UNIT-1

FUNDAMENTALS OF QUANTUM MECHANICAL & LINEAR ALGEBRA:

1.CLASSICAL V/S QUANTUM COMPUTATION ?

Classical Computer	Quantum Computer
It is large scale integrated multi-purpose computer(CPU)	It is high speed parallel computer based on quantum mechanics.
Information storage is bit based on voltage/charge etc.	Information storage is Quantum bit based on direction of an electron spin.
Computer runs on bits that have a value of either 0 or 1.	Quantum bits or "qubits" are similar in that for practical purposes we read them as a value of 0 or 1, but they can also hold much more complex information, or even be negative values. Before we read their value they are in an indeterminate state called superposition and can be influenced by other qubits(this is called entanglement). Qubits can be
Discrete number of possible states: 0 or 1. Deterministic: repeated computations on the same input will lead to the same output.	Infinite (continuous) number of possible states. Probabilistic: measurements on superposed states yield probabilistic answers (our confidence in these answers builds up through repeated computations) then reduced to 0 or 1.
Information processing is carried out by logic gates e.g. NOT, AND, OR etc in sequential basis	Information processing is carried out by Quantum logic gates in parallel basis
Only specifically defined results are available, inherently limited by an algorithm's design	Quantum answers (which are in quantity called amplitudes) are probabilistic, meaning that because of superposition and entanglement multiple possible answers are considered in a given computation.
Circuit behaviour is governed by classical physics.	Circuit behaviour is governed explicitly by quantum mechanics.
Operations are defined by Boolean Algebra.	Operations are defined by linear algebra over Hilbert Space and can be represented by unitary matrices with complex elements.
No restrictions exist on copying or measuring signals	Severe restrictions exist on copying and measuring signals
Circuits are easily implemented in fast, scalable and macroscopic technologies such as CMOS.	Circuits must use microscopic technologies that are slow, fragile and not yet scalable e.g. NMR (Nuclear magnetic resonance).

2.Complex Numbers ?

1. Definition

A complex number is a number of the form:

$z = a + bi$

Where:

a = real part

b = imaginary part

$i = \sqrt{(-1)}$ (imaginary unit)

✓ Example:

$z = 3 + 4i$

2. Real and Imaginary Parts

Real part: $Re(z) = a$

Imaginary part: $Im(z) = b$

✅ Example:

For $z = 5 + 2i \rightarrow Re(z) = 5, Im(z) = 2$

3. Modulus (Magnitude)

The modulus of a complex number is:

$|z| = \sqrt{(a^2 + b^2)}$

✅ Example:

$z = 3 + 4i \rightarrow |z| = \sqrt{(3^2 + 4^2)} = 5$

4. Conjugate of a Complex Number

The conjugate of $z = a + bi$ is:

$\bar{z} = a - bi$

✅ Example:

$z = 3 + 4i \rightarrow \bar{z} = 3 - 4i$

5. Arithmetic Operations

Addition

$(a + bi) + (c + di) = (a + c) + (b + d)i$

Subtraction

$(a + bi) - (c + di) = (a - c) + (b - d)i$

Multiplication

$(a + bi)(c + di) = (ac - bd) + (ad + bc)i$

Division

Multiply numerator and denominator by conjugate

✅ Example:

$(1 + i)(1 - i) = 1 - i^2 = 2$

6. Polar Form

Complex numbers can be written as:

$z = r(\cos\theta + i \sin\theta)$

Where:

r = modulus

θ = angle (argument)

7. Euler's Form

Using Euler's formula:

$z = re^{(i\theta)}$

✅ Example:

$e^{(i\pi)} + 1 = 0$

8. Argand Plane

Complex numbers are represented on a 2D plane:

X-axis → Real part

Y-axis → Imaginary part

9. Importance in Quantum Computing

Used to represent probability amplitudes
Essential for describing quantum states
Helps in calculations involving interference and superposition

3. VECTORS?

1. Definition of Vectors
   A vector is an ordered list of numbers.
   Represented as a column or row matrix.
   Used to represent quantum states.
   ✓ Example:
2. Types of Vectors
   Row vector: [a b c]
   Column vector:
3. Vector Operations
   Addition
   Add corresponding elements
   Scalar Multiplication
   Multiply each element by a scalar
   ✓ Example:
   2 × [?] = [?]
4. Definition of Matrices
   A matrix is a rectangular arrangement of numbers in rows and columns.
   Used to represent quantum gates and transformations.
   ✓ Example:
5. Types of Matrices
   Square matrix: Same number of rows and columns
   Identity matrix: Diagonal elements are 1
   Zero matrix: All elements are 0
   Unitary matrix: Important in quantum computing (preserves probability)

4. MATRIX OPERATIONS?

1.Addition
   Add corresponding elements
   Multiplication
   Row × Column multiplication rule
   ✓ Example:
2. Matrix–Vector Multiplication
   A matrix acting on a vector transforms it into another vector.
   This represents quantum gate operations.
3. Eigenvalues and Eigenvectors

Special vectors that do not change direction under transformation.
Important in quantum mechanics.

✓ Example:

$A\ v = \lambda v$

4. Importance in Quantum Computing

Vectors represent quantum states (qubits)
Matrices represent quantum gates
Operations on qubits are done using matrix multiplication

5. HILBERT SPACES AND DIRAC NOTATIONS ?

1. Hilbert Space – Definition

A Hilbert Space is a complete vector space with an inner product.
It is used to describe all possible quantum states of a system.
It can be finite or infinite dimensional.

✓ Example:

A single qubit exists in a 2-dimensional Hilbert space.

2. Properties of Hilbert Space

Supports vector addition and scalar multiplication
Has an inner product (used to find probabilities)
Is complete (all limits exist within the space)
Vectors can be normalized (length = 1)

3. Basis States

A set of vectors that can represent all states in the space.
For qubits, basis states are:
$|0\rangle$
$|1\rangle$

✓ Example:

Any qubit state:
$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

4. Dirac Notation – Introduction

A compact way to represent quantum states.
Also called Bra–Ket notation.
Widely used in quantum mechanics.

5. Ket Notation ($|\psi\rangle$)

Represents a column vector (state vector).

✓ Example:

$|0\rangle$ = ,
$|1\rangle$ =

6. Bra Notation ($\langle\psi|$)

Represents a row vector (conjugate transpose of ket).

✓ Example:
⟨0| = [1 0], ⟨1| = [0 1]

7. Inner Product (⟨φ|ψ⟩)
   Represents the overlap between two states.
   Gives probability amplitude.
   ✓ Example:
   ⟨0|1⟩ = 0 (orthogonal states)
8. Outer Product (|ψ⟩⟨φ|)
   Produces a matrix (operator).
   Used to represent quantum operations.
9. Importance in Quantum Computing
   Hilbert space defines where quantum states exist.
   Dirac notation simplifies representation and calculations.
   Used in quantum algorithms, gates, and measurements.

6. QUANTUM STATES AND QUBITS ?

1. Definition of Qubit
   A qubit (quantum bit) is the basic unit of quantum information.
   Unlike a classical bit (0 or 1), a qubit can exist in multiple states simultaneously.
   ✓ Example:
   A classical bit = 0 or 1
   A qubit = $\alpha|0\rangle + \beta|1\rangle$
2. Quantum State
   A quantum state describes the condition of a qubit.
   Represented as a linear combination of basis states $|0\rangle$ and $|1\rangle$.
   Coefficients (α, β) are complex numbers.
3. Normalization Condition
   The total probability must be 1:
   $|\alpha|^2 + |\beta|^2 = 1$
   ✓ Example:
   If $\alpha = 1/\sqrt{2}$ and $\beta = 1/\sqrt{2} \rightarrow$ valid quantum state
4. Superposition
   A qubit can exist in both $|0\rangle$ and $|1\rangle$ at the same time.
   This property enables parallel computation.
   ✓ Example:
   $|\psi\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$
5. Measurement
   Measurement collapses the quantum state into either 0 or 1.

Result is probabilistic:
Probability of 0 = $|\alpha|^2$
Probability of 1 = $|\beta|^2$

6. Representation of Qubit
   Represented using vectors (Dirac notation):
7. Bloch Sphere Representation
   A qubit can also be represented as a point on a Bloch sphere.
   Shows all possible states of a qubit.
8. Multiple Qubits
   Two or more qubits form a combined system.
   Represented using tensor products.
   ✓ Example:
   $|00\rangle, |01\rangle, |10\rangle, |11\rangle$
9. Entanglement
   Special property where qubits are correlated.
   State of one qubit depends on another.
   ✓ Example:
   $(|00\rangle + |11\rangle)/\sqrt{2}$
10. Importance in Quantum Computing
    Qubits are the foundation of quantum computers.
    Enable powerful concepts like superposition and entanglement.
    Used in quantum algorithms and computations.

7. SUPERPOSITION AND MEASURMENT ?

1. Superposition – Definition
   Superposition is a fundamental property of quantum systems.
   A qubit can exist in multiple states at the same time.
   It is a combination of basis states $|0\rangle$ and $|1\rangle$.
   ✓ Example:
2. Coefficients (Amplitudes)
   α and β are complex probability amplitudes.
   They determine the probability of each state.
3. Normalization Condition
   Total probability must be 1:
   ✓ Example:
   If $\alpha = 1/\sqrt{2}$ and $\beta = 1/\sqrt{2} \rightarrow$ equal probability for 0 and 1
4. Creating Superposition
   Achieved using quantum gates like Hadamard gate (H).
   ✓ Example:
   $|0\rangle \rightarrow (|0\rangle + |1\rangle)/\sqrt{2}$ after applying H gate
5. Advantages of Superposition

Allows parallel computation
Enables quantum speedup
Basis for quantum algorithms

6. Measurement – Definition
   Measurement is the process of observing a quantum state.
   It collapses the superposition into a single classical value (0 or 1).
7. Measurement Probabilities
   Probability of getting:
   $0 = |\alpha|^2$
   $1 = |\beta|^2$
   ✓ Example:
   If $|\psi\rangle = (|0\rangle + |1\rangle)/\sqrt{2} \rightarrow$
   50% chance of 0
   50% chance of 1
8. Collapse of Quantum State
   After measurement, the qubit loses superposition.
   It becomes a definite state.
9. Repeated Measurements
   Measuring many times gives a probability distribution.
   ✓ Example:
   Running 1000 times → ~500 results of 0 and ~500 of 1
10. Importance in Quantum Computing
    Superposition enables powerful computations.
    Measurement extracts useful information.
    Both are essential for quantum algorithms.

8. TENSOR PRODUCT AND MULTI QUBIT SYSTEM ?

1. Tensor Product – Definition
   The tensor product is a mathematical operation used to combine multiple qubits.
   It creates a larger state space from individual qubits.
   Represented by the symbol ⊗.
2. Purpose of Tensor Product
   Used to represent multi-qubit systems.
   Combines individual qubit states into a single joint state.
3. Tensor Product of Basis States
   For two qubits:

✓ Example:

4. Matrix Representation

Tensor product of vectors results in a larger vector.

✓ Example:

5. Multi-Qubit System

A system with more than one qubit.

Represented using tensor products.

6. State Space Growth

Number of states increases exponentially:

1 qubit → 2 states

2 qubits → 4 states

n qubits → 2^n states

7. General Multi-Qubit State

A two-qubit system is written as:

Coefficients must satisfy normalization condition.

8. Entanglement in Multi-Qubit Systems

Some states cannot be separated into individual qubits.

These are called entangled states.

✓ Example:

9. Importance in Quantum Computing

Tensor product helps build complex quantum systems.

Multi-qubit systems enable powerful computations.

Essential for quantum algorithms and circuits.

10. Conclusion

Tensor product is used to combine qubits into a single system.

Multi-qubit systems grow exponentially in power.

They enable advanced features like entanglement and parallelism.

UNIT-2
QUANTUM LOGIC GATES AND CIRCUITS

1.PAULI GATES?

Pauli-X Gate (X Gate)

Also called Quantum NOT gate because it flips the qubit.

Example

Input qubit: $|0\rangle$

Apply X:

Input qubit: $|1\rangle$

Apply X:

✓Meaning

It flips $0 \leftrightarrow 1$, same as classical NOT, but also works on superposition.

✓Superposition Example

If qubit =

Then

(which is the same state)

Pauli-Y Gate (Y Gate)

Adds both a bit flip and a phase rotation.

✓Action:

i = imaginary unit

Example

Input: $|0\rangle$

Input: $|1\rangle$

✓Meaning

It flips the qubit

But also adds a phase factor (i or −i)

This phase matters in interference and quantum algorithms.

✓Easy superposition example

If

Then output =

So the qubit becomes 1 with a phase shift.

Pauli-Z Gate (Z Gate)

Also called Phase Flip Gate.

✓Action:

It does not change amplitudes, only flips the phase of $|1\rangle$.

Example

Input $|0\rangle$:

Input $|1\rangle$:

✓ Meaning

Z gate keeps $|0\rangle$ same, but adds a minus sign to $|1\rangle$.

✓Superposition example

Let
Apply Z:
This is important in quantum phase estimation and interference.

2.Hadamard Gate (H-Gate) ?

The Hadamard gate is one of the most important quantum gates.
It is used to create superposition, which classical computers cannot do.
1. What does the Hadamard gate do?
The Hadamard gate transforms:
So:
If the qubit is $|0\rangle$, it becomes equal superposition of $|0\rangle$ and $|1\rangle$.
If the qubit is $|1\rangle$, it also becomes a superposition but with a minus sign on $|1\rangle$.
2. Hadamard Gate Matrix
The matrix representation is:
Hadamard gate is unitary, meaning it is reversible.
3. Example 1: Apply H on $|0\rangle$
Meaning:
The qubit now has:
50% probability of being measured as 0
50% probability of being measured as 1
This is a superposition state.
4. Example 2: Apply H on $|1\rangle$
This is also a superposition, but with a negative sign.
After measurement:
Still 50% chance 0
50% chance 1
But the phase (minus sign) affects interference in quantum algorithms.
5. Example 3: Apply H Twice (H•H)
A cool property:
So Hadamard is its own inverse.
Simple Explanation:
Applying H once → makes superposition
Applying H again → cancels and returns original state
Just like: H × H = Identity gate
6. Example 4: H on a Superposition
Let:
Apply H:
So the Hadamard gate can also remove superposition depending on the state.
This is used in:
Quantum algorithms
Quantum Fourier transform
Deutsch algorithm

7. Example 5: H on Two-Qubit System
Hadamard applied to first qubit only:
Input:
Apply H to first qubit:
So the final state is:
This creates a superposition of two 2-qubit states.
8. Why is Hadamard gate important?
It allows a quantum computer to explore multiple possibilities at once.
Almost every quantum algorithm begins with Hadamard gates.
Key uses:
Creates superposition
Starts quantum parallelism
Used in teleportation
Used in entanglement creation
Used in interference-based algorithms

3. Controlled-NOT (CNOT) Gate ?

The Controlled-NOT (CNOT) gate is one of the most important two-qubit quantum gates. It is used to create entanglement and is widely used in quantum algorithms and quantum teleportation.
A CNOT gate has two qubits:
Control qubit (C): Decides whether the operation is performed.
Target qubit (T): Gets flipped ($0 \leftrightarrow 1$) only when the control qubit is 1.
Working Principle
If the control qubit = 0, the target qubit remains unchanged.
If the control qubit = 1, the target qubit is flipped using the NOT (X) operation.
Truth Table
Control (C)
Target (T)
Matrix Representation
The CNOT gate is represented by the following 4×4 matrix:
Quantum Circuit Symbol
Control qubit : |C> ——●———
Target qubit : |T> ——X———
● = Control qubit
X = NOT operation on the target qubit
Examples
Example 1
Input: $|00\rangle$
Control = 0
Target = 0
Since the control qubit is 0, the target is not changed.

Output:
Example 2
Input: |01⟩
Control = 0
Target = 1
Control is 0, so no change.
Output:
Example 3
Input: |10⟩
Control = 1
Target = 0
Since the control qubit is 1, the target flips from 0 to 1.
Output:
Example 4
Input: |11⟩
Control = 1
Target = 1
Since the control qubit is 1, the target flips from 1 to 0.
Output:
Example: Creating Entanglement
Consider the following quantum circuit:
Qubit 1 : |0> ——H———●———
Qubit 2 : |0> ————————X———
Step 1: Apply the Hadamard (H) gate to the first qubit.
The two-qubit state becomes:
Step 2: Apply the CNOT gate.
|00⟩ remains |00⟩
|10⟩ changes to |11⟩
Final state:
This is called a Bell state, which is an entangled state.
Applications of CNOT Gate
Creates entanglement between qubits.
Used in quantum teleportation.
Used in quantum error correction.
Used in many quantum algorithms, such as Grover's and Shor's algorithms.

4.Unitary Operations & Reversibility ?

Quantum logic gates are not like classical logic gates.
Classical gates (AND, OR, NAND) lose information → they are irreversible
Quantum gates never lose information → they are reversible
This happens because every quantum gate is a unitary operator.

1. What is a Unitary Operation?
A matrix U is called unitary if:
Where:
$U^\dagger$ = conjugate transpose of U
I = identity matrix
This condition means:
The operation preserves information
The operation preserves length (probability = 1)
✓The inverse always exists → reversible
2. Why must quantum gates be unitary?
Because quantum states represent probability amplitudes.
If gates were not unitary:
Probability could increase/decrease (breaking physics)
Information could be lost
The operation couldn't be reversed
Thus, all quantum gates must be unitary → all are reversible.
3. Reversibility
If U is unitary:
So every quantum gate has:
A reverse (undo) operation
Sometimes the same gate is its own inverse
Example: Pauli-X, Z, Hadamard ($H \times H = I$)
4. Examples
Example 1: Pauli-X Gate (NOT Gate)
Matrix:
Check unitarity:
It is unitary
It is reversible
Its own inverse:
Meaning: applying X twice brings back the original state.
Example Action
Apply twice:
System returns to original → reversible.
Example 2: Hadamard Gate (H)
Matrix:
Check unitarity:
Its own inverse:
Example Action
Apply H again:
Original state restored → reversible.
Example 3: Pauli-Z Gate
Matrix:

Check:
So:
Unitary
Reversible
Self-inverse
Action:
Apply twice:
Original returned.
5. Example of a Non-Unitary (NOT Allowed) Gate
Classical AND gate:
Information is lost:
Multiple inputs (00, 01, 10) give the same output (0)
This violation means:
Cannot be reversed
Not unitary
Not allowed in quantum computing
6. Why Unitarity = Reversibility?
Because if:
Then:
So the inverse operation always exists

5.Quantum Circuit Representation ?

A quantum circuit is a graphical (diagram) way of showing:
Qubits (as horizontal lines)
Quantum gates applied to them
The order (sequence) of operations
Quantum circuits are used just like classical logic circuits, but for quantum operations.
1. Basic Elements of a Quantum Circuit
1. Qubit Wires
Each qubit is shown as a horizontal line.
Example:
|0> ——————————
|1> ——————————
Top line = qubit 1
Bottom line = qubit 2
2. Quantum Gates
Gates are shown as symbols placed on qubit wires.
Examples:
Gate
Symbol
Meaning

Pauli-X
X
NOT gate (flip)
Hadamard
H
Creates superposition
Z gate
Z
Phase flip
CNOT
●──X
Control + target
3. Time Flow
Quantum circuits read from left → right.
Left side = input state
Right side = output state
2. Simple Example: Applying a Hadamard Gate
Suppose you have one qubit starting in 0:
Circuit:
|0> ——— H ———
Meaning:
Apply H gate to the qubit
Mathematically:
3. Example: X (NOT) Gate on a Qubit
Circuit:
|1> ——— X ———
Since X flips the state:
4. Two-Qubit Circuit: CNOT Gate
This is the most common two-qubit circuit.
Circuit diagram:
Control qubit: |1> ——— ● ———
Target qubit: |0> ——— X ———
Explanation:
Top qubit = control
Bottom qubit = target
"●" means control
"X" means NOT (flip)
Since control = 1 → target flips
Input:
Output:
5. Multi-Gate Circuit Example
Let's apply two gates:

Apply Hadamard (H) on qubit 1
Then apply CNOT with qubit 1 as control
Circuit:
Qubit 1: |0> ── H ──── ● ────
Qubit 2: |0> ──────────── X ────
Step 1: Apply H on top qubit
State becomes:
Step 2: Apply CNOT
When control = 0 → no change (|00⟩)
When control = 1 → flip target (|10⟩ → |11⟩)
Final state:
This is an entangled Bell state.
6. Example: Circuit with 3 Gates
Circuit:
|0> ── H ──── X ──── Z ────
Meaning:
Apply Hadamard
Apply NOT
Apply Z phase flip
You read the circuit left → right.
7. Why Circuits Are Important
Quantum circuits help us:
✓Visually understand quantum algorithms
✓Show gate order clearly
✓Represent multi-qubit interactions
✓Design algorithms like:

7.Quantum Teleportation ?

Quantum teleportation is a technique to transfer an unknown quantum state from one qubit to another without physically moving the qubit itself.

Quantum teleportation uses three main ingredients:

1. Entanglement
2. Classical communication
3. Quantum Measurement

Participant's

Alice → sender
Bob → receiver
Qubit to teleport → unknown state
Alice → Sender (the person who sends information)
Bob → Receiver (the person who receives information)

***Exa*mple**

Suppose Alice has a qubit in an unknown state:

1. Create entanglement

Alice and Bob share an entangled pair:

2. Alice performs quantum operations

Alice applies:

CNOT gate

Hadamard gate

Then measures her two qubits.

3. Classical communication

Alice sends 2 classical bits (00, 01, 10, or 11) to Bob.

4. Bob applies correction

Based on Alice's bits, Bob applies:

No gate / X / Z / XZ

Applications

Quantum communication

Quantum networks

Distributed quantum computing

Quantum cryptography

8.Simulation of quantum circuits ?

Simulation of quantum circuits means using a classical computer to mimic how a quantum circuit works—how qubits change when quantum gates are applied and what results we get after measurement.

Why we simulate quantum circuits

Real quantum computers are limited and noisy

Helps in designing, testing, and debugging quantum algorithms

Useful for learning and research

How simulation works

1. Initialize qubits (usually in)
2. Apply quantum gates (H, X, CNOT, etc.) using matrix calculations
3. Track the quantum state
4. Measure qubits to get classical outputs

Example 1: Single-Qubit Simulation

Circuit:

Start with

Apply Hadamard (H) gate

Simulation result:

Measuring many times gives

50% → 0, 50% → 1

Example 2: Two-Qubit Simulation (Entanglement)
Circuit:
Start with
Apply Hadamard on qubit 1
Apply CNOT
Simulation result:
Measurements give only 00 or 11
Never 01 or 10 → shows entanglement
Popular simulation tools
Qiskit Aer
Cirq Simulator
QuTiP
Microsoft Quantum Simulator
Limitation
Simulation cost grows exponentially with qubits
(qubits → state values)

UNIT-3

QUANTUM PARALLELISM & INTERFERENCE IN QUANTUM COMPUTING

1. Quantum Parallelism

Quantum parallelism is the ability of a quantum computer to evaluate a function on many inputs simultaneously by using superposition.

A classical bit can be either 0 or 1, but a qubit can exist in a superposition of both states:

With n qubits, a quantum system can represent 2^n states at the same time. When a quantum gate or unitary operation is applied, it operates on all these states simultaneously.

Example:

Using Hadamard gates on 2 qubits:

$|00\rangle \xrightarrow{H\otimes H} \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$

Importance:

Quantum parallelism provides exponential computational power compared to classical systems.

2. Quantum Interference

Quantum interference refers to the phenomenon where probability amplitudes of quantum states combine, either constructively (amplifying correct answers) or destructively (canceling wrong answers).

Interference is controlled using quantum gates and is essential for extracting useful results from quantum parallelism.

Example:

In Grover's algorithm, interference is used to amplify the amplitude of the target state while suppressing others, increasing the probability of finding the correct solution.

3. Relationship Between Parallelism and Interference

Quantum parallelism allows simultaneous computation of many possibilities.

Quantum interference selectively enhances desired outcomes and removes unwanted ones.

Without interference, parallelism alone would not give a computational advantage.

4. Conclusion

Quantum parallelism and interference together form the foundation of quantum algorithms such as Deutsch–Jozsa, Grover's, and Shor's algorithms, enabling quantum computers to solve certain problems faster than classical computers.

2.Deutsch–Jozsa Algorithm in Quantum Computing ?

1. Introduction

The Deutsch–Jozsa algorithm is one of the first quantum algorithms that demonstrates a clear advantage of quantum computation over classical computation. It determines whether a given Boolean function is constant or balanced using only one function evaluation on a quantum computer.

2. Problem Definition

Given a function

f(x): \{0,1\}^n \rightarrow \{0,1\}

Constant function:

gives the same output (0 or 1) for all inputs.

Balanced function:

outputs 0 for half of the inputs and 1 for the other half.

3. Classical vs Quantum Approach

Classical computation:

Requires up to evaluations in the worst case.

Quantum computation:

Requires only one evaluation using quantum parallelism and interference.

4. Algorithm Steps

1. Initialize n input qubits to and 1 auxiliary qubit to .
2. Apply Hadamard gates to all qubits to create superposition.
3. Apply the oracle , which computes:

U_f |x\rangle|y\rangle = |x\rangle |y \oplus f(x)\rangle

5. Measure the input qubits.

5. Result Interpretation

If the measurement result is → the function is constant.

If the result is any other state → the function is balanced.

6. Example (n = 1)

For constant functions or , the output is .

For balanced functions or , the output is .

7. Role of Quantum Principles

Quantum Parallelism: Evaluates all inputs simultaneously.

Quantum Interference: Cancels wrong outcomes and amplifies the correct result.

8. Advantages

Exponential speedup over classical algorithms.

Demonstrates power of quantum interference.

9. Applications

Conceptual foundation for more advanced quantum algorithms.

Used for understanding oracle-based quantum computation.

3.Quantum parallelism and interference?

Quantum parallelism and interference are core principles in quantum computing that enable massive computational advantages through superposition and wave-like interactions of quantum states. Quantum parallelism allows processing exponentially many possibilities simultaneously, while interference amplifies correct solutions and suppresses incorrect ones.

Quantum Parallelism Basics

Quantum parallelism exploits qubit superposition, where n qubits represent 2^n states at once, unlike classical bits limited to one state. A quantum function evaluation creates a superposition of all inputs, computing outputs in parallel via unitary operations. This enables algorithms to explore vast search spaces efficiently, though measurement collapses the superposition to one outcome.

Superposition Foundation

Superposition underpins parallelism: Multiple qubits in superposition yield basis states with amplitudes, allowing parallel computation across all combinations. Classical computers process sequentially; quantum systems manipulate the full superposition in one step.

Quantum Interference Mechanics

Interference occurs when probability amplitudes overlap, leading to constructive reinforcement (higher probability) or destructive cancellation (lower probability). Gates like Hadamard create superpositions for interference patterns, distinguishing quantum from classical processing.This wave-like behavior extracts useful information from parallel computations.

Role in Algorithms

Shor's algorithm uses parallelism to evaluate periodic functions across superpositions, then applies Quantum Fourier Transform for interference revealing periods, factoring large numbers exponentially faster. Grover's search employs an oracle for parallel marking and diffusion for interference, achieving quadratic speedup over classical O(N) searches.

Key Differences from Classical

Classical parallelism divides tasks across processors; quantum parallelism represents all states inherently via superposition, with interference providing the speedup. Quantum requires coherence to avoid decoherence disrupting patterns,

unlike robust classical bits.No direct "exponential speedup for all problems"—benefits specific structured tasks

Quantum parallelism and interference form the backbone of quantum computing's power, enabling simultaneous evaluation of multiple states and selective amplification of solutions.

1:Introduction

Introduce quantum computing fundamentals, defining parallelism as superposition-based simultaneous computation and interference as amplitude reinforcement/cancellation. Outline paper structure and historical context like Deutsch's 1985 problem.

2: Classical vs. Quantum Basics

Contrast classical bits (0 or 1) with qubits in superposition . Explain how n classical bits yield n states sequentially, while n qubits yield superposed states.

3: Superposition Mechanics

Detail qubit superposition with Bloch sphere visualization. Cover tensor products for multi-qubit states, e.g., Bell states showing entanglement enabling parallelism.

4: Quantum Parallelism Defined

Define parallelism: unitary operators like compute f on all 2^n inputs in superposition simultaneously.

5: Parallelism in Action

Illustrate with Deutsch-Jozsa algorithm: black-box function evaluation on uniform superposition reveals determinism/balance in one query via parallelism.

6: Interference Fundamentals

Explain interference as wave amplitude addition: constructive , destructive cancels. Use double-slit analogy for qubits.

7: Interference Operators

Describe Hadamard (H) gates for superposition creation and phase shifts for interference. Show H on |0> yields equal amplitudes, enabling pattern formation.

8: Parallelism + Interference Synergy

Combine concepts: parallelism generates superpositions, interference extracts info (e.g., phase kickback in Grover's oracle marks solutions).

9: Key Algorithms

Cover Shor's (period-finding via QFT interference) and Grover's (diffusion operator for quadratic speedup). Include pseudocode and complexity: O(log N) vs. O(N).

10: Limitations and Future

Discuss decoherence disrupting interference, no universal speedup (BQP class), and NISQ-era applications. Conclude with scalability challenges and hybrid quantum-classical prospects.

2.Deutsch-jozsa algorithm ?

The Deutsch-Jozsa Algorithm is a landmark quantum algorithm that solves a specific mathematical problem exponentially faster than any classical computer. It serves as the primary "proof of concept" that quantum computers can use interference and superposition to outperform classical logic.

1. The Core Problem: Constant vs. Balanced

Suppose you have a hidden function f(x) that takes a binary input and returns either a 0 or a 1. You are promised that the function is either:

Constant: It gives the same output for every input (all 0s or all 1s).

Balanced: It gives 0 for exactly half of the inputs and 1 for the other half.

Example: For a 2-bit input (00, 01, 10, 11):

Constant: All four inputs result in 0.

Balanced: 00 and 01 result in 0, while 10 and 11 result in 1.

2. Efficiency Comparison

Classical: To be 100% sure a function is constant, you might have to check 2^{n-1} + 1 inputs (one more than half). For a 10-bit input, that's 513 checks.

Quantum: The Deutsch-Jozsa algorithm determines the answer in exactly one check (query), regardless of how many bits the input has.

3. Step-by-Step Explanation

The algorithm uses n qubits for the input and 1 "ancilla" (helper) qubit for the output.

Step 1: Initialization

We start with n qubits in state |0\rangle and the helper qubit in state |1\rangle.

Step 2: Creating Superposition

We apply a Hadamard (H) gate to every qubit. This puts the input qubits into a "superposition" of all possible inputs at once. The helper qubit enters the |-\rangle state.

Step 3: The Quantum Oracle (Function Evaluation)

We pass the qubits through a "black box" called the Oracle that represents f(x). Because of the |-rangle state of the helper qubit, a trick called Phase Kickback occurs: the output of the function is "kicked back" into the phase (the mathematical sign) of the input qubits.

***** If f(x) = 0, the state stays positive (+).

***** If f(x) = 1, the state becomes negative (-).

Step 4: Interference

We apply Hadamard gates to the input qubits again.

***** <u>If Constant:</u> The phases are all the same, causing constructive interference that pushes the qubits back to the original |00...0\rangle state.

***** <u>If Balanced:</u> The positive and negative phases cancel each other out (destructive interference), ensuring the final state is anything but all zeros.

Step 5: Measurement

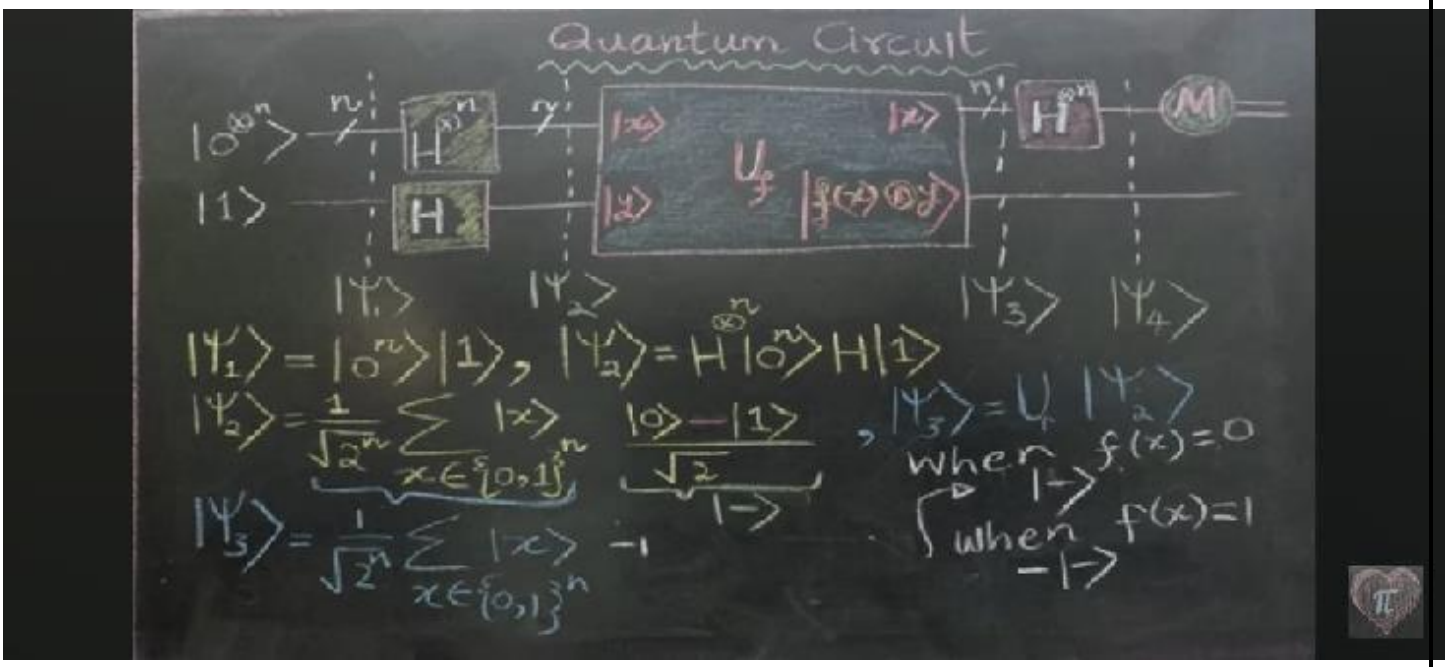
We measure the n input qubits.

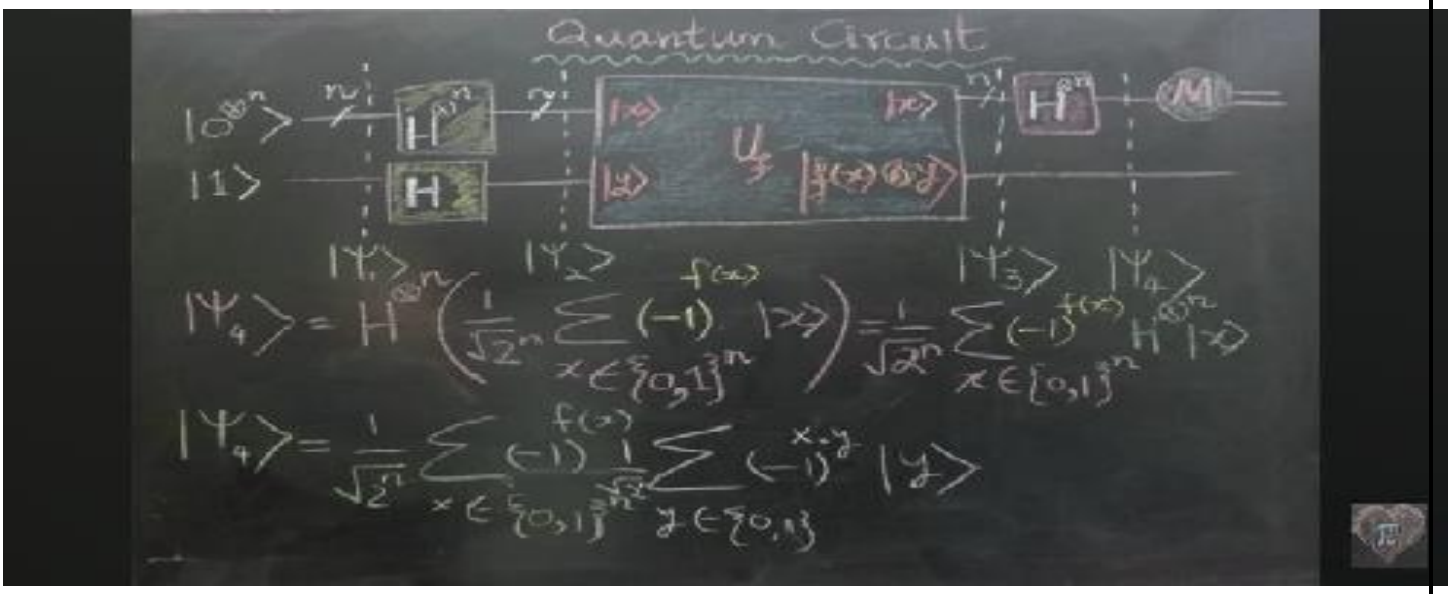
* If the result is all zeros (00...0), the function is Constant.

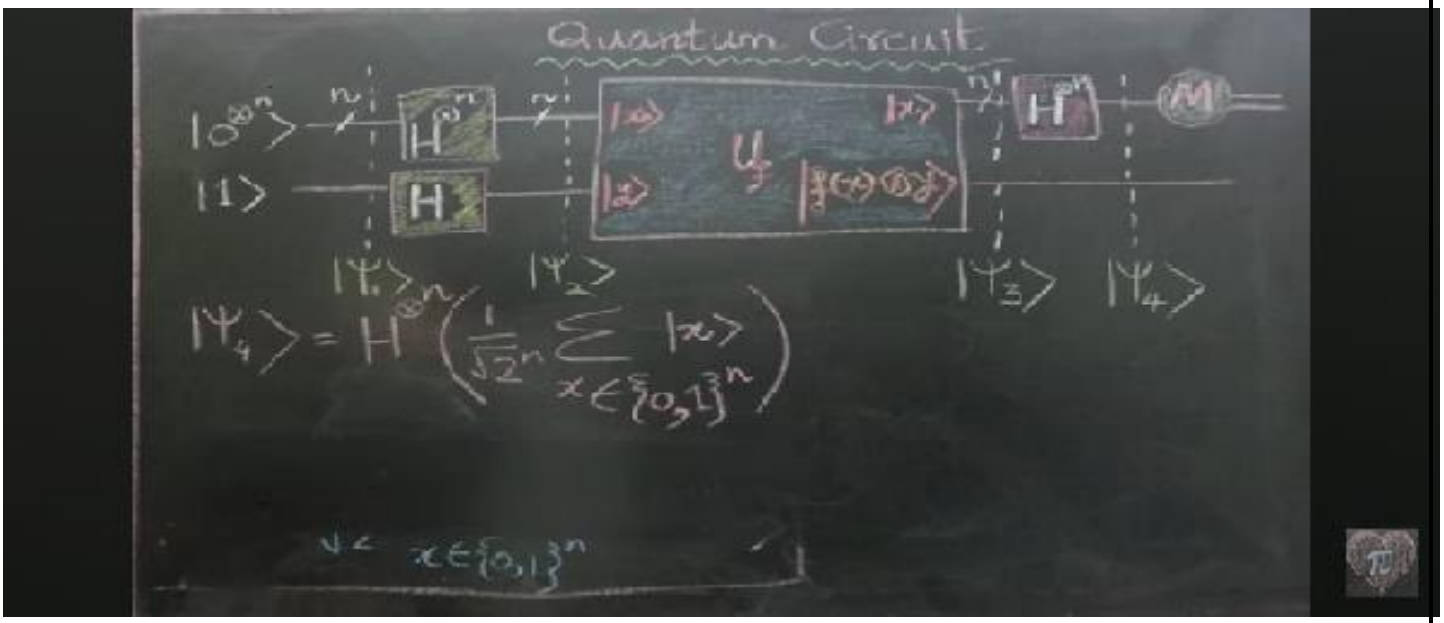
* If you see at least one 1, the function is Balanced.

<u>Why this matters</u>

The Deutsch-Jozsa algorithm proves that a quantum computer doesn't just "try all paths at once" (parallelism); it uses interference to cancel out wrong answers and amplify the global property of the function, solving in one step what would take a classical computer millions of years for large inputs.







Deutsch-Jozsa Algorithm, a fundamental quantum algorithm used to determine if a function is "constant" or "balanced."

Here is a breakdown of the letters and symbols used in the diagram:

1. Initial State Symbols (Left Side)

* |0\rangle^{\otimes n}: This represents n qubits initialized to the state 0. The \otimes n indicates a tensor product of n individual qubits.
* |1\rangle: A single "ancilla" (helper) qubit initialized to the state 1.
* H^{\otimes n} and H: These are Hadamard Gates. They put qubits into a state of superposition. H^{\otimes n} applies it to all n input qubits simultaneously.

2. Quantum States (\Psi)

The subscripts 1 through 4 represent the state of the system at different stages of the circuit:
* |\Psi_1\rangle = |0^n\rangle |1\rangle: The starting state before any operations.
* |\Psi_2\rangle: The state after the first set of Hadamard gates.
  * \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle: This represents the n qubits in an equal superposition of all possible 2^n binary strings.
  * |-\rangle: The state of the bottom qubit, which is \frac{|0\rangle - |1\rangle}{\sqrt{2}}.
* |\Psi_3\rangle: The state after the function oracle U_f has been applied.
* |\Psi_4\rangle: The final state after the second Hadamard gate, just before measurement.

3. The Oracle (U_f)

The large box in the center is the Quantum Oracle.
* U_f: A unitary transformation that implements the function f(x).
* |x\rangle: The input register (top wires).
* |y\rangle: The output/ancilla register (bottom wire).
* |f(x) \oplus y\rangle: The result of the oracle. It performs an XOR (addition modulo 2) between the function result f(x) and the current state of y.

4. Logic and Conditions

At the bottom right, the text explains the "Phase Kickback" mechanism:
* When f(x) = 0: The state remains |-\rangle (or |1\rangle in the specific context of the drawing's simplification).
* When f(x) = 1: The state acquires a negative sign, becoming -|-\rangle (labeled as -|1\rangle in the notes). This "kickback" allows the function's value to affect the phase of the top qubits.

5. Final Components (Right Side)

* M (in the circle): Represents the Measurement operation. The double lines coming out of it indicate classical bits.
* x \in \{0, 1\}^n: This notation means that x is a bitstring of length n consisting only of 0s and 1s.

3.Shor's Algorithm ?

1. Define Shor's Algorithm

Shor's Algorithm is a quantum algorithm used to factor large integers efficiently. It is exponentially faster than classical factoring algorithms.

2.Classical Reduction to Period Finding

Choose an integer a, where $0 < a < M$

Define the function:

$f(x) = a^x \mod M$

$a^r \equiv 1 \ (\text{mod } M)$

If r is even, then:

$(a^{r/2}-1)(a^{r/2}+1) \equiv 0 \ (\text{mod } M)$

Taking $\gcd(a^{r/2} \pm 1, M)$ gives non-trivial factors of M

3. Steps of Shor's Factoring Algorithm

Step 1:

Choose a random a

If $\gcd(a, M) \neq 1 \rightarrow$ factor found

Step 2:

Use quantum parallelism to compute:

$f(x) = a^x \mod M$

Step 3:

Apply Quantum Fourier Transform (QFT)

Measurement gives a value v close to

Step 4:

Use continued fraction expansion to estimate period r

Step 5:

If r is even:

$\gcd(a^{r/2}-1, M), \quad \gcd(a^{r/2}+1, M)$

Step 6:

Repeat if necessary

4. Quantum Core

Two quantum registers are used

Measuring the second register leaves the first register in a periodic superposition

QFT converts periodicity into sharp peaks (shown in figures)

5. Extracting the Period (Classical)

Measured value v

Condition:

$\left|\frac{v}{2^n} - \frac{j}{r}\right| < \frac{1}{2M^2}$

6. Example: Factoring M = 21

a = 11

Period r = 6
Since r is even:
gcd(11^3 - 1, 21) = 7
gcd(11^3 + 1, 21) = 3 □
Factors are 3 and 7

7. Efficiency

Classical factoring → exponential time
Shor's Algorithm → polynomial time:

4.Grover's algorithm?

Grover's algorithm is a quantum algorithm that solves the unstructured search problem. In an unstructured search problem, we are given a set of N elements and we want to find a single marked element. A classical computer would need to search through all N elements in order to find the marked element, which would take time O(N). Grover's algorithm, on the other hand, can find the marked element in time $O(\sqrt{N})$.

Grover's algorithm is a powerful tool that can be used to solve a variety of problems. For example, it can be used to find patterns in data, break cryptographic keys, and solve optimization problems. As quantum computers become more powerful, Grover's algorithm will become increasingly important.

Algorithm:

The algorithm works by applying a series of quantum operations to the input state, which is initialized as a superposition of all possible search states. The key idea behind Grover's algorithm is to amplify the amplitude of the marked state (i.e., the state containing the item that we are searching for) by iteratively applying a quantum operation known as the Grover operator.

The Grover operator has two quantum operations:

The reflection on the mean
The inversion of the marked state.
Here is a more detailed explanation of how Grover's algorithm works:

1. Initial state:

The algorithm starts in a state that is a superposition of all N elements. This state can be written as:
Click to enlarge
where $|x\rangle$ is the state corresponding to the element x.

2. Diffusion operator:

The diffusion operator is a quantum operation that amplifies the amplitudes of the states that correspond to the marked element. The diffusion operator can be written as
Click to enlarge

where I is the identity operator.

3. Measurement:

The algorithm measures the state of the system. This collapses the superposition and gives us the marked element. The repeated use of this operator increases the scope of the specified condition, making it easier to measure. Once the specified state is reached, the algorithm returns the index of the object corresponding to that state.

Proof of correctness:

The proof of the correctness of Grover's algorithm can be shown through the following steps:

Initialization: The algorithm begins with an input state, which is initialised as a superposition of all possible search states. The superposition state is given by: $|s\rangle = (1/\sqrt{N})\sum x|x\rangle$, where $|x\rangle$ represents the state of a particular item in the database, and N is the total number of items.

Marking the state: The algorithm then applies an oracle function, which marks the state containing the item that we are searching for. The oracle function flips the sign of the amplitude of the marked state:

$P(|s\rangle)=\sin 2((2k+1)\theta/2)$

$P(|s\rangle)=\sin 2((2k+1)\theta/2)$, where f(x) = 1 if x is the marked state, and f(x) = 0 otherwise.

Applying the Grover operator: The Grover operator is then applied iteratively to the state $|s'\rangle$. The Grover operator consists of two quantum operations: the reflection about the mean and the inversion about the marked state. The operator is applied O(sqrt(N)) times to amplify the amplitude of the marked state and decrease the amplitudes of the other states.

Measuring the state: Finally, the state is measured, and the algorithm returns the index of the marked state.

The correctness of the algorithm can be shown by analyzing the probability of measuring the marked state at the end of the algorithm. The probability of measuring the marked state after k iterations of the Grover operator is given by:

$P(|s\rangle)=\sin 2((2k+1)\theta/2)$

$P(|s\rangle)=\sin 2((2k+1)\theta/2)$, where $\theta = \arcsin(1/\sqrt{N})$ is the angle between the initial state and the marked state. As k increases, the probability of measuring the marked state approaches 1.

Therefore, Grover's algorithm can be shown to correctly find the marked state in O(sqrt(N)) time complexity with high probability.

Applications of Grover's Algorithm:

Data mining: Grover's algorithm can be used to find patterns in large datasets that would be impossible to find with a classical computer. For example, it could be

used to find fraudulent transactions in a financial database or to identify cancer cells in a medical image.

Cryptography: Grover's algorithm can be used to break cryptographic keys that are currently considered secure. This could have a major impact on the security of online communications and transactions.

Optimization: Grover's algorithm can be used to solve optimization problems that are difficult or impossible to solve with a classical computer. For example, it could be used to find the shortest route between two points or to find the optimal investment portfolio..

Limitations of Grover's Algorithm:

Quadratic speedup: Grover's algorithm gives a quadratic speedup compared to classical algorithms, which means that it can only provide a significant speedup for some problems that have a large search space. For problems with smaller search spaces, the speedup may not be good enough to justify the use of this algorithm.

Limited by hardware: Grover's algorithm requires the use of a large number of qubits to achieve a significant speedup, which is currently beyond the capabilities of most quantum hardware. This means that the algorithm may not be practical for many real-world applications until more powerful quantum hardware is developed.

Limited to unstructured search: Grover's algorithm is designed for unstructured search problems, which means that it may not be applicable to other types of problems such as optimization or simulation.

Error-prone: Grover's algorithm, like all quantum algorithms, is susceptible to errors due to noise and decoherence. These errors can affect the accuracy and reliability of the algorithm, which can limit its practical use in real-world applications.

Limited applicability in cryptography: Although Grover's algorithm can be used to break certain cryptographic algorithms, it is not applicable to all types of encryption. For example, public-key encryption is not vulnerable to Grover's algorithm due to its use of mathematical problems that are not efficiently solvable by quantum computers.

Grovers Algorithm

↳ used to find a required item from an unsorted database of size N in ~~O(n)~~ $O(\sqrt{N})$ Time.

problem statement

→ Given an unsorted database with $N = 2^n$ elements and a function

$$f(x) = \begin{cases} 1, & \text{if } x \text{ is the required element} \\ 0, & \text{otherwise.} \end{cases}$$

Classical v/s Quantum search

$O(N)$ $O(\sqrt{N})$ → Grover's Algorithm

main Components

1. Hadamard Gate → Creates Superposition
2. Oracle (Uf) → marks the solution by phase inversion.
3. Diffusion operator → amplifies probability of solution
4. Measurement → Gives final answer.

Algorithm steps

(1) initialize n qubits in state $|0\rangle$
(2) Apply Hadamard to all qubits → superposition
(3) Apply Oracle (Uf) → mark the soution by phase inversion.
(4) Apply Diffusion operator

(5) Repeat step (3) & (4) about $\sqrt{N}$ Times.

(6) measure the qubits → solution obtained with high probability.

Oracle operation

$$U_f |x\rangle = (-1)^{f(x)} |x\rangle$$

Diffusion operator

$$D = 2|s\rangle\langle s| - I$$

$$|s\rangle = \frac{1}{\sqrt{N}} \sum |x\rangle$$

Advantages → faster than classical search
↳ works for unsorted data

Limitations → Requires Quantum hardware
↳ Oracle must be known

Ex: Unsorted database of 4 elements
↳ 2 qubits.
(Assume Correct Answer is $|10\rangle$

(1) **Initialization**
start with both qubits $|00\rangle$, $|00\rangle$

(2) **apply Hadamad gate** → Creates equal superposition for all states.

$$|s\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

each state has equal probability = 1/4

(3) Oracle operation (U_f)

↳ flips the phase of the Correct state $|10\rangle$:

$\frac{1}{2}(|00\rangle + |01\rangle - |10\rangle + |11\rangle)$

only solution gets (−sign)

(4) Diffusion operator (Inversion about mean)

• mean Amplitude = $\frac{1+1-1+1}{4}$

• Amplitudes are reflected about the Mean

After Diffusion

$|10\rangle$ Gets Highest amplitude.

final state $|10\rangle$.

(5) measurement

↳ when measured, the result is $|10\rangle$.

1. Initialization of Qubits
All qubits are initialized to the **|0⟩ state**.
Explanation:
At the beginning, the quantum system is prepared in a known and stable state. Initializing qubits to |0⟩ ensures a clean starting point before applying quantum operations
2. Creation of Superposition (Hadamard Gates)
Hadamard gates are applied to all qubits to create an equal superposition of all possible states.
Explanation:
Applying the Hadamard gate allows the quantum system to represent **all N possible solutions simultaneously**. Each state has an equal probability amplitude, enabling parallel exploration of the search space.
3. Oracle Operation
The oracle is a quantum function that identifies the correct solution by flipping its phase.
Explanation:
The oracle does not reveal the answer directly. Instead, it **marks the correct state** by changing its phase from positive to negative, which distinguishes it from all other states.
4. Diffusion Operator (Amplitude Amplification)
The diffusion operator amplifies the probability of the marked state.
Explanation:
Also known as **inversion about the mean**, this step increases the amplitude of the target state while reducing the amplitudes of all other states. This makes the correct solution more likely to appear during measurement.
5. Iteration (Grover Iterations)
The oracle and diffusion operator are repeated approximately
π4N\frac{\pi}{4}\sqrt{N}4πN
times.
Explanation:
Each iteration gradually increases the probability of the correct state. Repeating the process an optimal number of times ensures maximum success probability without overshooting.
6. Measurement
The final quantum state is measured to obtain the result.
Explanation:
Measurement collapses the quantum superposition into a classical state. Due to amplitude amplification, the marked state has the highest probability of being observed as the output.

UNIT- 4
INTRODUCTION TO QISKIT & IBM QUANTUM EXPERIENCE

1.Introduction to Qiskit?

Qiskit is an open-source software development kit (SDK) developed by IBM for programming quantum computers.

It allows users to:

- Create quantum circuits
- Simulate quantum programs
- Run programs on real quantum hardware via the cloud
- Analyze quantum results

Main Components of Qiskit

- Terra – Used for creating quantum circuits and compiling them.
- Aer – Used for simulation of quantum circuits.
- IBM Quantum Provider – Used to access real IBM quantum computers.

Example (Creating a Bell State)

- Create 2 qubits
- Apply Hadamard (H) gate to first qubit
- Apply CNOT gate
- Measure both qubits

Result:

You get approximately:
50% $|00\rangle$
50% $|11\rangle$
This shows quantum entanglement.

1.Introduction to IBM Quantum Experience

IBM Quantum Experience is a cloud-based platform that allows users to:

- Access real quantum computers online
- Run quantum circuits through a web interface
- Use Jupyter notebooks for coding
- Visualize results
- It removes the need to own expensive quantum hardware.

Example

A student can:

- Log in to IBM Quantum Experience
- Create a simple circuit

Run it on:

- ✓ Simulator (ideal results)
- ✓ Real quantum device (noisy results)

This helps understand real-world quantum errors.

-

Advantages

Advantages of Qiskit

Open-source and free
Easy integration with Python
Large community and documentation
Access to real quantum hardware
Supports hybrid classical–quantum algorithms

Advantages of IBM Quantum Experience

Cloud-based access (no hardware required)
Real-time execution on quantum devices
Educational tools for students
Visualization tools (histograms, circuit diagrams)
Supports research experiments

Disadvantages

Disadvantages of Qiskit

Requires programming knowledge (Python)
Simulation of large circuits is computationally expensive
Results affected by hardware noise

Disadvantages of IBM Quantum Experience

Limited free access (queue waiting time)
Device availability depends on cloud scheduling
Real hardware produces noisy outputs

Limitations

Limitations of Qiskit

Dependent on current quantum hardware capabilities
Cannot eliminate hardware errors
Large-scale quantum algorithms not yet practical

Limitations of IBM Quantum Experience

Limited qubit count (NISQ devices)
No full quantum error correction
Cannot run large-scale fault-tolerant algorithms
Scalability challenges

2.Writing Quantum Circuits Qiskit ?

Introduction

Writing quantum circuits in Qiskit means designing a sequence of quantum gates applied to qubits, then measuring the output. Qiskit allows us to create, simulate, and execute these circuits on real quantum hardware.

A quantum circuit consists of:

1.Qubits
2.Quantum gates
3.Classical bits
4.Measurement operations

Steps to Write a Quantum Circuit in Qiskit

Step 1: Import Required Libraries
We import Qiskit modules.
Step 2: Create Quantum and Classical Registers
Define number of qubits and classical bits.
Step 3: Create Quantum Circuit
Initialize the circuit.
Step 4: Apply Quantum Gates
Examples:
H (Hadamard) – Creates superposition
X Gate – Bit flip
CNOT Gate – Creates entanglement
Step 5: Measure Qubits
Convert quantum information into classical bits.
Step 6: Execute the Circuit
Example 1: Simple Superposition Circuit
Create 1 qubit
Apply Hadamard gate
Measure
Initial state: $|0\rangle$
After H gate: $(|0\rangle + |1\rangle)/\sqrt{2}$
Measurement Result:
50% → 0
50% → 1
This demonstrates superposition.
Example 2: Bell State (Entanglement)
Steps:
A)Create 2 qubits
B)Apply H gate to first qubit
C)Apply CNOT (control=0, target=1)
D)Measure both
Output:

50% → 00
50% → 11

3.Measuring Qubits and Results ?

Introduction

Measurement is the process of extracting classical information from a qubit.

Before measurement, a qubit can exist in superposition, but after measurement, it collapses to either $|0\rangle$ or $|1\rangle$.

1) Measurement of a Single Qubit

A general qubit state is:

Where: α and β are probability amplitudes

$|\alpha|^2 + |\beta|^2 = 1$

When we measure:

Probability of getting 0 = $|\alpha|^2$

Probability of getting 1 = $|\beta|^2$

After measurement, the state collapses permanently to the measured value.

Example 1: Superposition Measurement

State:

Probabilities:

0 → 50%

1 → 50%

If measured:

Output is either 0 or 1 randomly

After measurement, the qubit becomes $|0\rangle$ or $|1\rangle$

This demonstrates wave function collapse.

2) Measurement of Two Qubits (Entangled State)

Consider Bell State:

Measurement results:

50% → 00

50% → 11

Important: If first qubit is measured as 0, second qubit automatically becomes 0.

If first qubit is 1, second becomes 1.

This shows entanglement correlation.

3) Measurement in Quantum Circuits

In quantum circuits:

Measurement converts quantum bits into classical bits

Results are collected over many runs (shots)

Output shown as histogram
Example: If run 1000 shots:
520 times → 00
480 times → 11
This approximates 50% probability.
4) Important Concepts
Probability amplitude – Complex number determining probability
Wave function collapse – State reduces after measurement
Shots – Number of circuit executions
Basis measurement – Usually computational basis (|0⟩ , |1⟩)

4.Classical -Quantum Hybrid Programs ?

1. Definition
Classical–Quantum Hybrid Programs combine classical computing and quantum computing.
The classical computer controls the program flow, while the quantum computer performs specific computations.
This approach is widely used because current quantum devices are limited (NISQ era).
2. Working Principle
The program is divided into two parts:
Classical part: Optimization, decision-making, parameter updates
Quantum part: Executes quantum circuits and returns measurement results
The classical computer sends inputs (parameters) to the quantum circuit.
The quantum processor runs the circuit and returns output.
The classical system analyzes results and updates inputs iteratively.
3. Key Features
Uses both classical and quantum resources efficiently
Suitable for current noisy quantum systems
Iterative in nature (loop-based execution)
Reduces complexity compared to fully quantum algorithms
4. Architecture (Steps)
1. Initialize parameters (classical computer)
2. Build quantum circuit with parameters
3. Execute circuit on quantum device/simulator
4. Measure output (probabilities)
5. Send results to classical computer
6. Update parameters using optimization
7. Repeat until optimal result is achieved
5. Advantages
Works well with limited qubits
Reduces effects of quantum noise
Practical for real-world applications

Easier to implement than fully quantum systems
6. Limitations
Requires continuous communication between classical and quantum systems
Slower due to repeated iterations
Still affected by noise and decoherence
Not suitable for all types of problems
7. Applications
Optimization problems
Machine learning
Chemistry simulations
Finance (portfolio optimization)

8.Example: Variational Quantum Eigensolver (VQE)
Used to find the minimum energy of a molecule
Hybrid approach:
Quantum computer prepares a parameterized state
Classical computer updates parameters to minimize energy
 Working:
Start with initial parameters
Run quantum circuit → measure energy
Classical optimizer updates parameters
Repeat until minimum energy is found
9. Another Example: QAOA (Quantum Approximate Optimization Algorithm)
Used for solving optimization problems
Uses parameterized quantum circuits + classical optimization loop
10. Conclusion
Classical–Quantum Hybrid Programs are essential in today's quantum computing era.
They bridge the gap between classical and quantum systems.
Widely used due to current hardware limitations.
Expected to play a major role in future quantum applications.

5.NISQ(Noisy Intermediate-Scale Quantum) ?

1. Definition
NISQ stands for Noisy Intermediate-Scale Quantum systems.
It refers to the current generation of quantum computers.

These systems have tens to a few hundred qubits but are not fully error-corrected.
Introduced by John Preskill (2018).
2. Key Characteristics
Noisy: Qubits are affected by errors due to environment (decoherence).
Intermediate-Scale: Limited number of qubits (not large-scale yet).
Short coherence time: Quantum states exist only for a short duration.
Probabilistic outputs: Results are not always exact.
3. Working Nature
NISQ devices execute quantum circuits but with imperfect accuracy.
Results are obtained by running circuits multiple times (shots).
Outputs are analyzed statistically.
4. Sources of Noise
Decoherence: Loss of quantum information due to environment
Gate errors: Imperfect quantum operations
Measurement errors: Inaccurate output readings
Thermal noise and hardware limitations
5. Advantages
First practical step toward quantum computing
Allows experimentation with real quantum hardware
Useful for research and algorithm development
Supports hybrid algorithms (classical + quantum)
6. Limitations
High error rates
Limited qubits
No full quantum error correction
Cannot solve very complex problems reliably
Results may be approximate, not exact
7. Applications
Optimization problems
Quantum machine learning
Chemistry and material simulations
Cryptography research
8. Examples of NISQ Devices
IBM Quantum processors
Google’s Sycamore processor
Rigetti quantum systems
9. Techniques to Handle Noise
Error mitigation (reducing errors without full correction)
Short-depth circuits
Hybrid algorithms like VQE and QAOA
Repeated measurements and averaging
10. Conclusion

NISQ systems represent the current stage of quantum computing.
Though limited and noisy, they are important for research and development.
They act as a bridge between classical systems and future fault-tolerant quantum computers.

6. LIMITATIONS OF QUANTUM STATES ?

1. Decoherence
Qubits lose their quantum state due to interaction with the environment.
This destroys superposition and entanglement.
Example:
A qubit may change state before computation finishes, giving wrong output.

2. Noise and Errors
Quantum operations are highly sensitive to noise.
Errors occur during gate operations and measurements.
Example:
Applying a gate may not produce the exact expected state due to hardware imperfections.
3. Limited Number of Qubits
Current quantum systems have only a small number of qubits.
Not sufficient for solving large-scale problems.
Example:
Factoring very large numbers is not yet practical with available qubits.
4. Error Correction Challenges
Quantum error correction is complex and requires many extra qubits.
Difficult to implement in real systems.
Example:
To protect 1 logical qubit, many physical qubits are needed.
5. Short Coherence Time
Qubits maintain their state only for a short time.
Computations must finish quickly before information is lost.
6. Measurement Limitations
Measuring a quantum state collapses it into a single outcome.
Cannot directly observe all possible states.
Example:

Even if a qubit is in superposition, measurement gives only 0 or 1.
7. Hardware Complexity
Quantum computers require extreme conditions like:
Very low temperatures
Isolation from environment
This makes them expensive and difficult to maintain.
8. Scalability Issues
Increasing the number of qubits while maintaining stability is difficult.
Large-scale quantum computers are still under development.
9. Algorithm Limitations
Not all problems benefit from quantum speedup.
Only specific problems (like factoring, search) show advantage.
10. High Cost and Accessibility
Quantum hardware is very expensive.
Limited access to real quantum machines.

7. Limitations of Current Quantum Systems in Quantum Hardware ?

Introduction

Although quantum computing is a powerful emerging technology, current quantum systems are still in the early stage (NISQ era) and have several limitations. These limitations prevent large-scale, fault-tolerant quantum computation.

1) Noise and Decoherence
Quantum systems are highly sensitive to environmental disturbances.
Decoherence: Qubits lose their quantum state quickly.
Interaction with environment destroys superposition and entanglement.
Example
If a qubit is in superposition, external noise may force it to collapse incorrectly before measurement.
2) High Error Rates
Quantum gates are not perfectly accurate.
Gate errors
Measurement errors
Crosstalk between qubits
Example

A CNOT gate may not produce perfect entanglement due to hardware imperfections.

3) Limited Qubit Count

Current devices have limited number of qubits (tens to a few hundred usable qubits).

Example

Running large-scale Shor's algorithm requires thousands to millions of error-corrected qubits, which are not available today.

4) Lack of Full Quantum Error Correction

Quantum error correction requires many physical qubits to create one logical qubit.

Huge hardware overhead

Not yet scalable

Example

To create 1 stable logical qubit, hundreds or thousands of physical qubits may be needed.

5) Short Coherence Time

Qubits remain stable only for a short duration.

Limits circuit depth

Complex algorithms fail before completion

Example

If coherence time is 100 microseconds, long circuits cannot be executed fully.

6) Scalability Challenges

Maintaining stability with increasing qubits is difficult

Cooling requirements (near absolute zero for superconducting qubits)

Complex hardware infrastructure

7) Connectivity Constraints

8) Limited Practical Applications

UNIT-5

1.QUANTUM MACHINE LEARNING & BASICS AND MODELS:

Quantum Machine Learning (QML) is the field that merges quantum physics with machine learning to process information in ways classical computers can't. While still in its early stages (the "NISQ" era—Noisy Intermediate-Scale Quantum), it promises to solve high-dimensional problems that are currently "intractable" or too complex for standard silicon chips.

1. The Core Basics

To understand QML, you have to look at how it differs from classical ML at the atomic level:

* **Qubits vs. Bits:** While a classical bit is either 0 or 1, a qubit can exist in Superposition (both states at once). This allows the model to "test" multiple possibilities simultaneously.
* **Entanglement:** Qubits can be linked so that the state of one instantly affects the other. This helps QML models find complex correlations in data that a classical computer might miss.
* **Hilbert Space:** Quantum systems operate in a mathematical "space" that grows exponentially with the number of qubits. This provides a massive "canvas" to map complex data.

2. Key QML Models

Most QML today follows a Hybrid Approach: a quantum computer does the heavy math (the "quantum circuit"), and a classical computer updates the parameters (the "optimizer").

A. Variational Quantum Circuits (VQC)

These are the quantum version of neural networks. You have a "circuit" with adjustable gates (parameters).

* **How it works:** You feed in data, rotate the qubits by certain angles (\theta), and measure the result. A classical optimizer then tweaks those angles to minimize error.
* **Ex:** Predicting whether a mole is cancerous based on image features encoded into qubit rotations.

B. Quantum Support Vector Machines (QSVM)

Classical SVMs find a "boundary" between data groups. When data is messy, they use "kernels" to project it into higher dimensions where it's easier to separate.

* **The Quantum Twist**: Quantum computers are naturally great at high-dimensional mapping. They use Quantum Kernels to find separations in data that are mathematically impossible for classical kernels to calculate.

Ex: Fraud detection in finance, where the "pattern" of a fraudulent transaction is hidden in hundreds of variables.

C. Quantum Neural Networks (QNN)

QNNs replace classical layers with quantum gates. They use "Amplitude Encoding" to squeeze massive datasets into a small number of qubits.

Ex: Generative models (like a Quantum GAN) that can create new molecular structures for drug discovery.

3. The QML Workflow (Step-by-Step)

If you were to build a QML model today, the process would look like this:

STEPS	ACTION	DESCRIPTION
1. Encoding	Classical ->Quantum	Turn your numbers/pixels into quantum states (e.g., Angle Encoding).
2. Processing	Quantum Circuit	Pass the qubits through gates (Hadamard, CNOT, etc.) to perform logic.
3. Measurement	Quantum -> Classical	"Collapse" the qubits to get a 0 or 1. This is your raw output.
4. Optimization	Training	Use a classical computer to adjust the circuit gates for the next run.

4. Popular Frameworks

If you want to start coding these, these are the three "Big Players":

* Qiskit (IBM): The most popular; great for beginners and integrates well with Python.
* PennyLane (Xanadu): Specifically built for QML and "differentiable quantum programming."
* Cirq (Google): Focused on writing and testing algorithms for Google’s Sycamore processors.

2.Quantum Cryptography & Quantum Key Distribution ?

While Quantum Machine Learning focuses on processing data, Quantum Cryptography focuses on securing it. Its "superpower" isn't just complex math—it is the laws of physics itself.

The crown jewel of this field is Quantum Key Distribution (QKD), which allows two parties to produce a shared, secret random key that can be used to encrypt and decrypt messages.

1. The Core Principle: No-Cloning & Observation

Standard encryption (like RSA) relies on the fact that it is mathematically hard to factor large prime numbers. Quantum cryptography relies on two physical rules:

* The No-Cloning Theorem: You cannot create an identical copy of an unknown quantum state. If a hacker tries to copy a "quantum key" while it's in transit, they will fail.
* The Observer Effect: In quantum mechanics, the act of measuring a system changes it. If an eavesdropper (usually called "Eve") tries to look at the key, she will leave "traces" (errors) that the sender and receiver will immediately detect.

2. Quantum Key Distribution (QKD)

QKD is the practical application of these principles. It doesn't send the message via quantum bits; it sends the encryption key.

The BB84 Protocol

Developed by Bennett and Brassard in 1984, this is the most famous QKD protocol.

* Preparation: Alice (the sender) sends a series of photons to Bob (the receiver). Each photon is polarized in a specific direction (e.g., Vertical, Horizontal, or Diagonal).
* The Quantum Channel: These photons travel through a fiber optic cable.
* Measurement: Bob chooses a "basis" (a filter) to measure each photon. Because of quantum uncertainty, if he chooses the wrong filter, his result is random.
* The Sifting Process: Alice and Bob talk over a standard, public phone line. They compare which filters they used (not the actual results). They keep the bits where their filters matched.
* Error Rate Check: They check a small sample for errors. If an eavesdropper was watching, the error rate will be high. If it's low, they have a secure, shared key.

3. Post-Quantum Cryptography (PQC) vs. QKD

It is important to distinguish between these two, as they are often confused:

Feature	Quantum Key Distribution (QKD)	Post-Quantum Cryptography (PQC)
Method	Uses hardware (lasers/photons).	Uses software (math algorithms).
Security Base	Laws of Physics.	Math problems hard for quantum computers.
Implementation	Requires new fiber-optic infrastructure.	Works on existing internet computers
Role	Generates a secure key.	Protects data against "Shor's Algorithm".

4. Real-World Applications

Quantum cryptography is no longer theoretical; it is currently being deployed in high-security sectors:

* Financial Networks: Banks use QKD to secure inter-branch data transfers.
* Government/Military: Protecting state secrets from "Harvest Now, Decrypt Later" attacks (where hackers steal encrypted data today to decrypt it once a powerful quantum computer exists).
* The Quantum Satellite: In 2017, China's Micius satellite achieved the first intercontinental QKD-secured video call between Beijing and Vienna.

3.Quantum Algorithms for AI & OPTIMIZATION ?

In the intersection of AI and Quantum Computing, we move from simply "processing data" to "solving logic." While classical AI struggles with the sheer number of combinations in large datasets, quantum algorithms use physics to find the "best" answer faster.

1. Key Quantum Algorithms for AI

These algorithms serve as the "engines" inside quantum-enhanced AI models.

A. The HHL Algorithm (Linear Systems)

Named after its creators (Harrow, Hassidim, and Lloyd), this algorithm solves systems of linear equations:

$Ax = b$.

* Why it matters for AI: Almost all Machine Learning—especially Neural Networks and Regressions—is built on linear algebra. HHL provides an exponential speedup in solving these equations for specific types of data.
* AI Example: Training a massive recommendation engine (like Netflix or Amazon) where the "matrix" of user preferences contains billions of data points.

B. Grover's Algorithm (Unstructured Search)

Grover's doesn't give an exponential speedup, but a quadratic one (\sqrt{N}).

* Why it matters for AI: In AI, we often have to search through a "state space" to find a specific solution (like finding the best move in a game). If a classical computer takes 1,000,000 steps, Grover's takes only 1,000.

* AI Example: Searching through unlabelled data for specific patterns or anomalies in cybersecurity.

2. Optimization in Quantum Computing

Optimization is the act of finding the "best" solution among millions of possibilities (the "Global Minimum"). Classical computers often get stuck in "Local Minima" (good solutions, but not the best).

A. Quantum Approximate Optimization Algorithm (QAOA)

This is a hybrid algorithm designed specifically for the current generation of "Noisy" (NISQ) quantum computers.

* How it works: It encodes an optimization problem into a "landscape" of energy. The quantum computer tries to find the lowest point of that landscape.

* Optimization Example: The Traveling Salesman Problem. If a delivery truck has 50 stops, there are more possible routes than atoms in the universe. QAOA explores these routes simultaneously to find the most fuel-efficient path.

B. Quantum Annealing

While QAOA uses "gates" (like a standard CPU), Quantum Annealing uses a physical process of "cooling."

* The Logic: It starts in a state of high-energy superposition (exploring everything) and slowly "cools" down. Quantum particles can tunnel through "energy barriers" that classical algorithms would have to climb over.

* Optimization Example: Portfolio Optimization. A bank needs to choose the best mix of 1,000 stocks to maximize profit while minimizing risk. An annealer can find the "sweet spot" in seconds.

3. Real-World Use Case: Drug Discovery

The most significant "Optimization + AI" application today is in Molecular Simulation.

* AI predicts which chemical structures might cure a disease.
* Quantum Algorithms (VQE/QAOA) optimize those structures by simulating how electrons interact at a subatomic level

4.Quantum Advantage & Supremacy ?

In quantum computing, the terms Quantum Supremacy and Quantum Advantage are often used interchangeably, but they represent two very different milestones on the road to a functional quantum future.

As of early 2026, the industry has largely moved past "Supremacy" (the scientific flex) and is now laser-focused on "Advantage" (the commercial reality).

1. Quantum Supremacy: The "Because We Can" Milestone

Quantum Supremacy is a purely technical threshold. It is reached when a quantum computer performs a calculation that would be practically impossible for any classical supercomputer to finish in a human timeframe.

* The Problem: The task doesn't have to be useful. In fact, most supremacy experiments involve "Random Circuit Sampling"—basically asking the computer to simulate its own noise.
* The Benchmark: Usually, this means taking a task that would take a classical supercomputer 10,000+ years and doing it in seconds.
* Key Achievement: In late 2024, Google's Willow chip performed a calculation in under five minutes that would have taken the world's most powerful supercomputer (Frontier) approximately 10 septillion years.

2. Quantum Advantage: The "Business Value" Milestone

Quantum Advantage occurs when a quantum computer solves a useful, real-world problem significantly better (faster, cheaper, or more accurately) than a classical computer.

* The Requirement: Unlike supremacy, this requires Error Correction. Because quantum bits are "noisy," you need a system that can fix its own mistakes to produce a reliable result for a business.
* The Focus (2025-2026): We are currently in the "Utility Era." Companies like IBM and Quantinuum are using 100+ qubit systems to find advantage in specific niches like Certified Randomness for cryptography and Material Science simulations.

3. Current State of the Race (2026 Update)

The conversation has shifted from "Can it beat a calculator?" to "Can it beat a laboratory?"

*Certified Randomness (2025): A major breakthrough occurred when researchers (including JPMorganChase) used a 56-qubit system to generate "truly" random numbers that are physically impossible to predict, a critical step for secure voting and fair gaming.
* Chemical Simulation: We are seeing the first signs of advantage in simulating small molecules for battery technology, where quantum systems can model electron behavior without the "approximations" classical computers are forced to use.
* The Hybrid Model: Most 2026 applications use a Quantum-Classical Hybrid. The quantum chip handles the "hard" math (like high-dimensional optimization), while the classical cloud manages the data input/output.

4. Why the Distrust

You might see "supremacy" claims being disputed. For example, when Google claimed supremacy in 2019, IBM countered that a better-programmed classical supercomputer could have done the same task in 2.5 days instead of 10,000 years. This "Classical Backlash" is a healthy part of science—it forces quantum engineers to aim for even higher benchmarks.

5.ETHICAL & SOCIETAL IMPACT OF QUANTUM TECHNOLOGIES?

As we enter 2026, quantum computing has moved from "theoretical physics" to "practical product." However, the ability to manipulate quantum circuits at scale introduces unique ethical and societal dilemmas that classical computing never faced.

These impacts are generally categorized into security, equity, and environmental sustainability.

1. The "Quantum Apocalypse" and Data Privacy

The most immediate ethical concern is the threat to global encryption. Quantum circuits running Shor's Algorithm can theoretically break RSA and ECC encryption—the "locks" on our bank accounts, medical records, and state secrets.

* "Harvest Now, Decrypt Later" (HNDL): Bad actors are currently stealing encrypted data and storing it, waiting for a powerful enough quantum computer to exist in a few years to unlock it.
* The Ethical Dilemma: Do tech companies have a moral obligation to transition to Post-Quantum Cryptography (PQC) immediately, even if the hardware isn't fully ready?
* Digital Trust: If quantum circuits can bypass privacy, the fundamental trust in digital identity and the "internet of value" could collapse without rapid governance.

2. The Quantum Divide (Equity and Access)

Unlike classical computers, which can be built in a garage, quantum computers require millions of dollars in cryogenic cooling and rare materials. This creates a risk of Technological Hegemony.

* Global Inequality: Developing nations may find themselves left behind as wealthy nations and "Big Tech" (Google, IBM, Microsoft) monopolize quantum power.
* Monopolization of Discovery: If a single company uses quantum circuits to discover a life-saving drug or a new battery material, do they have the right to

patent it exclusively, or should "quantum-discovered" breakthroughs be open-source for the common good?

3. Environmental and Resource Impact

Quantum circuits are "green" in terms of algorithmic efficiency, but their hardware footprint is significant.

* Energy Consumption: Maintaining qubits at temperatures colder than outer space (milli-Kelvin range) requires massive, constant power for dilution refrigerators.
* Ecological Resources: The production of quantum hardware relies on rare-earth metals and noble gases (like Helium-3), which are finite and often mined in ethically questionable conditions.
* The "Carbon-Aware" Shift: In 2026, researchers are pushing for Sustainable Quantum Computing frameworks that benchmark a quantum chip's carbon footprint against the "energy saved" by its faster calculations.

4. Economic and Labor Displacement

Quantum optimization (QAOA) and Machine Learning will likely disrupt specific industries faster than others.

* Job Displacement: High-level roles in financial modeling, logistics, and pharmaceutical research may see automation far beyond what classical AI achieved.
* Dual-Use Technology: A quantum circuit optimized for "finding a new chemical catalyst" for fertilizer can also be used to "find a new nerve agent" for chemical warfare. The ethical "dual-use" nature of these algorithms requires strict international oversight.

5. Governance and "Quantum Ethics" (2026 Update)

To address these issues, 2026 has seen the rise of Quantum Mission Policy Frameworks.

* Responsible Innovation: Governments are now requiring "Ethical Impact Assessments" before funding large-scale quantum projects.
* Standardization: International bodies are working to ensure that the benefits of quantum computing—like climate modeling and drug discovery—are shared equitably across the globe rather than being hoarded by a "Quantum Superpower."

6.FUTURE TRENDS & RESEARCH DIRECTIONS?

As we move through 2026, quantum computing has transitioned from a "physics experiment" to an "engineering race." The focus has shifted from building more qubits to building better, error-corrected qubits and integrating them into the existing global computing infrastructure.

1. The Rise of "Quantum-Centric Supercomputing"

The biggest trend in 2026 is the move away from standalone quantum computers. Instead, we are seeing the emergence of Hybrid Quantum-Classical Infrastructure.

* The Mosaic Architecture: Quantum processors are being treated like GPUs—specialized "accelerators" that handle specific, heavy-duty math problems while classical supercomputers manage the data flow and logic.
* Orchestration Layers: Software now automatically decides which part of a problem should go to a CPU, a GPU, or a QPU (Quantum Processing Unit).

2. Breakthroughs in Fault Tolerance (QEC)

In previous years, we lived in the "NISQ" era (Noisy computers). 2026 marks the early dawn of Fault-Tolerant Quantum Computing (FTQC).

* Logical Qubits: Research has successfully moved beyond individual "physical" qubits (which are error-prone) to "logical" qubits. By grouping hundreds of physical qubits together, systems like Google's Willow chip have demonstrated exponential error suppression.
* Below-Threshold Performance: Hardware is finally reaching the "threshold" where adding more qubits actually reduces the total error rate rather than increasing it.

3. Quantum Networking and the "Quantum Internet"

Research is expanding from computing at a location to communicating between locations using quantum states.

* Quantum Repeaters: A major research direction is the development of repeaters that can boost a quantum signal (entanglement) across fiber optic cables without "measuring" and destroying it.
* Distributed Quantum Computing: In 2026, experiments are scaling where two separate quantum computers are linked via a quantum network to work together as a single, larger virtual machine.

4. Novel Qubit Modalities

While superconducting qubits (IBM/Google) led the first wave, 2026 is seeing a surge in alternative "modalities" that might be easier to scale:

* Photonic Qubits (Light): Using photons allows for room-temperature quantum computing, potentially removing the need for massive, energy-hungry dilution refrigerators.
* Trapped Ions: Known for high "fidelity" (accuracy), these are being used for deep-circuit calculations that superconducting systems can't yet handle.
* Neutral Atoms: These use lasers to hold atoms in place, allowing for massive scalability (thousands of qubits) in a single processor.

5. The "Quantum-as-a-Service" (QaaS) Boom

You no longer need to own a quantum computer. In 2026, Cloud Quantum Computing is the standard. Platforms like AWS Braket, Microsoft Azure Quantum, and IBM Quantum allow developers to run experiments on multimillion-dollar hardware for a "pay-as-you-go" fee, democratizing access to this research.