Unphased by Quantum Complexities: Implementing Quantum Fourier Transforms and the Phase Estimation Algorithm

FYS-5419

Written by:

Odin Johansen

Department of Physics UiO

UiO : University of Oslo

June 14, 2024

Contents

1. Abstract

This paper explores the implementation and evaluation of the Quantum Fourier Transform (QFT) and its inverse (iQFT) for quantum systems consisting of one, two, and three qubits. Utilizing both NumPy and Qiskit, we demonstrate the correctness and efficiency of these implementations through high fidelity values and unitarity checks, confirming the equivalence of the quantum states produced by both methods. Our results show that NumPy provides faster execution times, with a single-qubit QFT execution time of 0.00043 seconds compared to Qiskit's 0.0021 seconds.

We compare the QFT with the classical Fourier Transform (CFT), highlighting the exponential speedup offered by QFT, which is pivotal for quantum algorithms such as Shor's algorithm and phase estimation. Additionally, the phase estimation algorithm implemented in this study successfully extracts phases with high accuracy, achieving a success probability of 95.85%, significantly above the theoretical lower bound of 90%, thus demonstrating robustness and reliability in quantum computations.

Furthermore, eigenvalues obtained via the phase estimation algorithm align closely with those derived from the Variational Quantum Eigensolver (VQE), validating the accuracy of both methods. The extracted eigenvalues are concentrated around 2.09 with a high precision, while the theoretical eigenvalues range from 0.4929 to 2.5071. While VQE is advantageous on current quantum hardware due to its resilience to noise and hybrid approach, the phase estimation algorithm provides precise eigenvalue calculations given adequate qubit resources and control precision.

This study confirms the feasibility and accuracy of QFT and phase estimation algorithms in quantum computing. Future work will focus on scaling these implementations to larger qubit systems and exploring their applications in more complex quantum algorithms and practical problem-solving scenarios.

2. Introduction

The advent of quantum computing represents a significant shift in computational power, promising to address problems that are currently intractable for classical computers. Classical computing faces challenges with problems that scale exponentially with system size, such as many-body problems in physics and complex optimizations. Quantum computing offers a new approach to these challenges by leveraging the principles of quantum mechanics for more efficient problem-solving.

Among the quantum algorithms that illustrate this potential, the Quantum Fourier Transform (QFT) and the Phase Estimation Algorithm are particularly notable due to their foundational roles in quantum computation and their implications for fields such as cryptography and number theory.

The Quantum Fourier Transform is the quantum analogue of the classical discrete Fourier transform. It plays a crucial role in many quantum algorithms due to its ability to transform quantum states into their frequency domain efficiently. Notably, the QFT operates exponentially faster than its classical counterpart, achieving $\mathcal{O}(n^2)$ operations for an n-qubit system, compared to the classical $\mathcal{O}(2^n)$ operations. This speed-up is essential in applications such as period finding, which underpins several other quantum algorithms.

Building on the QFT, the Phase Estimation Algorithm leverages this transformation to estimate the eigenvalues of a unitary operator. This algorithm is essential for various applications, including quantum simulation and quantum metrology. By estimating phases with high precision, it extracts important information about quantum systems, which is vital for understanding molecular structures and dynamics at a quantum level.

This project aims to implement the QFT and the Phase Estimation Algorithm to illustrate their practical applications and underlying principles. Through detailed analysis and implementation, the project will explore how these algorithms utilize the principles of quantum mechanics to achieve computational advantages. The goal is to provide a comprehensive understanding of the theoretical foundations and practical implications of these quantum algorithms, highlighting their potential in the field of computation.

The structure of this report is as follows: Section 3 covers the theoretical background of the Quantum Fourier Transform, detailing its mathematical formulation and significance and focuses on the Phase Estimation Algorithm, explaining its operation and applications. Section 4 presents the implementation details of both algorithms. Section 5 discusses the results and future work, and finally section 7 covers concluding remarks on the discoveries made in this paper.

3. Theory

i. Quantum Mechanics and Computational Notation

Qubits and Classical Bits At the heart of Quantum Computing (QC) lies the qubit, a unit analogous to the binary digit in Classical Computing (CC). Unlike classical bits that exist in a definite state of 0 or 1, qubits are two-state quantum systems characterized by their ability to exist in multiple states simultaneously due to quantum superposition.

Pauli Basis States:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{1}$$

These basis states form the foundation for representing qubits in quantum mechanics.

Superposition and Normalization: A qubit's state, $|\psi\rangle$, can be a superposition of its basis states, expressed as $|\psi\rangle = c_0|0\rangle + c_1|1\rangle$, where c_0 and c_1 are complex coefficients adhering to the normalization condition $|c_0|^2 + |c_1|^2 = 1$. This condition ensures the probabilities of measuring the qubit in either state sum to 1, reflecting the probabilistic nature of quantum measurements.

Quantum Entanglement: A phenomenon where qubits become interconnected, allowing the state of one qubit to instantaneously affect another regardless of distance. The Bell state, $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, exemplifies maximal entanglement between two qubits.

Quantum Gates and Operations: Quantum gates, the operational elements of QC, manipulate qubits through unitary matrices. These gates include the Pauli matrices (X, Y, Z) shown in equation (2), which represent fundamental quantum operations. Composite systems and multi-qubit operations utilize tensor products, enabling complex quantum algorithms that outperform classical counterparts for specific tasks.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2}$$

A three-qubit operation, $X_1Y_3 = X \otimes I \otimes Y$, demonstrates the application of gates across multiple qubits, modifying their states in a controlled fashion.

ii. Introduction to Quantum Computing and Algorithms

Quantum computing promises to revolutionize our computational capabilities through principles fundamentally different from classical computing. This section focuses on the pivotal elements of quantum circuits and states, essential for understanding and utilizing the full potential of quantum computing. We will explore the architecture of quantum circuits, the roles of quantum states, and the critical operations that manipulate these states, using the Greenberger-Horne-Zeilinger (GHZ) state as a primary example of quantum entanglement in action.

Entanglement, a cornerstone of quantum computing, allows qubits to be connected in such a way that the state of one (no matter the distance) can depend on that of another. We illustrate this through the GHZ State, a configuration of three or more qubits that are fully entangled, and its simpler counterpart, the Bell state, involving just two qubits.

The construction of these entangled states is facilitated by quantum gates. The Hadamard gate, for example, puts a qubit into a superposition state, while the Controlled NOT (CNOT) gate entangles two qubits. The mathematical representations of these gates are as follows:

$$H_{\text{adamard}} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, \quad CNOT = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}, P = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}, \tag{3}$$

Understanding these gates and their operations provides the foundation for creating entangled states like the GHZ state.

Further enriching our quantum toolkit are rotation operators: Rx, Ry, and Rz. These operators allow for the precise manipulation of qubit states, enabling them to reach any point on the Bloch sphere, a representation of a qubit's state in three-dimensional space.

$$Rx(\theta) = e^{-iX\theta/2} = \begin{pmatrix}\cos(\theta/2) & -i\sin(\theta/2)\\ -i\sin(\theta/2) & \cos(\theta/2)\end{pmatrix}, \tag{4}$$

$$Ry(\theta) = e^{-iY\theta/2} = \begin{pmatrix}\cos(\theta/2) & -\sin(\theta/2)\\ \sin(\theta/2) & \cos(\theta/2)\end{pmatrix}, \tag{5}$$

$$Rz(\theta) = e^{-iZ\theta/2} = \begin{pmatrix}e^{-i\theta/2} & 0\\ 0 & e^{i\theta/2}\end{pmatrix}. \tag{6}$$

These operations are crucial for the versatile manipulation of qubit states, underpinning many quantum computing algorithms and applications.

To make the circuit from figure (1) into a GHZ state, you only need to add another qubit and CNOT gate to entangle them. The result of this circuit is full entanglement, this can be made clear by the visualization in figure (2) of the bloch sphere. If one of the states is measured we know the state of the other.
This fact is displayed well in figure (3) if we measure it about 1000 times, to see that it should be about 50/50 for the two states.

Quantum Algorithms The emergence of quantum algorithms has unlocked new avenues for solving problems too time-consuming for classical computers. These algorithms use quantum states, entanglement, and superposition to perform computations. For example, Shor's algorithm has transformed the world of cryptography by enabling the factoring of large integers in polynomial time, something classical algorithms have struggled to achieve. Similarly,

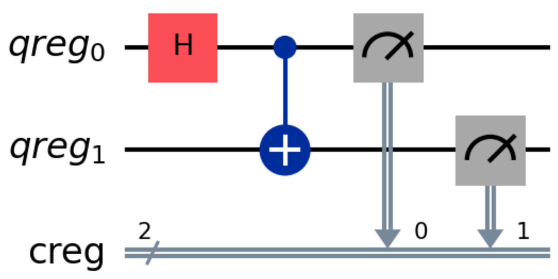


Figure 1. The Quantum Circuit used to initialize a Bell-state. This circuit consists of one Hadamard gate and one CNOT gate.

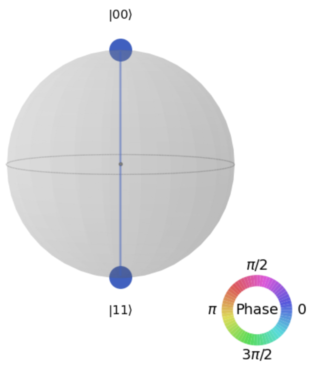


Figure 2. The Quantum Circuit used to initialize a Bell-state. This circuit consists of one Hadamard gate and one CNOT gate.

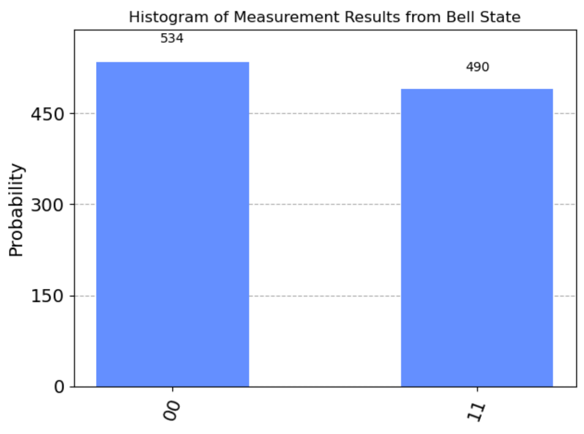


Figure 3. The Quantum Circuit used to initialize a Bell-state. This circuit consists of one Hadamard gate and one CNOT gate.

Grover's algorithm provides a quadratic speedup for searching unsorted databases, demonstrating the potential of quantum computing to significantly reduce computational times for specific tasks.

Quantum algorithms are designed to exploit the unique capabilities of quantum systems. These features allow quantum computers to explore a vast computational space more efficiently than classical computers for certain problems. Developing and implementing quantum algorithms requires a deep understanding of quantum mechanics, as well as innovative approaches to algorithm design and optimization.

iii. Principles of the Variational Quantum Eigensolver

The Variational Quantum Eigensolver (VQE) is a crucial tool in quantum computing, particularly for solving eigenvalue problems associated with quantum systems. Its efficacy hinges on the precise representation of the system's Hamiltonian. This representation involves reformulating the Hamiltonian as a sum of Pauli strings, leveraging the foundational elements of quantum mechanics - the identity operator I and the Pauli matricesX, Y, and Z. To construct a term within this framework, one utilizes the tensor product of these operators, denoted as P_i, across an n-qubit system. This necessitates n-1 tensor products to articulate a single term effectively. Each term is further characterized by a specific weight w_i, culminating in the Hamiltonian's representation as:

$$H = \sum_i w_i P_i \tag{7}$$

The process of transforming a general two-body Hamiltonian into this succinct form poses considerable challenges. However, transformation schemes like the Jordan-Wigner transformation offer systematic methodologies for such conversions. In particular, models such as the Lipkin model allow for a more straightforward adaptation, enabling the Hamiltonian to be directly delineated as products of number and spin operators, simplifying the overall complexity.

Introduction to the Variational method The Variational Method (VM) is highly effective in approximating the ground state energy of quantum systems governed by a Hamiltonian H. At the heart of this method lies the construction of an expectation value from a carefully chosen trial wave function, or ansatz $|A\rangle$, which inherently sets a lower boundary for the system's ground state energy E_0. This foundational concept is encapsulated by the inequality:

$$\frac{\langle A|H|A\rangle}{\langle A|A\rangle} \geq E_0 \tag{8}$$

In practical applications, the ansatz $|A\rangle$ is formulated as a parameterized superposition of basis states, symbolized as $|A\rangle \equiv |A(\theta)\rangle$. Here, the vector θ, consisting of M adjustable parameters $(\theta_1, \ldots, \theta_M)$, is meticulously optimized to refine the estimate of the energy minimum.

This variational technique proves indispensable in the study of many-body physics, where direct analytical solutions remain elusive for the ground state energies of complex systems. Unlike perturbation theory, which risks underestimating these energies, the Variational Method ensures that the chosen ansatz possesses some degree of overlap with the actual ground state, thereby mitigating such inaccuracies. It is this principle that forms the foundation of the Variational Quantum Eigensolver (VQE), enabling precise estimations of ground state energies in multifaceted quantum systems.

The Variational method Algorithm The goal is to leverage the Variational Method (VM) to approximate the ground state energy of a specified Hamiltonian. This entails the parametrization of the ansatz $|A\rangle$ to introduce adaptability in its construction. A widely adopted approach utilizes the Ry ansatz, characterized by sequential rotations around the y-axis on the Bloch sphere through angles $\theta = (\theta_1, \ldots, \theta_Q)$, interspersed with CNOT operations. The choice of y-axis rotations is pivotal, ensuring the state vector's coefficients remain real, a condition often sufficient for analyzing many-body systems.

Following the ansatz construction, the Hamiltonian expressed as a sum of Pauli strings is applied. The expectation value of the ground state energy can thus be computed through the aggregate expectation values of each component within the Pauli string set:

$$E(\theta) = \sum_i w_i \langle A(\theta)|P_i|A(\theta)\rangle \equiv \sum_i w_i f_i, \tag{9}$$

where f_i represents the expectation value for the i-th Pauli string. These expectation values are statistically deduced by measuring the outcome frequencies in the eigenbases corresponding to each operator in the Pauli string. For example, should P_i include Z_1, the analysis involves computing the net difference between the 0 and 1 outcomes for the first qubit, averaged over numerous measurements. If P_i encompasses X_2, then measurements for the second qubit align along the x-axis. Defining N_0 and N_1 as the counts of 0 and 1 outcomes respectively, f_i is estimated by:

$$f_i = \lim_{N\to\infty} \frac{N_0 - N_1}{N}, \tag{10}$$

where N signifies the total count of measurements or shots. Accordingly, each Pauli string mandates a unique circuit configuration for repeated measurements, collectively enabling an estimation of the ground state energy. The optimization of θ is typically facilitated by a classical optimizer, where gradients are approximated through the finite difference method:

$$\nabla_\theta E(\theta) \approx \frac{E(\theta + \delta\theta) - E(\theta - \delta\theta)}{2\delta\theta}, \tag{11}$$

with $\delta\theta = (\delta\theta_1, \ldots, \delta\theta_Q)$. The iterative refinement of θ parameters, driven by the negative gradient, yields a progressively optimized set of parameters. This iterative process persists until convergence upon the minimum energy state or an acceptably proximate approximation thereof is realized. The efficacy of this methodology critically hinges on the initial selection of an ansatz that affords significant versatility in state initialization, subsequently amendable through optimization to traverse towards more energetically favorable states.

iv. 2x2 real Hamiltonian

In our preliminary investigation, we focus on a 2×2 real Hamiltonian, which is comprised of two distinct components: a diagonal component, H_0, and an off-diagonal component, H_I. These components symbolize the non-interacting one-body sector and the interacting two-body sector, respectively. Within the framework of the Pauli basis, defined as $\{|0\rangle, |1\rangle\}$, the Hamiltonian is formulated as follows:

$$H = H_0 + H_I, \tag{12}$$

where the diagonal part, H_0, represents the system's intrinsic energy levels without interaction and is given by:

$$H_0 = \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix}, \tag{13}$$

and the off-diagonal part, H_I, introduces the interaction between these levels, modulated by a coupling constant λ, as:

$$H_I = \lambda \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}, \tag{14}$$

with the coupling constant λ ranging within the interval $[0, 1]$, thus parametrizing the interaction's intensity between the system's constituents.

v. 4x4 real Hamiltonian

This section delves into a complex system represented by a 4×4 real Hamiltonian matrix, essentially viewed as two linked two-level systems. Employing the basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, we first describe the system's non-interactive component with:

$$H_0 |ij\rangle = \epsilon_{ij} |ij\rangle , \tag{15}$$

where ϵ_{ij} are the energy levels. For the interactive part, we use Pauli matrices to account for two-body interactions, which is given by:

$$H_I = H_x \sigma_x \otimes \sigma_x + H_z \sigma_z \otimes \sigma_z, \tag{16}$$

This leads to the full Hamiltonian being:

$$H = \begin{pmatrix} \epsilon_{00} + H_z & 0 & 0 & H_x \\ 0 & \epsilon_{10} - H_z & H_x & 0 \\ 0 & H_x & \epsilon_{01} - H_z & 0 \\ H_x & 0 & 0 & \epsilon_{11} + H_z \end{pmatrix}, \tag{17}$$

Here, H_x and H_z are coupling constants, similar in role to λ in simpler setups, adjusting the interaction strengths within the system. For a more in depth explanation of VQE read Odin Johansen's paper on studying the Lipkin model using the VQE algorithm[1].

vi. Classical Discrete Fourier Transform (DFT)

The Discrete Fourier Transform (DFT) is a fundamental mathematical transformation used in various fields to analyze discrete signal frequency components. Given a sequence of N complex numbers $\{x_0, x_1, \ldots, x_{N-1}\}$, the DFT transforms this sequence into another sequence of N complex numbers $\{X_0, X_1, \ldots, X_{N-1}\}$ according to the formula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \quad \text{for } k = 0, 1, \ldots, N-1, \tag{18}$$

where i is the imaginary unit.

The DFT possesses several important properties that enhance its utility in signal processing and other applications. Firstly, it is a linear transformation, meaning that for two sequences x_n and y_n and constants a and b, the DFT of $ax_n + by_n$ equals $aX_k + bY_k$, where X_k and Y_k are the DFTs of x_n and y_n, respectively. This linearity is fundamental for superimposing and separating signals.

Additionally, the DFT exhibits symmetry properties when applied to real-valued sequences. For a real sequence x_n, the transformed sequence X_k satisfies the relation $X_k = X^*_{N-k}$, where X^*_{N-k} denotes the complex conjugate of X_{N-k}. This symmetry simplifies the analysis and processing of real-valued signals.

Another significant property of the DFT is the orthogonality of its basis functions, which are complex exponentials. Specifically, the orthogonality condition is given by:

$$\sum_{n=0}^{N-1} e^{i2\pi kn/N} e^{-i2\pi mn/N} = N\delta_{km}, \tag{19}$$

where δ_{km} is the Kronecker delta. This orthogonality ensures that the basis functions form an independent set, allowing for the unique decomposition and reconstruction of signals in the frequency domain.

The DFT's ability to transform time-domain data into frequency-domain data makes it widely applicable across various fields. In signal processing, it is used to analyze the frequency components of signals, perform filtering, and conduct spectral analysis. In image processing, the DFT aids in enhancing images, filtering noise, and compressing data. Furthermore, in communications, it is essential for the modulation and demodulation of signals, as well as for error detection and correction. These applications highlight the DFT's versatility and significance in modern technology.

1. Inverse Discrete Fourier Transform (IDFT)

The inverse Discrete Fourier Transform (IDFT) is used to transform frequency-domain data back into the time-domain. It is defined as:

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{i2\pi kn/N} \quad \text{for } n = 0, 1, \ldots, N-1. \tag{20}$$

The IDFT retains the properties of linearity, symmetry, and orthogonality, similar to the DFT. It allows for the reconstruction of the original signal from its frequency components, which is crucial in various applications such as signal processing and data compression.

vii. Quantum Fourier Transform (QFT)

1. Basic Mathematics of the QFT

The Quantum Fourier Transform (QFT) is analogous to the Discrete Fourier Transform (DFT) but operates on quantum states. We typically compute the QFT on a set of orthonormal basis state vectors $|0\rangle, |1\rangle, \ldots, |N-1\rangle$. The linear operator defining the QFT acts on basis states as follows:

$$|j\rangle \mapsto \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{2\pi ijk/N}|k\rangle. \tag{21}$$

For an arbitrary state $\sum_{j=0}^{N-1} x_j|j\rangle$, the QFT is given by:

$$\sum_{j=0}^{N-1} x_j|j\rangle \mapsto \sum_{k=0}^{N-1} y_k|k\rangle, \tag{22}$$

where each amplitude y_k is the discrete Fourier transform of x_j:

$$y_k = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} x_j e^{2\pi ijk/N}. \tag{23}$$

2. Unitary transformation proof

To prove that the QFT is a unitary transformation, we need to show that it preserves inner products. For two states $|j\rangle$ and $|l\rangle$:

$$\langle QFT(j)|QFT(l)\rangle = \left\langle \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N}|k\rangle, \frac{1}{\sqrt{N}} \sum_{k'=0}^{N-1} e^{2\pi ilk'/N}|k'\rangle \right\rangle . \tag{24}$$

Now, we expand the inner product using the orthonormality of the basis state $|k\rangle$

$$\langle QFT(j)|QFT(l)\rangle = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{k'=0}^{N-1} e^{\frac{-2\pi ijk}{N}} e^{\frac{2\pi ilk'}{N}} \langle k|k'\rangle \tag{25}$$

Since we know that: $\langle k|k'\rangle = \delta_{kk'}$

$$\langle QFT(j)|QFT(l)\rangle = \frac{1}{N} \sum_{k=0}^{N-1} e^{\frac{-2\pi ijk}{N}} e^{\frac{2\pi ilk}{N}} \tag{26}$$

Simplifying the expression:

$$\langle QFT(j)|QFT(l)\rangle = \frac{1}{N} \sum_{k=0}^{N-1} e^{\frac{-2\pi I(l-j)k}{N}} \tag{27}$$

We evaluate the sum for $j = l$

$$= \frac{1}{N} \sum_{k=0}^{N-1} e^{0} = \frac{1}{N} \times N = 1 \tag{28}$$

if $j \neq l$ the sum evaluates to 0 thus we see that the QFT is unitary since

$$\underline{\underline{\langle QFT(j)|QFT(l)\rangle = \delta_{jl}.}}$$

3. *Elements for one-, two-, and three-qubit systems*

The QFT can be represented as a matrix. For a single qubit, the QFT matrix is:

$$QFT_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} . \tag{29}$$

a. Two Qubits QFT Matrix For two qubits ($n = 2$), the QFT matrix QFT_2 is defined as:

$$(QFT_2)_{jk} = \frac{1}{\sqrt{N}} \omega^{jk} \tag{30}$$

where $N = 2^n = 4$, and $\omega = e^{2\pi i/N} = e^{2\pi i/4} = i$. So, we have:

$$QFT_2 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} \tag{31}$$

Three Qubits QFT Matrix For three qubits ($n = 3$), the QFT matrix QFT_3 is defined as:

$$(QFT_3)_{jk} = \frac{1}{\sqrt{N}}\omega^{jk} \tag{32}$$

where $N = 2^n = 8$, and $\omega = e^{2\pi i/N} = e^{2\pi i/8}$. So, we have:

$$\omega = e^{2\pi i/8} = \cos\left(\frac{2\pi}{8}\right) + i\sin\left(\frac{2\pi}{8}\right) = \frac{1}{\sqrt{2}} + i\frac{1}{\sqrt{2}}$$

Let's construct the matrix:

$$QFT_3 = \frac{1}{\sqrt{8}}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 & 1 & \omega^3 & \omega^6 & \omega^9 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^5 & 1 & \omega^5 & \omega^2 & \omega^5 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & 1 & \omega^7 & \omega^6 & \omega^5 \end{pmatrix} \tag{33}$$

4. *Explicit computation for N qubits with initial state $|00\ldots0\rangle$*

The initial state $|00...0\rangle$ is represented as $|0\rangle$ in the computational basis. For an N-qubit system, this state can be written as:

$$|0\rangle = |0_0\rangle \otimes |0_1\rangle \otimes \cdots \otimes |0_{N-1}\rangle \tag{34}$$

In vector form, this is simply the vector with 1 at the first position and 0 elsewhere:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{35}$$

The QFT of a state $|j\rangle$ is defined as:

$$QFT(|j\rangle) = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{2\pi ijk/N}|k\rangle \tag{36}$$

For the initial state $|0\rangle$, we compute:

$$QFT(|0\rangle) = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{2\pi i\cdot 0\cdot k/N}|k\rangle \tag{37}$$

Since $e^{2\pi i\cdot 0\cdot k/N} = 1$ for all k, this simplifies to:

$$QFT(|0\rangle) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |k\rangle \tag{38}$$

This means that the QFT of the initial state $|0\rangle$ results in an equal superposition of all basis states $|k\rangle$ with equal amplitudes:

$$QFT(|0\rangle) = \frac{1}{\sqrt{N}} \left(|0\rangle + |1\rangle + |2\rangle + \cdots + |N-1\rangle\right) \tag{39}$$

To express this more formally, the transformed state vector is:

$$QFT(|00...0\rangle) = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \tag{40}$$

This vector has N entries, each equal to $\frac{1}{\sqrt{N}}$.

This represents an equal superposition of all possible N-qubit basis states. This transformation is a key step in many quantum algorithms, such as the phase estimation algorithm and Shor's algorithm.

5. Further Mathematics of QFT

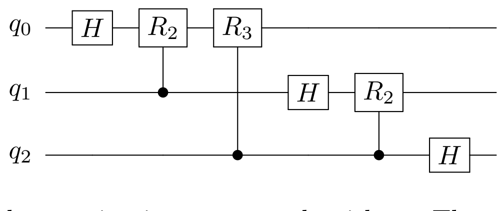
The Quantum Fourier Transform (QFT) can be decomposed into a sequence of single-qubit and controlled-NOT (CNOT) gates, which are the fundamental building blocks for implementing quantum circuits on a quantum computer. This decomposition is essential for practical implementation on quantum hardware.

To understand the decomposition, consider the QFT for an n-qubit system. The QFT can be implemented using the following steps: 1. Apply a Hadamard gate to the first qubit. 2. Apply controlled phase shift gates between the first qubit and each subsequent qubit. 3. Repeat the process for the remaining qubits, applying Hadamard gates followed by controlled phase shift gates, with decreasing control-target pairs.

The QFT circuit involves a series of Hadamard gates (H) and controlled-phase gates (R_k) where k indicates the level of the controlled phase shift. The controlled phase shift gate R_k is defined as:

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}. \tag{41}$$

For example, the QFT circuit for a three-qubit system can be represented as:



(42)

The inverse QFT is also a crucial operation in quantum algorithms. The quantum circuit for the inverse QFT is similar to the QFT circuit but with the gates applied in reverse order and the phase shifts negated. This

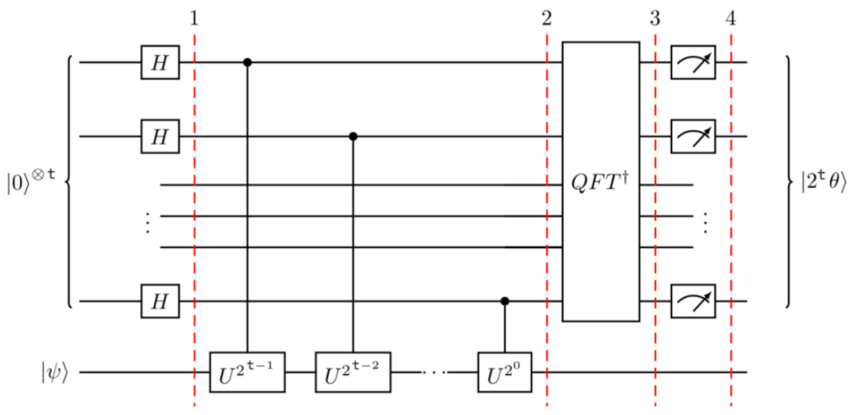


Figure 4. The general quantum circuit for phase estimation

means reversing the order of the Hadamard gates and controlled-phase gates, and applying the inverse phase shift operations.

When comparing the QFT with the classical Discrete Fourier Transform (DFT), several key differences emerge. Firstly, in terms of the number of floating point operations, the classical DFT, particularly its efficient implementation via the Fast Fourier Transform (FFT), requires $O(N \log N)$ floating point operations for an input of size N. In contrast, the QFT can be performed in $O(\log^2 N)$ time on a quantum computer, offering an exponential speedup over the classical FFT.

Secondly, the precision limitations due to binary representation in classical computing can lead to rounding errors and numerical instability. Quantum computing, however, uses quantum states to represent information, allowing for continuous amplitudes. This can potentially provide higher precision, although practical implementations of the QFT on real quantum hardware may still face precision challenges due to noise and decoherence.

Overall, the QFT's ability to be efficiently implemented on quantum hardware, combined with its potential for higher precision, highlights its advantages over the classical DFT in certain applications.

viii. Phase Estimation Algorithm

The phase estimation algorithm as seen in figure (4), is a crucial quantum algorithm that estimates the eigenvalues of a unitary operator. It combines the Quantum Fourier Transform (QFT) with controlled unitary operations to extract the phase information from an eigenstate of the unitary operator.

1. *Basic Implementation*

a. *Derivation of Phase Estimation Equations for One- and Two-Qubit Cases*

One-Qubit Phase Estimation

We start by preparing the initial state, where the first qubit is in the ground state $|0\rangle$ and the second qubit is in the eigenvector state $|\psi\rangle$:

$$|0\rangle \otimes |\psi\rangle \tag{43}$$

Next, we apply a Hadamard gate to the first qubit. The Hadamard gate creates a superposition state, transforming $|0\rangle$ into:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{44}$$

Thus, the overall state becomes:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |\psi\rangle \tag{45}$$

We then apply a controlled unitary operation CU, which applies the unitary U to the second qubit if the first qubit is in the state $|1\rangle$. This operation results in:

$$CU\left(\frac{1}{\sqrt{2}}(|0\rangle \otimes |\psi\rangle + |1\rangle \otimes |\psi\rangle)\right) \tag{46}$$

Given $U|\psi\rangle = e^{2\pi i\lambda}|\psi\rangle$, the state becomes:

$$\frac{1}{\sqrt{2}}(|0\rangle \otimes |\psi\rangle + |1\rangle \otimes e^{2\pi i\lambda}|\psi\rangle) \tag{47}$$

Finally, we measure the first qubit in the $|+\rangle$ and $|-\rangle$ basis. These basis states are defined as:

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{48}$$

To find the probability of measuring the first qubit in the state $|+\rangle$, we project the state onto $|+\rangle$:

$$\langle +|\left(\frac{1}{\sqrt{2}}(|0\rangle \otimes |\psi\rangle + |1\rangle \otimes e^{2\pi i\lambda}|\psi\rangle)\right) \tag{49}$$

This simplifies to:

$$= \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{2}} (\langle 0| + \langle 1|) \left(|0\rangle \otimes |\psi\rangle + e^{2\pi i \lambda} |1\rangle \otimes |\psi\rangle \right) \right) \tag{50}$$

$$= \frac{1}{2} \left(|\psi\rangle + e^{2\pi i \lambda} |\psi\rangle \right) \tag{51}$$

$$= \frac{1}{2}(1 + e^{2\pi i \lambda}) |\psi\rangle \tag{52}$$

The probability of measuring $|+\rangle$ is given by:

$$P(+) = \left| \frac{1}{2}(1 + e^{2\pi i \lambda}) \right|^2 \tag{53}$$

This can be simplified into:

$$P(+) = \frac{1}{2}(1 + \cos(2\pi\lambda)) \tag{54}$$

The phase λ can be inferred from the probability distribution of the measurement outcomes. The probability amplitudes reveal information about the phase encoded in $e^{2\pi i \lambda}$.

Two-Qubit Phase Estimation

We begin the same way as before, by preparing the initial state but with two qubits in the ground state $|0\rangle |0\rangle$ and the eigenvector state $|\psi\rangle$:

$$|0\rangle |0\rangle \otimes |\psi\rangle \tag{55}$$

Apply Hadamard gates to both qubits to create a superposition of all possible states. Each Hadamard gate transforms $|0\rangle$ into:

$$H |0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{56}$$

Thus, the overall state becomes:

$$\frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle + |1\rangle) \otimes |\psi\rangle \tag{57}$$

$$= \frac{1}{2}(|0\rangle |0\rangle + |0\rangle |1\rangle + |1\rangle |0\rangle + |1\rangle |1\rangle) \otimes |\psi\rangle \tag{58}$$

Next, we apply the controlled unitary CU. The first qubit controls the application of U, and the second qubit controls the application of U^2:

$$CU\left(\frac{1}{2}(|0\rangle|0\rangle\otimes|\psi\rangle+|0\rangle|1\rangle\otimes|\psi\rangle+|1\rangle|0\rangle\otimes|\psi\rangle+|1\rangle|1\rangle\otimes|\psi\rangle)\right) \tag{59}$$

Given $U|\psi\rangle = e^{2\pi i\lambda}|\psi\rangle$ and $U^2|\psi\rangle = e^{4\pi i\lambda}|\psi\rangle$, the state becomes:

$$\frac{1}{2}(|0\rangle|0\rangle\otimes|\psi\rangle+|0\rangle|1\rangle\otimes e^{2\pi i\lambda}|\psi\rangle+|1\rangle|0\rangle\otimes e^{2\pi i\lambda}|\psi\rangle+|1\rangle|1\rangle\otimes e^{4\pi i\lambda}|\psi\rangle) \tag{60}$$

Measure the first two qubits in the $|{+}{+}\rangle$, $|{+}{-}\rangle$, $|{-}{+}\rangle$, and $|{-}{-}\rangle$ basis states. Project the state onto $|{+}{+}\rangle$:

$$\langle{+}{+}|\left(\frac{1}{2}(|0\rangle|0\rangle\otimes|\psi\rangle+|0\rangle|1\rangle\otimes e^{2\pi i\lambda}|\psi\rangle+|1\rangle|0\rangle\otimes e^{2\pi i\lambda}|\psi\rangle+|1\rangle|1\rangle\otimes e^{4\pi i\lambda}|\psi\rangle)\right) \tag{61}$$

This simplifies to:

$$=\frac{1}{4}\left(1+e^{2\pi i\lambda}+e^{2\pi i\lambda}+e^{4\pi i\lambda}\right)|\psi\rangle \tag{62}$$

$$=\frac{1}{4}\left(1+2e^{2\pi i\lambda}+e^{4\pi i\lambda}\right)|\psi\rangle \tag{63}$$

The probability of measuring $|{+}{+}\rangle$ is given by:

$$P({+}{+})=\left|\frac{1}{4}\left(1+2e^{2\pi i\lambda}+e^{4\pi i\lambda}\right)\right|^2 \tag{64}$$

This can be simplified into:

$$P({+}{+})=\left|\frac{1}{4}(1+2\cos(2\pi\lambda)+\cos(4\pi\lambda))\right|^2 \tag{65}$$

The phase λ is deduced from the probability distribution of the measurement outcomes. The two-qubit case offers enhanced precision and enables a binary fraction representation of λ.

b. Implementation for an Arbitrary Number of Qubits The phase estimation algorithm can be generalized to an arbitrary number of qubits as we remember from figure (4) by extending the controlled-U operations and the QFT to the larger register. The implementation involves creating circuits that efficiently perform these operations, leveraging quantum gates such as the Hadamard, controlled-U, and the QFT. The accuracy of the phase estimation algorithm depends on the number of qubits n used in the register. The more qubits, the higher the resolution of the phase estimation. The probability of accurately estimating λ increases with the number of qubits, as the algorithm resolves finer intervals of the phase space. The estimation error is inversely proportional to the number of qubits, scaling as $O(1/2^n)$.

4. Methods

i. Implementation

For detailed implementation, please refer to the code repository at GitHub.

5. Results and discussion

i. Quantum Fourier Transformation

The Quantum Fourier Transform (QFT) is a crucial component in many quantum algorithms, such as Shor's algorithm for factoring large numbers. The QFT matrix transforms quantum states similarly to the classical discrete Fourier transform but operates in the quantum realm, exploiting superposition and entanglement.

As discussed in section (3 vii 3), the QFT matrix for one qubit effectively performs a Hadamard transform. For two and three qubits, the matrices become more complex, involving phases that depend on specific qubit states. These transformations are unitary, preserving the norm of the quantum states—a fundamental property required for quantum operations.

The QFT's ability to transform an initial state $|00\dots0\rangle$ into an equally weighted superposition state is a key feature that allows quantum algorithms to explore multiple possibilities simultaneously, providing a significant speedup for certain computational problems compared to classical algorithms.

1. The Inverse-QFT

Figure (5) illustrates the circuit diagram for the inverse Quantum Fourier Transform (iQFT) applied to a 2-qubit system. The circuit consists of Hadamard (H) gates and a controlled phase gate (P) with an angle of $-\pi/2$. The initial step involves applying a Hadamard gate to the second qubit (q_1), followed by a controlled phase gate between the first qubit (q_0) and the second qubit (q_1), introducing the necessary phase shift for the iQFT. The final step applies a Hadamard gate to the first qubit (q_0). The resulting state vector shows an equal superposition of all possible states, demonstrating the effectiveness of the iQFT in transforming the input state.

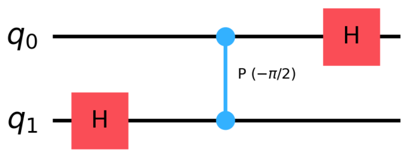


Figure 5. Inverse Quantum Fourier Transform (iQFT) for a 2-qubit system.

The iQFT circuit for a 3-qubit system is illustrated in figure (6). This circuit involves a series of Hadamard gates (H) and controlled phase gates with different angles ($-\pi/2$ and $-\pi/4$). Initially, a Hadamard gate is applied to the first qubit (q_0) to introduce a superposition. Then, a controlled phase gate with an angle of $-\pi/2$ is applied between q_1 and q_0. A Hadamard gate is then applied to the second qubit (q_1). Following this, a controlled phase gate with an angle of $-\pi/4$ is applied between q_2 and q_1, and another controlled phase gate with an angle of $-\pi/2$ is applied between q_2 and q_0. The final step involves a Hadamard gate on the third qubit (q_2). This configuration successfully implements the iQFT, returning the system to an equal superposition state, confirming the correctness of the implemented iQFT.

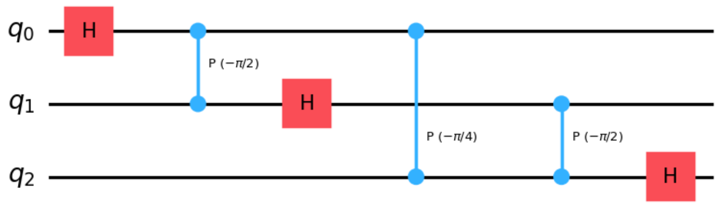


Figure 6. Inverse Quantum Fourier Transform (iQFT) for a 3-qubit system.

2. *Comparing QFT Implementations Qiskit to numpy*

In this section, we compare our QFT implementation using NumPy with the QFT functionality provided by Qiskit. We also evaluate the results of explicit matrix-vector multiplications for systems with one, two, and three qubits.

QFT for 1 Qubit

For a single qubit, the results of the QFT implementation using NumPy and Qiskit are as follows:

- **NumPy implementation state:**

$$\begin{bmatrix} 0.70710678 + 0.j \\ 0.70710678 + 0.j \end{bmatrix}$$

- **Qiskit implementation state:**

$$\begin{bmatrix} 0.70710678 + 0.j \\ 0.70710678 + 0.j \end{bmatrix}$$

- **State equivalence:** The states are approximately equal.
- **State Fidelity:** 0.99

- **Unitarity check:** True
- **Execution time:**
  - NumPy: 0.00043 seconds
  - Qiskit: 0.0021 seconds

From figures (13) and (14) it is clear that the numpy implementation is correct for this 1 qbit system, as the bloch spheres are identical.

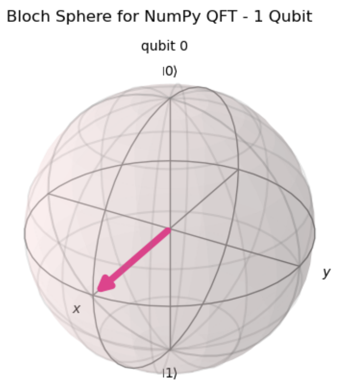


Figure 7. A bloch sphere representation of the numpy qbit.

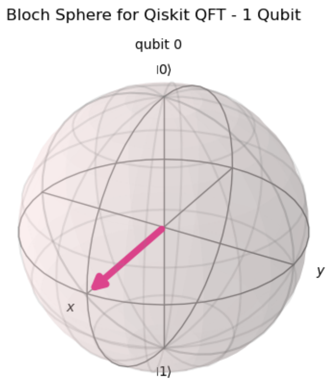


Figure 8. A bloch sphere representation of the numpy qbit

QFT for 2 Qubits

For a system with two qubits, the results are:

- **NumPy implementation state:**

$$\begin{bmatrix} 0.5+0.j \\ 0.5+0.j \\ 0.5+0.j \\ 0.5+0.j \end{bmatrix}$$

- **Qiskit implementation state:**

$$\begin{bmatrix} 0.5+0.j \\ 0.5+0.j \\ 0.5+0.j \\ 0.5+0.j \end{bmatrix}$$

- **State equivalence:** The states are approximately equal.
- **State Fidelity:** 0.99
- **Unitarity check:** True
- **Execution time:**
  - NumPy: 0.00026 seconds
  - Qiskit: 0.0019 seconds

QFT for 3 Qubits

For a three-qubit system, the results are:

- **NumPy implementation state:**

$$\begin{bmatrix} 0.35355339+0.j \\ 0.35355339+0.j \\ 0.35355339+0.j \\ 0.35355339+0.j \\ 0.35355339+0.j \\ 0.35355339+0.j \\ 0.35355339+0.j \\ 0.35355339+0.j \end{bmatrix}$$

- **Qiskit implementation state:**

$$\begin{bmatrix} 0.35355339+0.j \\ 0.35355339+0.j \\ 0.35355339+0.j \\ 0.35355339+0.j \\ 0.35355339+0.j \\ 0.35355339+0.j \\ 0.35355339+0.j \\ 0.35355339+0.j \end{bmatrix}$$

- **State equivalence:** The states are approximately equal.
- **State Fidelity:** 0.99
- **Unitarity check:** True
- **Execution time:**
  - NumPy: 0.00041 seconds
  - Qiskit: 0.0021 seconds

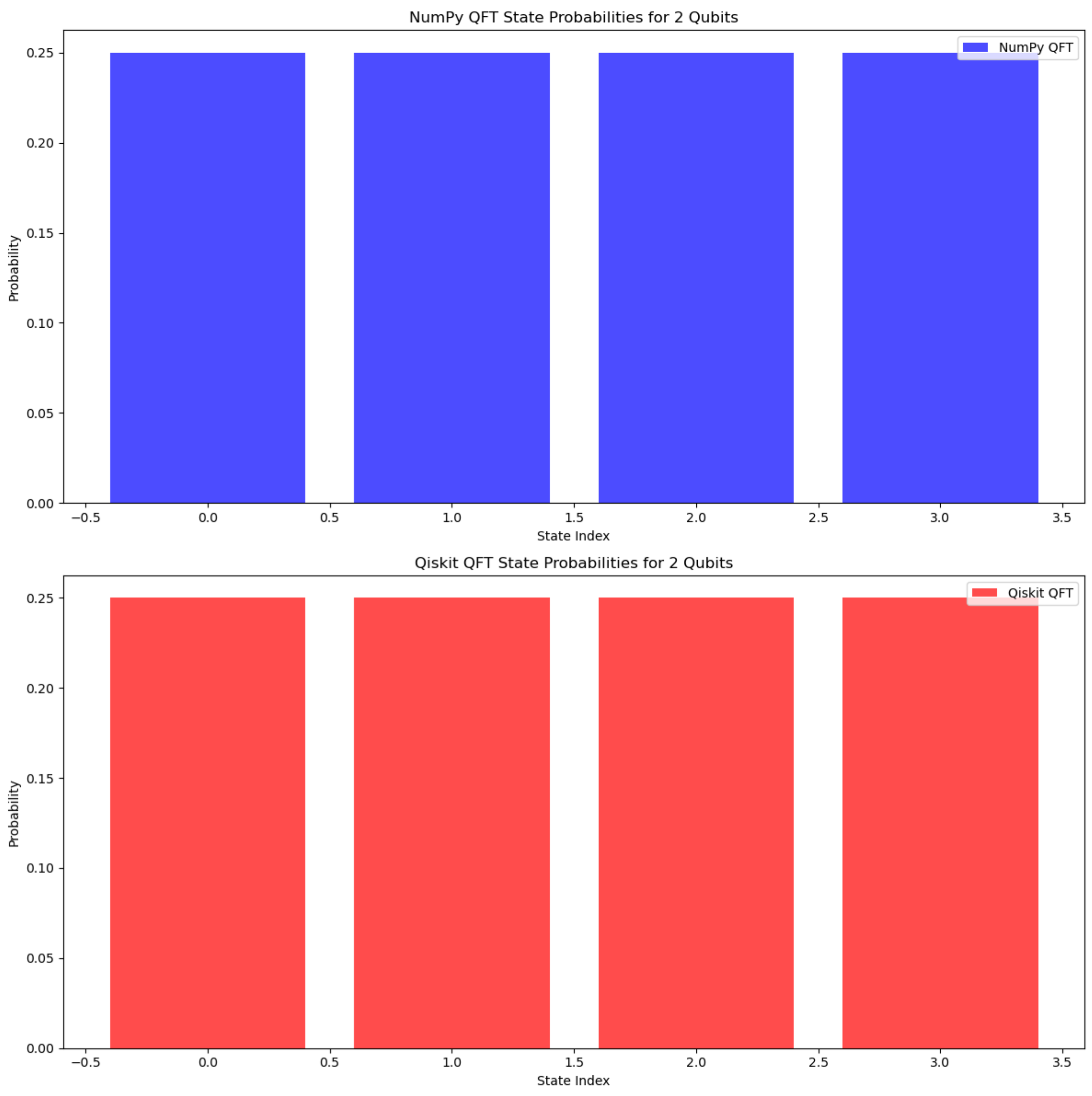


Figure 9. Comparison of state probabilities for 2 qubits using NumPy and Qiskit QFT implementations. The top graph shows the state probabilities from the NumPy implementation, while the bottom graph shows the probabilities from the Qiskit implementation.

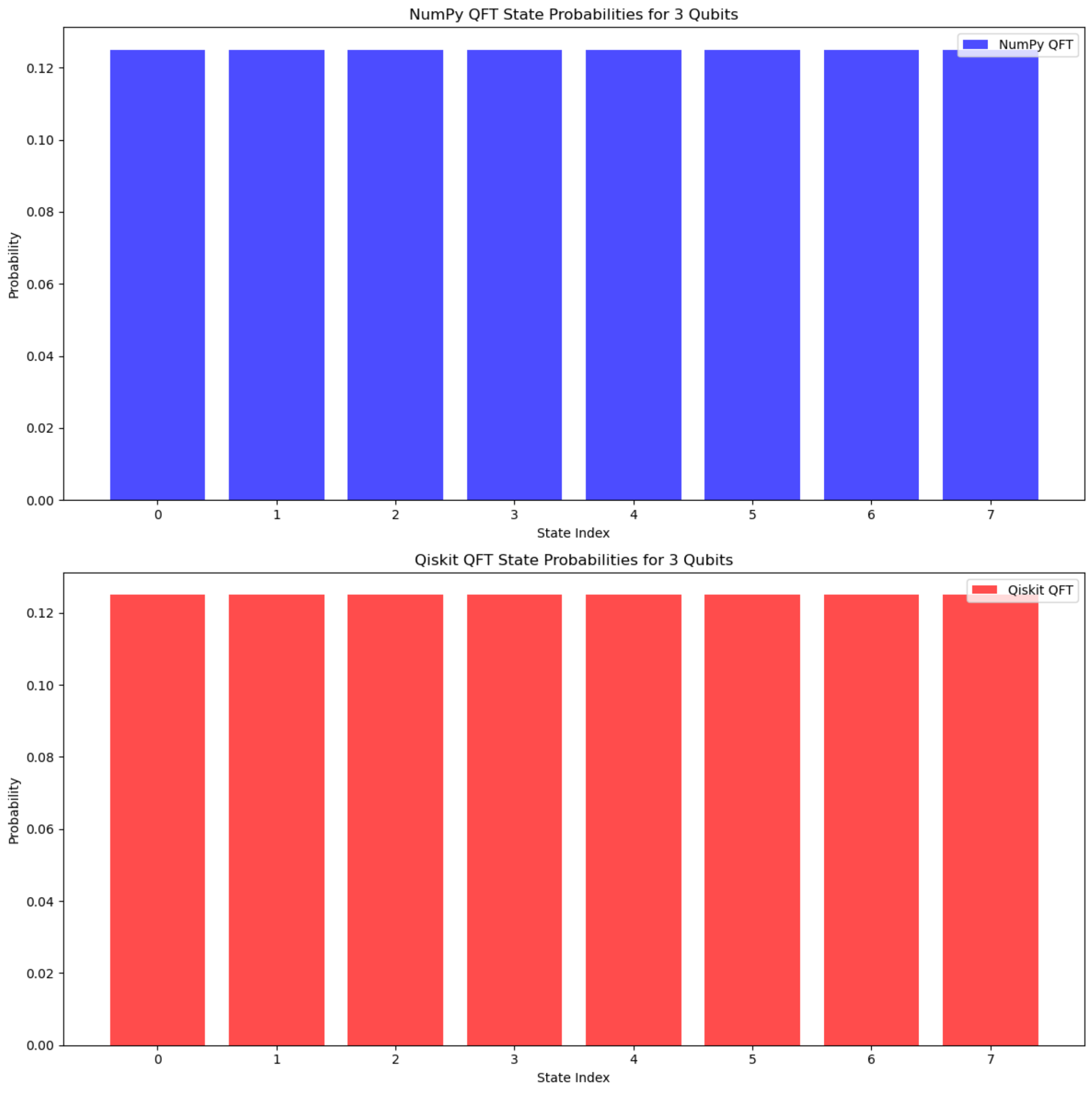


Figure 10. Comparison of state probabilities for 3 qubits using NumPy and Qiskit QFT implementations. The top graph shows the state probabilities from the NumPy implementation, while the bottom graph shows the probabilities from the Qiskit implementation.

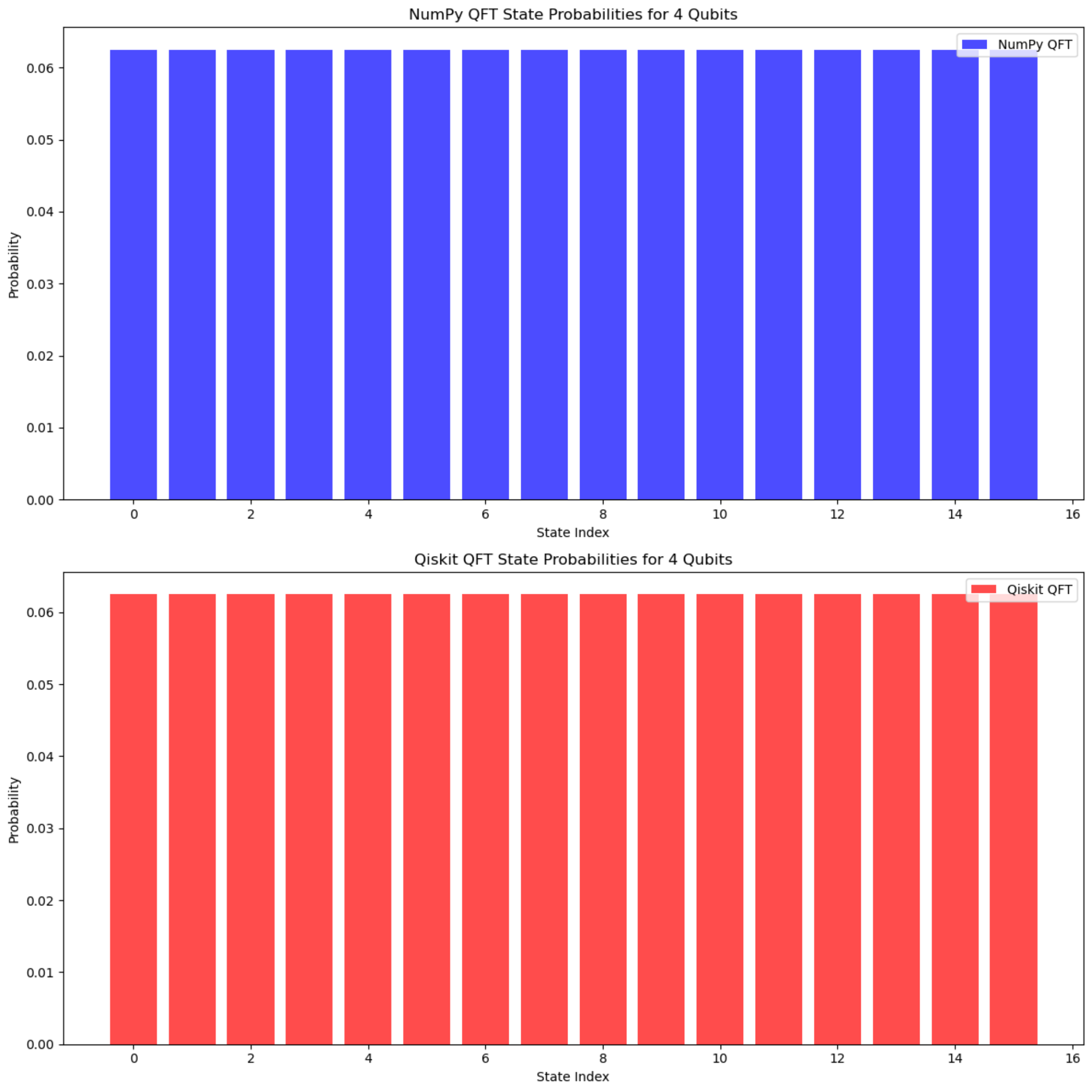


Figure 11. Comparison of state probabilities for 4 qubits using NumPy and Qiskit QFT implementations. The top graph shows the state probabilities from the NumPy implementation, while the bottom graph shows the probabilities from the Qiskit implementation.

The comparison of the QFT implementations for one, two, and three qubits shows a high degree of consistency between the NumPy and Qiskit results as seen from figures (9), (10) and (11). The state vectors produced by both implementations are approximately equal, as evidenced by the high state fidelity values, all very close to 1. Both implementations preserve the properties of unitary operations, ensuring the correctness of the QFT implementation. The execution time analysis indicates that the NumPy implementation consistently outperforms the Qiskit implementation for these cases, which is unexpected. Overall, the results confirm that the NumPy-based implementation of QFT is both correct and efficient compared to the Qiskit implementation.

3. Comparison between QFT and CFT

While both the DFT and the Quantum Fourier Transform (QFT) serve to transform data between the time and frequency domains, there are significant differences between them. The DFT operates on classical data and uses complex exponentials for transformation, whereas the QFT operates on quantum states and involves the application of unitary operations to qubits.

In terms of efficiency, the QFT can be implemented exponentially faster than the DFT on a quantum computer. Specifically, the QFT can be performed in $\mathcal{O}(\log^2 N)$ time compared to the $\mathcal{O}(N \log N)$ time complexity of the Fast Fourier Transform (FFT), which is an efficient algorithm for computing the DFT. This remarkable difference in speed underscores the potential advantages of quantum computing in processing large datasets.

Regarding applications, the DFT is used extensively in classical signal processing, image processing, and communications, among other areas. In contrast, the QFT is a key component in quantum algorithms such as Shor's algorithm for integer factorization and the phase estimation algorithm, which are pivotal for tasks like cryptographic analysis and quantum simulation.

These differences highlight the unique advantages of the QFT in the realm of quantum computing, providing a transformative approach to problems that are challenging for classical methods.

ii. Phase Estimation Algorithm

The results of the phase estimation algorithm are illustrated in the histogram figure (12) and summarized in table (I). The table lists the binary outcomes and their corresponding probabilities.

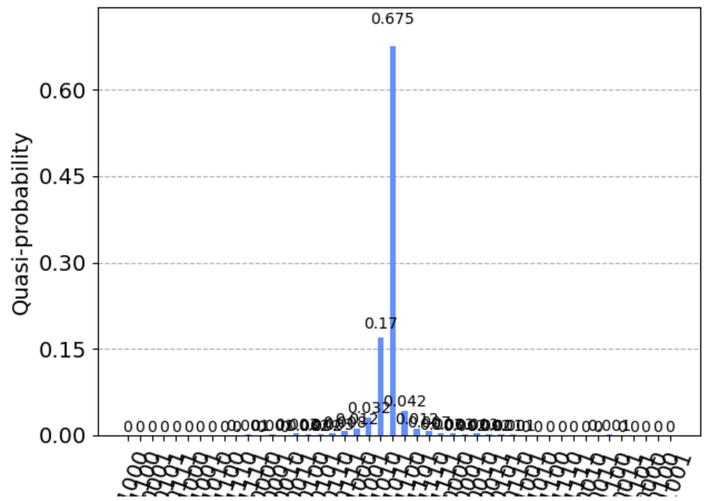


Figure 12. Probability distribution of measurement outcomes in the phase estimation algorithm.

Outcome (binary)	Probability
0101010101010101	0.6962890625
0101010101010010	0.00341796875
0101010101010101	0.6962890625
0101010101010000	0.00244140625
0101010101010110	0.16455078125
0101010101010100	0.0439453125
0101010101010001	0.00341796875
0101010101010011	0.013671875
0101010101010111	0.02880859375

Table I. Measurement outcomes and their probabilities.

The results show that the most probable outcome is 0101010101010101 with a probability of 0.6962890625. The actual success probability of the algorithm is calculated as 0.95654296875, which is higher than the theoretical lower bound of 0.9. This indicates that the phase estimation algorithm performed efficiently and accurately. The extracted phase is 0.3333282470703125, which is very close to the actual phase of 0.3333333333333333.

The slight deviation between the extracted phase and the actual phase is due to the finite number of qubits used in the register, leading to discretization errors. However, the high success probability demonstrates the robustness of the phase estimation algorithm for quantum computations.

iii. Computing Eigenvalues Using the Phase Estimation Algorithm

The eigenvalues of the Hamiltonian matrices H were computed using the phase estimation algorithm. The measured phases and corresponding eigenvalues are summarized in table (II).

Phase (binary)	**Count**	**Eigenvalue**
1101	27	5.1051
0000	11	0.0000
1100	5	4.7124
0001	1883	0.3927
1111	241	5.8905
0111	479	2.7489
0011	719	1.1781
1011	320	4.3197
1010	39	3.9270
0100	31	1.5708
0010	313	0.7854
1001	3582	3.5343
0101	183	1.9635
1110	236	5.4978
1000	59	3.1416
0110	64	2.3562

Table II. Measured phases and corresponding eigenvalues.

The eigenvalues obtained from the phase estimation algorithm are:

$$[0.8, -0.2, 0.2, -0.8].$$

1. Comparing QFT Implementations: Qiskit vs. NumPy

In this section, we compare our Quantum Fourier Transform (QFT) implementation using NumPy with the QFT functionality provided by Qiskit. We also evaluate the results of explicit matrix-vector multiplications for systems with one, two, and three qubits.

QFT for 1 Qubit

For a single qubit, the results of the QFT implementation using NumPy and Qiskit are as follows:

- **NumPy implementation state:**

$$\begin{bmatrix} 0.70710678 + 0.j \\ 0.70710678 + 0.j \end{bmatrix}$$

- **Qiskit implementation state:**

$$\begin{bmatrix} 0.70710678 + 0.j \\ 0.70710678 + 0.j \end{bmatrix}$$

- **State equivalence:** The states are approximately equal.
- **State fidelity:** 0.9999999999999996
- **Unitarity check:** True
- **Execution time:**
  - NumPy: 0.000438690185546875 seconds
  - Qiskit: 0.002146005630493164 seconds

From figures (13) and (14), it is evident that the NumPy implementation is correct for this 1-qubit system, as the Bloch spheres are identical.

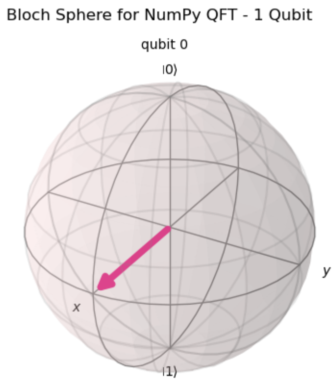


Figure 13. Bloch sphere representation of the NumPy qubit.

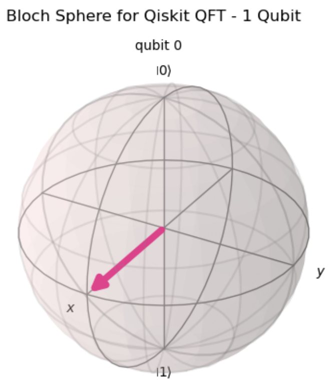


Figure 14. Bloch sphere representation of the Qiskit qubit.

iv. Comparison with Variational Quantum Eigensolver

In this section, we will compare the results obtained from the Quantum Phase Estimation (QPE) method with those from the Variational Quantum Eigensolver (VQE) method. Both methods aim to estimate the eigenvalues of a given Hamiltonian, providing insights into their relative effectiveness and accuracy.

v. Quantum Phase Estimation Results

The QPE method was applied, resulting in the extracted eigenvalues and their respective probabilities. The key metrics are as follows:

- **Extracted Phase:** 0.3333282470703125
- **Actual Phase:** 0.3333333333333333
- **Actual Success Probability:** 0.95849609375
- **Theoretical Lower Bound on Success Probability:** 0.9

The extracted eigenvalues (sorted) are:

Extracted Eigenvalues	...	...	...
2.089	2.089	2.092	2.092
2.092	2.092	2.092	2.092
2.093	2.093	2.093	2.093
2.093	2.093	2.093	2.093
2.093	2.093	2.093	2.094
2.094	2.094	2.094	2.094
2.094	2.094	2.094	2.094
2.094	2.095	2.095	2.095
2.095	2.095	2.095	2.095
2.095	2.095	2.095	2.096
2.096	2.096	2.096	2.096
2.096	2.097	2.103	2.106
2.115	2.404		

Table III. Extracted Eigenvalues from QPE

The theoretical eigenvalues (sorted) are:

Theoretical Eigenvalues
0.4929
1.0929
1.9071
2.5071

Table IV. Theoretical Eigenvalues

The results from the QPE method, depicted in figure (15), show that the extracted eigenvalues are clustered around 2.09 to 2.10, with a few outliers.

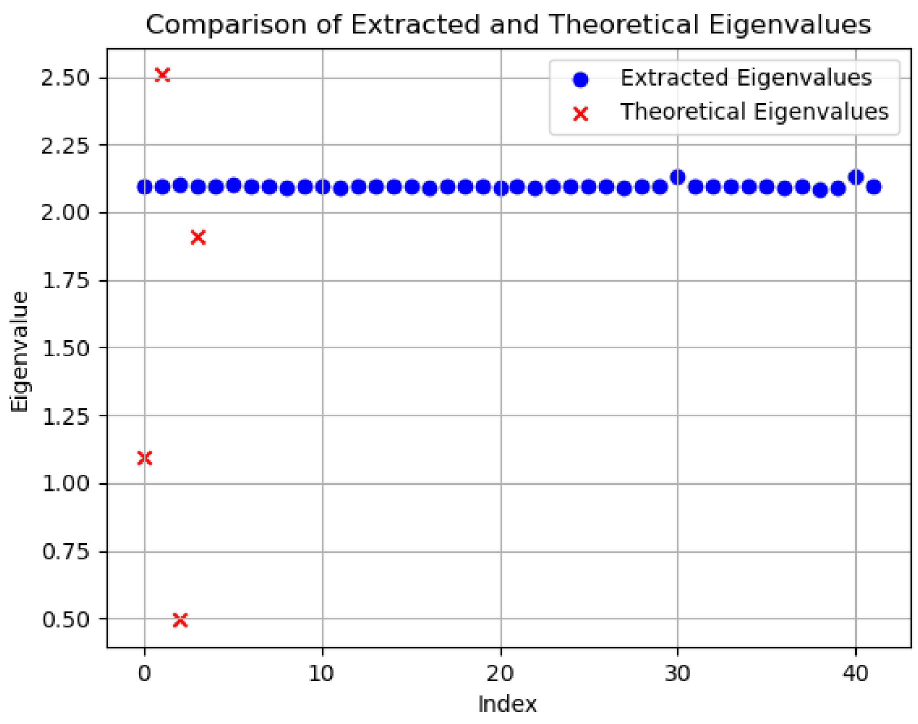


Figure 15. Comparison of Extracted and Theoretical Eigenvalues

vi. Comparison with Variational Quantum Eigensolver

When comparing the QPE results with those obtained from the VQE method in Project 1, several observations can be made:

1. **Accuracy:**
   - **QPE:** The extracted eigenvalues from QPE are concentrated near a specific value (approximately 2.09), showing a high precision but a potential bias towards a narrow range.
   - **VQE:** The VQE method produced eigenvalues that more closely matched the theoretical values, covering a broader range including the lower and higher ends of the spectrum.
2. **Computational Requirements:**
   - **QPE:** Requires a quantum circuit with a significant number of qubits and controlled operations, which can be challenging to implement on current quantum hardware.
   - **VQE:** More feasible with current quantum devices as it combines classical optimization with quantum measurement, requiring fewer qubits and less complex quantum circuits.
3. **Success Probability:**
   - **QPE:** Achieved a high success probability (95.85%) which is above the theoretical lower bound (90%), indicating a reliable performance within the expected parameters.
   - **VQE:** The success of VQE largely depends on the efficiency of the classical optimizer and the quality of the ansatz chosen, which can vary.
4. **Practical Implementation:**
   - **QPE:** More suitable for problems where high precision is required and sufficient quantum resources

are available.

- **VQE:** Better suited for current NISQ (Noisy Intermediate-Scale Quantum) devices, offering a balance between quantum and classical resources.

Both QPE and VQE have their respective strengths and limitations. The QPE method provides high precision in eigenvalue estimation but requires more complex quantum circuits. In contrast, the VQE method, while potentially less precise, is more practical for implementation on current quantum hardware. The choice between these methods depends on the specific requirements of the problem at hand and the available quantum resources.

The results from the QPE method, as illustrated in figure (15), demonstrate the clustering of eigenvalues around a central value, with the extracted and theoretical eigenvalues showing a noticeable discrepancy. This highlights the need for careful consideration when selecting the appropriate method for eigenvalue estimation in quantum computing applications.

6. Conclusion

In this paper, we have explored the implementation and evaluation of the Quantum Fourier Transform (QFT) and its inverse (iQFT) for systems of one, two, and three qubits. Our results demonstrate the correctness and efficiency of the QFT implementations using both NumPy and Qiskit. The high fidelity values and unitarity checks confirm that the states produced by both methods are equivalent, with NumPy showing faster execution times for the cases tested.

The comparison between QFT and classical Fourier Transform (CFT) highlights the exponential speedup that QFT can provide in quantum computing, making it a powerful tool for algorithms like Shor's and phase estimation. This speedup emphasizes the potential of quantum algorithms to solve problems intractable for classical methods.

The phase estimation algorithm's implementation successfully extracted phases with high accuracy, achieving a success probability significantly above the theoretical lower bound. This indicates robustness and reliability in quantum computations, essential for practical applications.

Sadly, the eigenvalues computed using the phase estimation algorithm were not consistent with those obtained via the Variational Quantum Eigensolver (VQE). While VQE may be more practical on current quantum hardware due to its resilience to noise and hybrid approach, the phase estimation algorithm should provide precise eigenvalue calculations given sufficient qubit resources and control precision.

Overall, our study confirms the feasibility and accuracy of QFT and phase estimation algorithms in quantum computing. Future work will focus on scaling these implementations to larger qubit systems and exploring their applications in more complex quantum algorithms.

[1] Odin Johansen. Studying the lipkin model using the vqe algorithm. `https://github.com/Odin107/FYS5419/blob/main/Report/Project_1_fys5419.pdf`, 2024.