithm in quantum computing with diverse applications frequently used as a subroutine in other quantum algorithms, such as Shor's algorithm, the quantum algorithm for linear systems of equations, and the quantum counting algorithm. 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. It also 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

## 3.1 Qiskit Implementation

Qiskit, developed by IBM, serves as an open-source framework tailored for the development of quantum computing software. It provides users with a comprehensive suite of tools for managing quantum circuits, algorithms, and simulators, as well as access to physical quantum hardware via the IBM Quantum Experience platform. By utilizing a high-level programming language based on Python, users can design quantum circuits by specifying quantum gates, qubit operations, and measurements to create quantum algorithms. Qiskit also includes built-in simulators that enable users to simulate the behavior of quantum circuits and offers visualization tools like circuit diagrams and state vectors on conventional computers. These simulators are invaluable for testing and refining quantum algorithms before deploying them on actual quantum hardware. Through the IBM Quantum Experience platform, users can access IBM's real quantum processors, which come with various configurations of qubit quantities and error rates, facilitating practical experimentation and progress in quantum computing.

## 3.2 Execution of Algorithms

*(Circuits with their Executions and Outputs can be accessed in the GitHub links.)*
Link to the Repository:
https://github.com/hudhapm/P471-QIQC-project

### 3.2.1 Quantum Fourier Transform (QFT)

https://github.com/hudhapm/P471-QIQC-project/blob/main/QFT_Implementation.ipynb

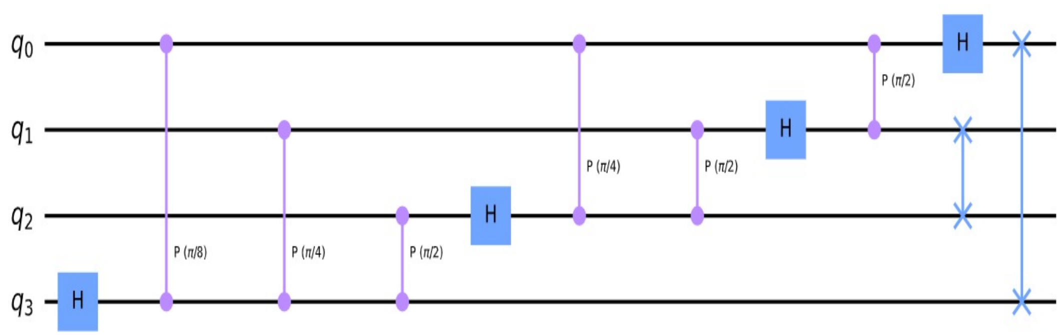


Figure 4: A general quantum circuit for quantum Fourier transform using 4 qubits.

### 3.2.2 Quantum Phase Estimation (QPE)

https://github.com/hudhapm/P471-QIQC-project/blob/main/QPE_Implementation.ipynb

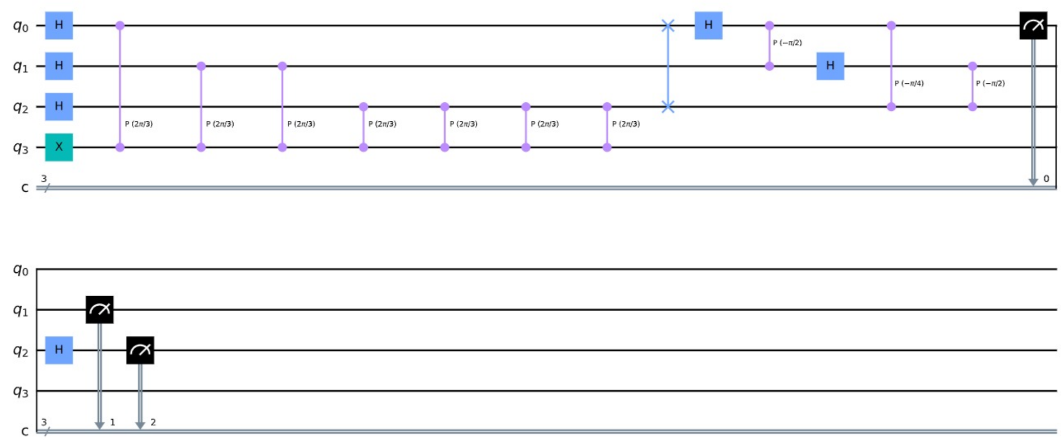


Figure 5: A general quantum circuit for quantum phase estimation using 4 qubits.

# IV Results & Discussion

- Wee successfully implemented and executed quantum circuits for the quantum Fourier transform (QFT) and quantum phase estimation (QPE) algorithms using Qiskit. Comprehensive tests were conducted to ensure the accuracy of our circuits,

- For the QFT circuit carrying out a Quantum fourier transformation on a Gaussian probability distribution function gave back a Gaussian function thus verifying the circuit. Remarkably, even when we altered the mean position, the peak of the Fourier-transformed Gaussian consistently remained at zero, and for an increased 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. On increasing the number of qubits, an increased precision in phase estimation is guaranteed.

Overall, these tests validated the accuracy of our circuits and underscored the efficacy of Qiskit for constructing and verifying quantum algorithms.

Qiskit's simulators offered a cost-effective solution for simulating quantum circuits, aiding algorithm development and testing without needing quantum hardware enabling scalable exploration of circuits with various qubits and gates, ensuring flexibility and speed for efficient experimentation in quantum computing. Quantum Fourier Transform (QFT) could be implemented on various functions and verified classically to provide insights into the efficiency of the quantum algorithms and its applicability in solving specific problems, aiding in algorithm optimization and refinement. The essence of our 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. Further exploration of the quantum counterparts of transforms such as the Laplace transform and Chirp-Z transform can provide an avenue for assessing their speed and effectiveness in quantum computing. A comparative analysis helps identify the strengths and limitations of quantum transforms, facilitating the optimization and adaptation of quantum algorithms for specific tasks.

# References


[1] Qpe. https://link.springer.com/chapter/10.1007/978-3-031-10861-7_2.

[2] wikepedia. Qft. https://en.wikipedia.org/wiki/Quantum_Fourier_transform.

[3] Qpe. https://darveshiyat.medium.com/implementing-quantum-phase-estimation-algorithm-usi