Quantum Computation – 15-Session Course

Contents

This document provides a **complete, teaching-ready syllabus** for a 15-session course on **Quantum Computation**. Each session includes:

- Learning Objectives
- Core Concepts / Outline
- Theory Intuition
- Examples
- In-Class Exercises
- Homework
- Python/Qiskit Coding Suggestions
- Suggested Figures (Bloch sphere, circuits, etc.)

You can use this as:

- A printable handout for students
- A base to create slides
- A structure for designing assignments and exams

---

1 Session 1: Introduction Mathematical Foundations

1.1 Learning Objectives

By the end of this session, students should be able to:

- Explain why quantum computation is interesting and potentially powerful.
- Distinguish classical bits from quantum bits (qubits).
- Describe the notion of a quantum state as a complex vector.
- Understand basic linear algebra notions: vectors, norms, inner products.
- Install Python and Qiskit, and run a minimal quantum circuit.

1.2 Core Concepts / Outline

- Motivation: limitations of classical computing, Moores law, new applications.
- Bits vs. Qubits.
- State vectors and the 2D complex vector space for a single qubit.
- Normalization and probabilities.
- Tensor products (conceptual preview).
- Overview of the circuit model of quantum computation.
- Tooling: Python, Qiskit, and simulators.

1.3 Theory Intuition

Classical computation is based on **bits** that can take values 0 or 1. A classical register of n bits can be in exactly one of 2^n possible states at any time.

In contrast, a **qubit** is described by a **normalized complex 2D vector**:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad \text{with } |\alpha|^2 + |\beta|^2 = 1.$$

Here, $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

The squared magnitudes $|\alpha|^2$ and $|\beta|^2$ represent measurement probabilities in the computational basis. The fact that a qubit can be in a **superposition** of basis states enables new kinds of computation.

1.4 Example

Consider the state:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle.$$

The probabilities of measuring 0 or 1 are both 1/2. This is analogous to a fair coin, but represented as a quantum state.

1.5 In-Class Exercises

1. Represent the states $|0\rangle$, $|1\rangle$, and $(|0\rangle + |1\rangle)/\sqrt{2}$ as column vectors. 2. Verify normalization for several example states. 3. Discuss qualitatively how a 2-qubit system would be represented (preview of tensor products).

1.6 Homework

1. Normalize the state

$$|\phi\rangle = 2|0\rangle + 3|1\rangle,$$

and compute measurement probabilities. 2. Write a short paragraph explaining, in your own words, the difference between a classical bit and a qubit. 3. Install Python and Qiskit on your computer and run the sample code below. Take a screenshot of the output.

1.7 Python/Qiskit Coding

```
# Minimal Qiskit example: single-qubit Hadamard and measurement
from qiskit import QuantumCircuit, Aer, execute

# Create a circuit with 1 qubit and 1 classical bit
qc = QuantumCircuit(1, 1)

# Apply a Hadamard gate to create superposition
qc.h(0)

# Measure the qubit
qc.measure(0, 0)

# Simulate the circuit
```

```
backend = Aer.get_backend('qasm_simulator')
job = execute(qc, backend, shots=1024)
result = job.result()
counts = result.get_counts()

print("Circuit:")
print(qc)
print("Measurement counts:", counts)
```

1.8 Suggested Figures

- **Conceptual diagram**: Classical bit vs. qubit (bit as a two-state system, qubit as a vector in a 2D complex space).
- **Vector diagram**: Show $|0\rangle$ and $|1\rangle$ as orthogonal basis vectors.
- **Simple circuit**: One-qubit circuit with an H gate followed by a measurement:

  ```
  q0: H

  c0:
  ```

---

2 Session 2: Linear Algebra, Dirac Notation Operators

2.1 Learning Objectives

By the end of this session, students should be able to:

- Recall the main concepts of linear algebra used in quantum mechanics.
- Work with inner products, norms, and orthonormal bases.
- Understand Dirac (bra-ket) notation.
- Recognize unitary and Hermitian matrices and their roles in quantum computing.

2.2 Core Concepts / Outline

- Vector spaces and basis vectors.
- Inner product and norm.
- Orthonormal bases.
- Dirac notation: kets, bras, and bra-ket products.
- Unitary and Hermitian operators.
- Global phase vs. relative phase.

2.3 Theory Intuition

Quantum states live in **complex vector spaces** (Hilbert spaces). An **inner product** allows us to define angles and lengths in this space:

$$\langle\phi|\psi\rangle = \sum_i \phi_i^* \psi_i.$$

In **Dirac notation**, kets $|\psi\rangle$ represent column vectors, while bras $\langle\psi|$ represent row vectors of complex conjugates. The inner product is written as $\langle\phi|\psi\rangle$.

Unitary operators U (with $U^\dagger U = I$) represent valid quantum evolutions; they preserve norms and thus total probability. **Hermitian operators** are observables (measurable quantities) in quantum mechanics.

A key point is that **global phase** has no physical effect:

$$|\psi\rangle \text{ and } e^{i\theta}|\psi\rangle$$

represent the same physical state.

2.4 Example

Show that the Hadamard gate

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

is unitary by verifying $H^\dagger H = I$.

2.5 In-Class Exercises

1. Compute $\langle 0|1\rangle$, $\langle 1|1\rangle$, and $\langle +|+\rangle$, where $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. 2. Given a 2Œ2 matrix, determine whether it is unitary. 3. Show that multiplying a state by a global phase does not affect measurement probabilities.

2.6 Homework

1. For an arbitrary normalized qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, compute $\langle\psi|\psi\rangle$ and confirm that it equals 1. 2. Read a short section on Dirac notation from your main textbook and summarize key points in half a page.

2.7 Python/Qiskit Coding

```
import numpy as np

# Define basis states
zero = np.array([[1], [0]], dtype=complex)
one  = np.array([[0], [1]], dtype=complex)

# Define Hadamard
H = (1/np.sqrt(2)) * np.array([[1, 1],
                                [1, -1]], dtype=complex)

# Verify unitarity: H H
identity_check = H.conj().T @ H
print("H^ H =")
print(identity_check)
```

2.8 Suggested Figures

- A diagram showing bras and kets as row and column vectors.
- Illustration of orthonormal basis vectors (like x and y axes in 2D).
- A box summarizing properties of unitary and Hermitian matrices.

---

3 Session 3: Qubits and the Bloch Sphere

3.1 Learning Objectives

By the end of this session, students should be able to:

- Describe the Bloch sphere representation of a single qubit.
- Map a general qubit state to Bloch sphere coordinates (,).
- Understand the geometric interpretation of quantum states.
- Relate rotations on the Bloch sphere to quantum gates.

3.2 Core Concepts / Outline

- General form of a 1-qubit state:

$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$.

- Parametrization:

$|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle$.

- Bloch sphere: mapping to (x, y, z) on the unit sphere.
- Poles and equator: |0, |1, |+, |.
- Geometric action of single-qubit gates as rotations.

3.3 Theory Intuition

Any normalized qubit state can be written as:

$$|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle,$$

for some real angles $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi)$, ignoring global phase. We define Bloch coordinates:

$$x = \sin\theta\cos\phi, \quad y = \sin\theta\sin\phi, \quad z = \cos\theta.$$

These coordinates describe a point on the **unit sphere**, called the **Bloch sphere**.

Common states:

- $|0\rangle$: north pole (0, 0, 1)

- $|1\rangle$: south pole (0, 0, 1)
- $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$: point on the +x axis
- $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$: point on the x axis

Single-qubit gates effect rotations of this Bloch vector.

3.4 Example

Take $|\psi\rangle = (|0\rangle + i|1\rangle)/\sqrt{2}$. Express $|\psi\rangle$ in the parametrized form and find its Bloch sphere coordinates.

3.5 In-Class Exercises

1. Write down Bloch sphere coordinates for $|0\rangle$, $|1\rangle$, $|+\rangle$, $|-\rangle$, and $(|0\rangle + i|1\rangle)/\sqrt{2}$. 2. Discuss intuitively what applying H, X, Z does to points on the Bloch sphere.

3.6 Homework

1. Prove that every pure qubit state corresponds to exactly one point on the Bloch sphere (up to global phase). 2. Derive the expression for x, y, z in terms of and .

3.7 Python/Qiskit Coding

```
from qiskit.visualization import plot_bloch_vector
import matplotlib.pyplot as plt
import numpy as np

# Example: |+> state
theta = np.pi/2
phi = 0
x = np.sin(theta) * np.cos(phi)
y = np.sin(theta) * np.sin(phi)
z = np.cos(theta)

fig = plot_bloch_vector([x, y, z])
plt.show()
```

3.8 Suggested Figures

- **Bloch sphere diagram**:
- North pole labeled |0, south pole |1.
- Equator showing |+ and |.
- A generic state | with angles and indicated.
- **Gate action sketches**:
- X gate: 180ř rotation around x-axis.
- Z gate: 180ř rotation around z-axis.

4 Session 4: Single-Qubit Gates and Rotations

4.1 Learning Objectives

By the end of this session, students should be able to:

- Identify and use standard single-qubit gates (X, Y, Z, H, S, T).
- Represent these gates as matrices and understand their effects.
- Relate certain gates to rotations on the Bloch sphere.
- Build basic 1-qubit circuits and reason about their outputs.

4.2 Core Concepts / Outline

- Pauli gates: X, Y, Z.
- Hadamard gate H and its role in creating superposition.
- Phase gates S and T.
- Rotation gates: Rx(), Ry(), Rz().
- Gate identities and simple compositions (e.g., HZH = X).

4.3 Theory Intuition

Single-qubit gates correspond to **unitary 2Œ2 matrices**. For example:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

The Hadamard gate creates superposition:

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

Rotation gates such as $R_x(\theta)$ correspond to rotations of the Bloch sphere around a given axis by angle .

4.4 Example

Verify the identity:

$$HZH = X.$$

Compute HZH and show it equals the X matrix.

4.5 In-Class Exercises

1. Apply H followed by X to $|0\rangle$ and compute the resulting state. 2. Compute the effect of S and T on the states |+ and |. 3. Discuss how a rotation gate Rx(/2) moves a state on the Bloch sphere.

4.6 Homework

1. Prove that the Pauli matrices X, Y, Z are Hermitian and unitary. 2. Show that H$^2 = I.3.Constructasequenceofgatesthatmaps|0to|1viaanintermediatesuperposition.$

4.7 Python/Qiskit Coding

```
from qiskit import QuantumCircuit

qc = QuantumCircuit(1, 1)

# Prepare |0>, apply H then Z then H
qc.h(0)
qc.z(0)
qc.h(0)
qc.measure(0, 0)

print(qc)
```

4.8 Suggested Figures

- Matrix table listing X, Y, Z, H, S, T.
- Bloch sphere diagrams showing how X, Y, Z act as 180ř rotations around axes.
- A simple circuit diagram illustrating sequences like H Z H.

---

5 Session 5: Multi-Qubit States, Entanglement and Bell States

5.1 Learning Objectives

By the end of this session, students should be able to:

- Work with multi-qubit systems using tensor products.
- Describe entangled states, especially Bell states.
- Understand the difference between separable and entangled states.
- Construct simple entangling circuits using CNOT and H gates.

5.2 Core Concepts / Outline

- Tensor products of Hilbert spaces.
- Computational basis for multiple qubits (|00, |01, |10, |11, ...).
- Separable vs. entangled states.
- Bell states and their preparation circuits.
- Two-qubit gates: CNOT, CZ, SWAP.

5.3 Theory Intuition

For two qubits, the state space is 4-dimensional. The basis states are:

$$|00\rangle, |01\rangle, |10\rangle, |11\rangle.$$

A state is **separable** if it can be written as $|\psi\rangle_A \otimes |\phi\rangle_B$. States that cannot be written this way are **entangled**.

The **Bell states** are maximally entangled two-qubit states, such as:

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}.$$

5.4 Example

Prepare the Bell state $|\Phi^+\rangle$ by: 1. Starting with |00. 2. Applying H on qubit 0. 3. Applying CNOT with control qubit 0 and target qubit 1.

5.5 In-Class Exercises

1. Show that $|\Phi^+\rangle$ is not separable. 2. Write the states for other Bell states: $|\Phi^-\rangle$, $|\Psi^+\rangle$, $|\Psi^-\rangle$. 3. Use the CNOT gate to create different Bell states by changing the input.

5.6 Homework

1. Prove that no single-qubit gates acting independently on each qubit can create entanglement starting from a product state. 2. Compute measurement correlations for Bell states in the computational basis.

5.7 Python/Qiskit Coding

```
from qiskit import QuantumCircuit, Aer, execute

qc = QuantumCircuit(2, 2)
qc.h(0)       # Create superposition on qubit 0
qc.cx(0, 1)   # Entangle qubit 0 and 1
qc.measure([0,1], [0,1])

backend = Aer.get_backend('qasm_simulator')
result = execute(qc, backend, shots=1024).result()
print("Counts:", result.get_counts())
```

5.8 Suggested Figures

- Circuit diagram for generating |:

  ```
  q0: H

  q1: X
  ```

- Diagram showing entangled pairs as linked qubits.
- Table comparing separable and entangled states.

6 Session 6: Measurement, Born Rule and Noise

6.1 Learning Objectives

By the end of this session, students should be able to:

- Explain the Born rule and measurement outcomes.
- Describe projective measurements in the computational basis.
- Understand the concept of decoherence and basic noise models.
- Use Qiskit to simulate noisy quantum circuits.

6.2 Core Concepts / Outline

- Born rule for measurement probabilities.
- Projective measurements and post-measurement states.
- Measurement in different bases.
- Noise channels: bit-flip, phase-flip, depolarizing (conceptual).
- Decoherence and T1/T2 times (high level).

6.3 Theory Intuition

Measurement in quantum mechanics is probabilistic. For state:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

measuring in the computational basis yields 0 with probability $|\alpha|^2$ and 1 with probability $|\beta|^2$. After measurement, the state collapses to the corresponding basis state.

Real quantum systems are open and interact with their environment, causing **decoherence**. Noise can be modeled with channels like bit-flip or depolarizing channels.

6.4 Example

Simulate repeated measurements of the state (|0 + |1)/2 and show that frequencies approach 5050 as shots increase.

6.5 In-Class Exercises

1. Given a state, compute measurement probabilities and post-measurement states. 2. Discuss qualitatively how decoherence affects superpositions and entanglement.

6.6 Homework

1. Read about T1 and T2 times for qubits and summarize in half a page. 2. Consider a simple noise model (bit-flip with probability p) and reason its impact on a single-qubit state.

6.7 Python/Qiskit Coding

```
from qiskit import QuantumCircuit, Aer, execute

qc = QuantumCircuit(1,1)
qc.h(0)
qc.measure(0,0)

backend = Aer.get_backend('qasm_simulator')
result = execute(qc, backend, shots=2048).result()
print(result.get_counts())
```

(Optionally extend with Qiskit Aer noise models if desired.)

6.8 Suggested Figures

- Probability tree diagrams showing measurement outcomes.
- Bloch sphere with arrows showing shrinking of the vector due to decoherence.
- Circuit with measurement symbol at the end of a wire.

---

7 Session 7: Quantum Circuit Model and Universality

7.1 Learning Objectives

By the end of this session, students should be able to:

- Describe the quantum circuit model of computation.
- Understand how complex unitaries are built from basic gate sets.
- Know what it means for a gate set to be universal.
- Express simple algorithms as quantum circuits.

7.2 Core Concepts / Outline

- Quantum circuit as a sequence of gates.
- Wires as qubits, boxes as gates, measurement symbols.
- Universality and standard gate sets H, T, CNOT etc.
- Circuit depth and width.
- Decomposition of multi-qubit unitaries into 1- and 2-qubit gates (conceptual).

7.3 Theory Intuition

The **circuit model** is the central paradigm of quantum computation. It is analogous to classical logic circuits but with unitary gates operating on qubits. A finite set of gates is **universal** if any unitary can be approximated by some circuit built from those gates.

7.4 Example

Show that H, T, CNOT is a universal gate set (conceptually: H and T generate arbitrary single-qubit unitaries, plus CNOT for entanglement).

7.5 In-Class Exercises

1. Draw the circuit for a simple algorithm (e.g., preparing a Bell state). 2. Discuss how you would implement a 3-qubit unitary using only 1- and 2-qubit gates.

7.6 Homework

1. Read a section about universality in your main text and write a summary. 2. Sketch a circuit that takes |00 to (|00 + |11)/2 and then measures both qubits.

7.7 Python/Qiskit Coding

```
from qiskit import QuantumCircuit

qc = QuantumCircuit(2, 2)
qc.h(0)
qc.cx(0, 1)
qc.measure([0,1],[0,1])
print(qc.draw())
```

7.8 Suggested Figures

- General block diagram of a quantum circuit: input state sequence of gates measurement.
- Example circuits for Bell state and small algorithms.

---

8 Session 8: Deutsch and DeutschJozsa Algorithms

8.1 Learning Objectives

By the end of this session, students should be able to:

- State the Deutsch problem and the DeutschJozsa problem.
- Describe the oracle model of computation.
- Implement the DeutschJozsa algorithm as a circuit.
- Understand the advantage over classical deterministic algorithms.

8.2 Core Concepts / Outline

- Oracle functions f: 0,$1^n 0, 1$.

- Constant vs. balanced functions.
- Deutsch algorithm (n = 1 case).
- DeutschJozsa algorithm (general n).
- Query complexity and quantum advantage.

8.3 Theory Intuition

The DeutschJozsa algorithm determines whether a function f is constant or balanced with **one oracle query** in the quantum model, whereas any deterministic classical algorithm may require up to $2^{n-1} + 1$ queries.

The algorithm: 1. Prepare |0...0|1. 2. Apply H to all qubits. 3. Apply oracle U$_f .4. ApplyHagaintothefirstnqubi$

8.4 Example

Implement the Deutsch algorithm for a single-bit function and verify that measurement outcomes distinguish constant and balanced functions.

8.5 In-Class Exercises

1. Write down the circuit for DeutschJozsa with n = 2. 2. Discuss the number of oracle queries required in quantum vs. classical settings.

8.6 Homework

1. Show mathematically that constant functions yield measurement outcome |0...0 in the DeutschJozsa algorithm. 2. Implement a 2-qubit version in Qiskit with a chosen oracle.

8.7 Python/Qiskit Coding

```
from qiskit import QuantumCircuit, Aer, execute

def deutsch_oracle(is_constant):
    qc = QuantumCircuit(2)
    if is_constant:
        qc.x(1)  # Example constant oracle
    else:
        qc.cx(0,1)  # Example balanced oracle
    return qc.to_gate(label="Uf")

qc = QuantumCircuit(2,1)
qc.x(1)
qc.h([0,1])

oracle = deutsch_oracle(is_constant=False)
qc.append(oracle, [0,1])
```

```
qc.h(0)
qc.measure(0,0)

backend = Aer.get_backend('qasm_simulator')
result = execute(qc, backend, shots=1024).result()
print(result.get_counts())
```

8.8 Suggested Figures

- Circuit diagram for Deutsch algorithm (n=1).
- Circuit diagram for DeutschJozsa (n general) with oracle block U_f.
- Table showing constant vs. balanced examples.

---

9 Session 9: Grovers Search Algorithm

9.1 Learning Objectives

By the end of this session, students should be able to:

- Explain the unstructured search problem.
- Understand the idea of amplitude amplification.
- Describe the Grover iterate (oracle + diffusion).
- Implement Grovers algorithm for small N using Qiskit.

9.2 Core Concepts / Outline

- Search problem over N = $2^n items.$
- Oracle marking the good state(s).
- Grover operator G = (2|ss| I) ů (I 2||).
- Optimal number of iterations O(N).
- Behavior of amplitudes over iterations.

9.3 Theory Intuition

Grovers algorithm provides a **quadratic speedup** over classical search. It repeatedly rotates the state vector in the 2D subspace spanned by the marked state and the uniform superposition, increasing the amplitude of the target state.

9.4 Example

Apply Grovers algorithm to a 2-qubit system (N = 4) with exactly one marked element, and track the amplitudes by hand or numerically.

9.5 In-Class Exercises

1. For N = 4, compute how many Grover iterations are optimal. 2. Analyze how amplitudes change during one full Grover iteration.

9.6 Homework

1. Implement Grovers algorithm in Qiskit for N = 4 and verify success probabilities. 2. Write a short explanation of why the algorithm fails if you over-iterate.

9.7 Python/Qiskit Coding

```
from qiskit import QuantumCircuit, Aer, execute

# Example: Grover for 2-qubit search space with state |11> marked
qc = QuantumCircuit(2,2)

# Step 1: prepare uniform superposition
qc.h([0,1])

# Step 2: oracle marking |11>
qc.cz(0,1)  # phase flip for |11>

# Step 3: diffusion operator
qc.h([0,1])
qc.x([0,1])
qc.h(1)
qc.cx(0,1)
qc.h(1)
qc.x([0,1])
qc.h([0,1])

qc.measure([0,1],[0,1])

backend = Aer.get_backend('qasm_simulator')
result = execute(qc, backend, shots=1024).result()
print(result.get_counts())
```

9.8 Suggested Figures

- Geometric picture: rotation in a 2D subspace between |s and |.
- Circuit diagram for Grovers algorithm with oracle block and diffusion block.
- Plot of success probability vs. number of iterations (conceptual).

---

10 Session 10: Shors Algorithm and Period Finding

10.1 Learning Objectives

By the end of this session, students should be able to:

- Understand the high-level structure of Shors algorithm.
- Explain the role of period finding and modular exponentiation.
- Recognize the use of the Quantum Fourier Transform (QFT).
- Describe why Shors algorithm threatens classical public-key cryptography.

10.2 Core Concepts / Outline

- Integer factorization problem.
- Period finding as the core quantum subroutine.
- Modular exponentiation circuit (black-box view).
- QFT and its role in extracting the period.
- Classical post-processing to obtain factors.

10.3 Theory Intuition

Shors algorithm factors a composite integer N in polynomial time, using a quantum subroutine to find the period r of the function:

$$f(x) = a^x \mod N.$$

The quantum part prepares a superposition over x, computes f(x) in a second register, and uses the QFT to extract information about r. Classical algorithms then compute the gcd to derive non-trivial factors of N.

10.4 Example

Work through a toy example of factoring N = 15 with a chosen base a (e.g., a = 2), tracing the period-finding idea (without fully building the full circuit).

10.5 In-Class Exercises

1. Given N and a, compute f(x) = a$^{x} modNforsmallxandguesstheperiodr.2.Discussqualitativelywhyperiodfin$

10.6 Homework

1. Read a simplified explanation of Shors algorithm from a textbook or online source and write a 1-page summary. 2. Implement the QFT on a small number of qubits (3 or 4) in Qiskit as a separate exercise (see next session).

10.7 Python/Qiskit Coding

(Focus on QFT building blocks, see Session 11 for concrete code.)

10.8 Suggested Figures

- Block diagram of Shors algorithm showing classical and quantum components.
- Illustration of the function f(x) = a$^x modNasaperiodicsignal.$
- Circuit diagram showing registers and QFT block (high level).

---

11 Session 11: Quantum Fourier Transform (QFT)

11.1 Learning Objectives

By the end of this session, students should be able to:

- Define the Quantum Fourier Transform (QFT).
- Write down the QFT circuit for n qubits.
- Implement QFT and its inverse in Qiskit.
- Understand the role of QFT in algorithms like Shors and phase estimation.

11.2 Core Concepts / Outline

- Definition of QFT on N = $2^n states.$
- Decomposition of QFT into Hadamard and controlled-phase gates.
- Complexity O(n$^2) of QFT circuit.$
- Inverse QFT and its implementation.

11.3 Theory Intuition

The QFT on N = $2^n basis states is defined by :$ QFT$|x\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ixk/N} |k\rangle$.The circuit can be decomposed into a sequence of H and controlled-phase gates, along with swaps to reverse qubit order.

11.4 Example

Write the QFT on 2 qubits explicitly and compare with the matrix decomposition.

11.5 In-Class Exercises

1. Draw the QFT circuit for n = 3 qubits. 2. Discuss how to implement the inverse QFT by reversing the order of gates and conjugating phase angles.

11.6 Homework

1. Implement QFT for n = 3 qubits in Qiskit and test it on computational basis states. 2. Explore approximating QFT by omitting some small-angle controlled-phase gates.

11.7 Python/Qiskit Coding

```
from qiskit import QuantumCircuit
import numpy as np

def qft(n):
    qc = QuantumCircuit(n)
    for j in range(n):
        qc.h(j)
        for k in range(j+1, n):
            qc.cp(np.pi / (2**(k-j)), k, j)
    # Swap qubits to reverse order
    for i in range(n//2):
        qc.swap(i, n-1-i)
    return qc

qc = qft(3)
print(qc.draw())
```

11.8 Suggested Figures

- Circuit diagram of QFT for 3 qubits.
- Table comparing classical FFT and quantum QFT (structure and complexity).

---

12 Session 12: Quantum Error Correction (QEC)

12.1 Learning Objectives

By the end of this session, students should be able to:

- Explain why quantum error correction is necessary.
- Describe the basic idea behind redundancy and encoding.
- Understand simple codes such as the 3-qubit bit-flip code conceptually.
- Recognize the notions of syndrome measurement and recovery.

12.2 Core Concepts / Outline

- Fragility of quantum information and no-cloning theorem.
- Bit-flip and phase-flip errors.
- Encoding logical qubits into multiple physical qubits.
- Syndrome measurement using ancilla qubits.
- High-level description of Shor code or Steane code.

12.3 Theory Intuition

Quantum error correction encodes a logical qubit into multiple physical qubits, so that errors affecting a subset of the qubits can be detected and corrected. Although we cannot measure the state directly without collapsing it, we can measure **syndromes**parity checks that reveal where an error occurred without revealing the logical state.

12.4 Example

Discuss the 3-qubit bit-flip code which encodes:

$$|0_L\rangle = |000\rangle, \quad |1_L\rangle = |111\rangle.$$

Explain how majority voting can correct a single bit-flip error.

12.5 In-Class Exercises

1. Show how the bit-flip code corrects an X error on any one qubit. 2. Discuss why phase-flip errors also need to be corrected, and outline the phase-flip code.

12.6 Homework

1. Read about Shors 9-qubit code and summarize its structure. 2. Write a conceptual explanation of syndrome measurement without collapsing the logical state.

12.7 Python/Qiskit Coding

(Qiskit Ignis or newer tools may be used; for simplicity, you may just simulate errors and majority voting in Python.)

12.8 Suggested Figures

- Encoding diagram for the 3-qubit bit-flip code.
- Syndrome measurement circuit sketch.
- Conceptual block diagram of an error-correcting cycle.

---

13 Session 13: Variational Quantum Algorithms: VQE QAOA

13.1 Learning Objectives

By the end of this session, students should be able to:

- Explain the idea behind variational (hybrid) quantum-classical algorithms.
- Describe the Variational Quantum Eigensolver (VQE) and its workflow.
- Understand QAOA as an example of a variational algorithm for optimization.
- Implement a simple VQE-like circuit in Qiskit.

13.2 Core Concepts / Outline

- NISQ era and limitations of deep circuits.
- Parameterized quantum circuits (ansätze).
- Classical optimizer loop (gradient-free or gradient-based).
- VQE for approximating ground-state energies.
- QAOA for combinatorial optimization.

13.3 Theory Intuition

Variational algorithms use parameterized circuits:

$$|\psi(\vec{\theta})\rangle = U(\vec{\theta})|0...0\rangle,$$

and a classical optimizer that updates parameters $\vec{\theta}$ to minimize a cost function, often an expectation value of a Hamiltonian.

13.4 Example

Build a simple 1- or 2-qubit variational circuit with a few rotation gates and use a classical optimizer to minimize the expectation of Z on one qubit.

13.5 In-Class Exercises

1. Sketch the workflow of VQE (quantum subroutine + classical optimizer). 2. Discuss why variational circuits can be more noise-resilient than deep non-parameterized circuits.

13.6 Homework

1. Implement a simple variational circuit in Qiskit and manually tune parameters to minimize a simple cost. 2. Read a short article or paper on VQE and summarize it.

13.7 Python/Qiskit Coding

```
from qiskit import QuantumCircuit, Aer, execute
import numpy as np

def ansatz(theta):
    qc = QuantumCircuit(1,1)
    qc.ry(theta, 0)
    qc.measure(0,0)
    return qc

backend = Aer.get_backend('qasm_simulator')

def expectation(theta):
    qc = ansatz(theta)
    result = execute(qc, backend, shots=1024).result()
    counts = result.get_counts()
    # Expectation of Z: P(0) - P(1)
```

```
p0 = counts.get('0', 0) / 1024
p1 = counts.get('1', 0) / 1024
return p0 - p1
```

13.8 Suggested Figures

- Workflow diagram: quantum device classical optimizer loop.
- Simple parameterized circuit with RY() gates.
- Conceptual figure illustrating a cost landscape and parameter updates.

---

14 Session 14: Quantum Hardware and NISQ Era

14.1 Learning Objectives

By the end of this session, students should be able to:

- Identify main physical platforms for quantum hardware.
- Describe superconducting qubits, trapped ions, and photonic qubits at a high level.
- Understand what NISQ means and its implications.
- Recognize hardware constraints: decoherence times, gate errors, connectivity.

14.2 Core Concepts / Outline

- Superconducting qubits (e.g., transmons).
- Trapped ion qubits.
- Photonic qubits.
- NISQ devices: noisy, intermediate-scale quantum.
- Hardware metrics: T1, T2, gate fidelity, connectivity graphs.

14.3 Theory Intuition

Quantum hardware implementations differ in:

- How qubits are physically realized.
- How gates are implemented (microwave pulses, laser interactions, optical elements).
- Their strengths and weaknesses: scalability, coherence times, gate speeds.

The NISQ era is characterized by devices with tens to a few hundred qubits, limited coherence, and noisetoo small and noisy for full fault-tolerant computation, but large enough to explore interesting algorithms.

14.4 Example

Compare superconducting qubits and trapped ions in terms of:

- Typical coherence times.
- Gate speed.
- Connectivity between qubits.

14.5 In-Class Exercises

1. Students research one hardware platform and present a short summary. 2. Discuss what kinds of algorithms might be feasible on NISQ hardware.

14.6 Homework

1. Choose a specific quantum hardware vendor or platform (e.g., IBM, IonQ, Xanadu) and write a short report about their technology. 2. Explain why error correction is challenging yet essential for scalable quantum computers.

14.7 Python/Qiskit Coding

```
from qiskit import IBMQ

# (If using IBM Quantum Experience)
# IBMQ.load_account()
# provider = IBMQ.get_provider()
# backend = provider.get_backend('ibmq_qasm_simulator')
# print(backend.configuration())
```

14.8 Suggested Figures

- Comparison table of different hardware platforms.
- Conceptual diagram of a superconducting qubit circuit or ion trap.
- Illustration of a connectivity graph for a real device.

---

15 Session 15: Final Project Design and Presentations

15.1 Learning Objectives

By the end of this session, students should be able to:

- Design a small quantum algorithm or experiment from scratch.
- Translate a problem statement into a quantum circuit and/or variational ansatz.
- Implement, simulate, and analyze results using Qiskit.
- Present their project clearly and critically discuss limitations.

15.2 Core Concepts / Outline

- Project design process:
- Choose a problem.
- Select appropriate quantum model/algorithm.
- Design circuit and simulation strategy.
- Analysis and interpretation of results.
- Limitations due to noise, qubit count, and depth.
- Future directions and open questions.

15.3 Theory Intuition

This is a synthesis session where students bring together:

- Theoretical understanding of qubits, gates, circuits, and algorithms.
- Practical skills in Qiskit and simulation.
- Awareness of hardware constraints and noise.

Projects can be algorithm-focused (e.g., implementing Grover or VQE for a particular toy problem) or concept-focused (e.g., exploring the effect of noise on entanglement).

15.4 Example Project Ideas

- Implement Grovers algorithm for a specific search problem of size N = 8 or 16.
- Build a small VQE circuit to approximate the ground-state energy of a 2-qubit Hamiltonian.
- Explore decoherence by applying simple noise models and measuring entanglement decay.

15.5 In-Class Activities

- Students present short project proposals and receive feedback.
- Work in small groups to refine algorithms and circuits.
- Live debugging of circuits and code with instructor guidance.

15.6 Homework / Final Assignment

- Complete a mini-project:
- 48 pages report in LaTeX or Markdown, including motivation, method, results, and discussion.
- Qiskit code as an appendix or GitHub repository.
- Prepare a short presentation (510 minutes) summarizing the project.

15.7 Python/Qiskit Coding

(Project-dependent; students are expected to integrate all previous knowledge and examples.)

15.8 Suggested Figures

- Flowchart of quantum project workflow: Problem Algorithm Circuit Simulation Results.
- Circuit diagrams from students projects (e.g., Grover, VQE, Bell tests).
- Plots of measurement statistics or cost function vs. iteration for variational algorithms.
