Entanglement of Qubits in Hilbert Space

It integrates and expands on your uploaded notes, adding conceptual clarity, mathematical depth, examples (including Bell states), real-world applications, and Qiskit code demonstrations suitable for classroom or research-level explanation.

---

1. Introduction

Entanglement is one of the most profound and counterintuitive phenomena in quantum mechanics — a feature that distinguishes the quantum world from classical physics.

When two or more **qubits** are *entangled*, the state of one qubit cannot be described independently of the other, regardless of how far apart they are in space. This phenomenon creates a **non-local correlation** between their states.

In other words, measuring one qubit *instantly determines* the state of the other — even if they are separated by light-years. This property defies classical logic and underpins the extraordinary power of quantum computing, quantum communication, and quantum cryptography.

2. Entanglement in Hilbert Space

In quantum mechanics, every qubit exists in a **Hilbert space**, which is a mathematical vector space representing all possible quantum states.
For a single qubit, this space is **2-dimensional**:

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

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

When multiple qubits are considered, their combined system is represented by the **tensor product** of individual Hilbert spaces.

For **two qubits**, the Hilbert space becomes **4-dimensional**:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

A **separable (non-entangled)** state can be expressed as a product of individual states:

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$$

If the joint state **cannot** be written as such a product, it is said to be **entangled**.

3. Mathematical Definition of Entanglement

Let the state of two qubits be:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

If there are no coefficients $\alpha, \beta, \gamma, \delta$ that allow this to be factored as:

$$|\psi\rangle = (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle)$$

then the qubits are **entangled**.

A **simple test**:

The state is **entangled** if $\alpha\delta - \beta\gamma \neq 0$.

4. Bell States — Maximally Entangled States

The **Bell states** are the most well-known examples of maximally entangled two-qubit states. Named after **John Bell**, who studied their implications on quantum reality, these four states form an **orthonormal basis** for the 4D Hilbert space of two qubits.

They are defined as:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$
$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$$
$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$$
$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$$

Each Bell state represents a **perfect correlation** (or anti-correlation) between two qubits, even when separated by distance.

5. Proof That Bell States Are Entangled

Consider:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

If this state were separable, we could express it as:

$$|\Phi^+\rangle = (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle)$$

Expanding the right-hand side:

$$= ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle$$

Comparing with the left-hand side, we must have:

$$ac = \frac{1}{\sqrt{2}}, \quad ad = 0, \quad bc = 0, \quad bd = \frac{1}{\sqrt{2}}$$

From $ad = bc = 0$, either $a = 0$ or $d = 0$, but if $a = 0$, ac can't be non-zero; similarly for $d = 0$, bd can't be non-zero. Hence, it is **impossible** to express $|\Phi^+\rangle$ as a product — proving **entanglement**.

6. Visualization Using Qiskit

Let's create and visualize an entangled Bell pair.

 # Cell1

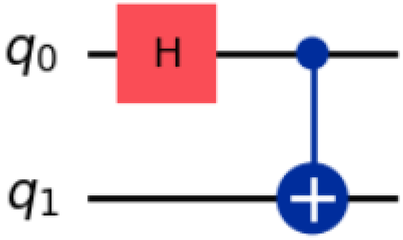
```
# Entanglement Visualization: Bell Pair and Single-Qubit Reduced States
This notebook demonstrates:
- Creation of a Bell pair |Φ⁺⟩ = (|00⟩ + |11⟩)/√2 using H and CNOT.
- Visualization of the joint state (state-city plot).
- Computation of reduced density matrices for each qubit.
- Manual calculation and plotting of Bloch vectors for reduced (single-qubit) states.
- Purity checks and a comparison with a separable state (|++⟩).

Run cells in order. Requires `qiskit` and `qiskit-aer`.
```

 # Cell 2: Imports and plotting setup

```
%matplotlib inline

# Close any existing matplotlib figures to avoid empty placeholders
import matplotlib.pyplot as plt
plt.close('all')

from IPython.display import display
import numpy as np

from qiskit import QuantumCircuit
from qiskit.quantum_info import Statevector, DensityMatrix, partial_trace, Pauli
from qiskit.visualization import plot_state_city, plot_bloch_vector
```

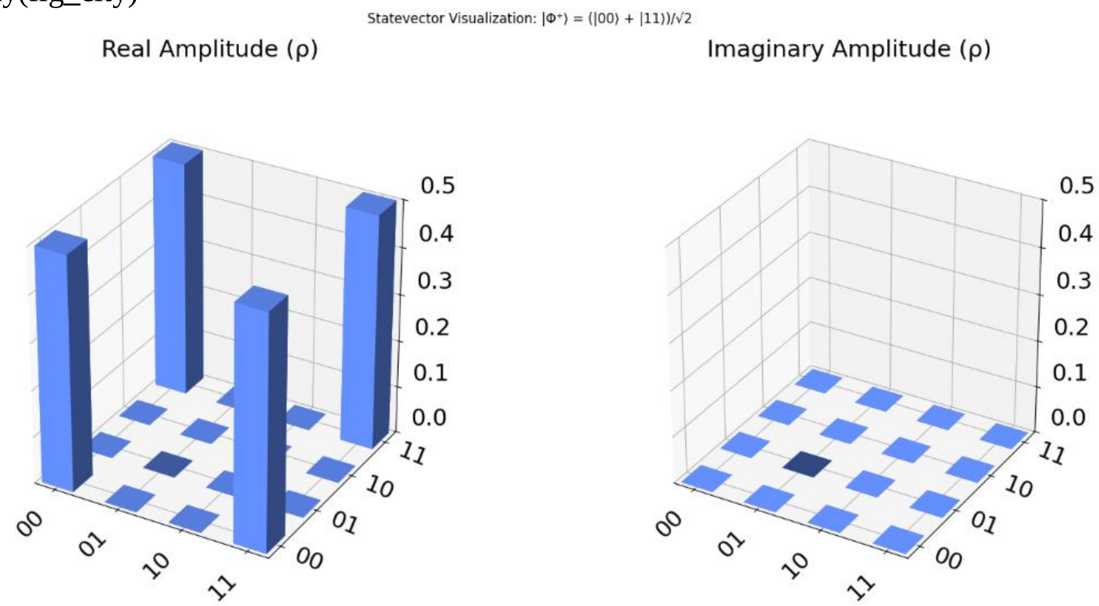
Explanation:
Load required libraries, configure inline plotting, and clear previous figures so new plots render cleanly.

```
# Cell 3: Create the Bell state circuit |Φ+> = (|00> + |11>)/√2
qc = QuantumCircuit(2)
qc.h(0)      # Hadamard on qubit 0 -> superposition
qc.cx(0, 1)  # CNOT with control 0, target 1 -> entangles

# Display the circuit diagram
display(qc.draw('mpl'))
```



Explanation:
This constructs the standard H then CNOT sequence that produces the Bell state.

```
# Cell 4: Compute the full two-qubit statevector and plot the joint state (city plot)
state = Statevector.from_instruction(qc)
# plot_state_city returns a matplotlib Figure; capture and display it explicitly
fig_city = plot_state_city(state)
fig_city.suptitle("Statevector Visualization: |Φ⁺⟩ = (|00⟩ + |11⟩)/√2", fontsize=12)
display(fig_city)
```



Explanation:
`plot_state_city` visualizes real/imag parts of amplitudes for |00>, |01>, |10>, |11>. Capturing and displaying the returned `Figure` avoids empty/placeholder axes

```
# Cell 5 (Corrected): Compute reduced density matrices (trace out the other qubit)
rho_total = DensityMatrix(state)        # total 2-qubit density matrix (pure)
rho_q0 = partial_trace(rho_total, [1])  # reduced state for qubit 0 (trace out qubit 1)
rho_q1 = partial_trace(rho_total, [0])  # reduced state for qubit 1 (trace out qubit 0)

# Display types and useful info
print("Type of rho_total:", type(rho_total))
print("Type of rho_q0:", type(rho_q0))
print("Type of rho_q1:", type(rho_q1))

# Display the underlying matrix shapes and content
print("\nMatrix shape details:")
print("rho_total matrix shape:", rho_total.data.shape)
print("rho_q0 matrix shape:", rho_q0.data.shape)
print("rho_q1 matrix shape:", rho_q1.data.shape)

# Optional: display actual matrices for insight
print("\nFull 2-qubit density matrix:\n", rho_total.data)
print("\nReduced 1-qubit rho_q0:\n", rho_q0.data)
print("\nReduced 1-qubit rho_q1:\n", rho_q1.data)
```

```
Type of rho_total: <class 'qiskit.quantum_info.states.densitymatrix.DensityMatrix'>
Type of rho_q0: <class 'qiskit.quantum_info.states.densitymatrix.DensityMatrix'>
Type of rho_q1: <class 'qiskit.quantum_info.states.densitymatrix.DensityMatrix'>

Matrix shape details:
rho_total matrix shape: (4, 4)
rho_q0 matrix shape: (2, 2)
rho_q1 matrix shape: (2, 2)

Full 2-qubit density matrix:
 [[0.5+0.j 0. +0.j 0. +0.j 0.5+0.j]
 [0. +0.j 0. +0.j 0. +0.j 0. +0.j]
 [0. +0.j 0. +0.j 0. +0.j 0. +0.j]
 [0.5+0.j 0. +0.j 0. +0.j 0.5+0.j]]

Reduced 1-qubit rho_q0:
 [[0.5+0.j 0. +0.j]
 [0. +0.j 0.5+0.j]]

Reduced 1-qubit rho_q1:
 [[0.5+0.j 0. +0.j]
 [0. +0.j 0.5+0.j]]
```

Explanation:
We form the full density matrix and then `partial_trace` to obtain each qubit's reduced density matrix. For maximally entangled Bell pair, these will be maximally mixed.

Interpretation

- Each **reduced density matrix** (ρ_0 and ρ_1) = ½ I_2, i.e., a **maximally mixed** qubit.
- The **full system** is pure (rank-1 density matrix with purity = 1).
- This confirms the **entanglement** — individually mixed, but jointly pure.

 # Cell 6: Helper to compute Bloch vector [x,y,z] from a single-qubit density matrix using Pauli expectations

```
def get_bloch_vector(rho):
    """
    rho: single-qubit DensityMatrix (or Operator-like) object.
    Returns [x, y, z] where x = Tr(rho X), y = Tr(rho Y), z = Tr(rho Z).
    """
    # Use Pauli objects to compute expectation values
    x = rho.expectation_value(Pauli('X'))
    y = rho.expectation_value(Pauli('Y'))
    z = rho.expectation_value(Pauli('Z'))
    return [float(x), float(y), float(z)]

# Compute Bloch vectors for the reduced states
bloch_q0 = get_bloch_vector(rho_q0)
bloch_q1 = get_bloch_vector(rho_q1)

print("Bloch vector for qubit 0:", bloch_q0)
print("Bloch vector for qubit 1:", bloch_q1)
```

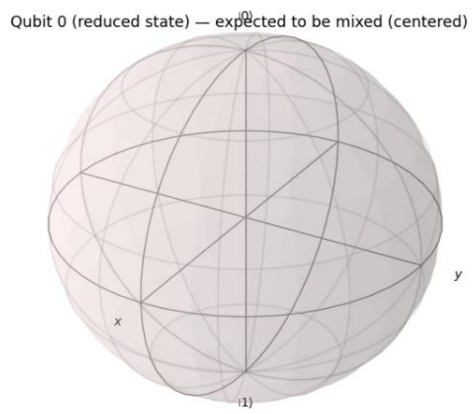
```
Bloch vector for qubit 0: [0.0, 0.0, 0.0]
Bloch vector for qubit 1: [0.0, 0.0, 0.0]
```

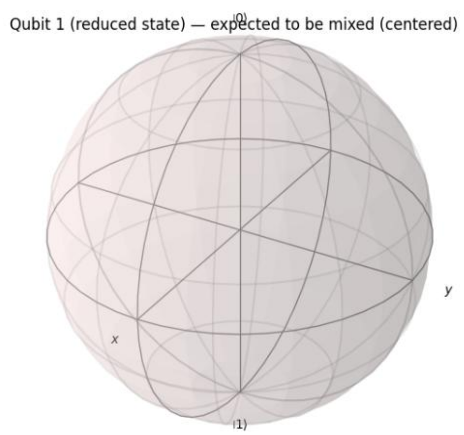
Explanation:

Newer Qiskit versions do not provide `.bloch_vector()` directly on DensityMatrix, so we compute components as expectation values of Pauli X, Y, Z.

 # Cell 7: Plot each single-qubit Bloch vector (figures returned; display them)

```
fig_b0 = plot_bloch_vector(bloch_q0)
fig_b0.suptitle("Qubit 0 (reduced state) — expected to be mixed (centered)", fontsize=12)
display(fig_b0)

fig_b1 = plot_bloch_vector(bloch_q1)
fig_b1.suptitle("Qubit 1 (reduced state) — expected to be mixed (centered)", fontsize=12)
display(fig_b1)
```





Explanation:
For a Bell pair the Bloch vectors will be ~[0,0,0], so the plotted spheres will be centered indicating mixed states (no definite direction) — this is the signature of entanglement.

 # Cell 8: Purity of whole system vs single-qubit reduced states

```
purity_total = rho_total.purity()
purity_q0 = rho_q0.purity()
purity_q1 = rho_q1.purity()

print(f"Purity of full 2-qubit system: {purity_total:.6f}")
print(f"Purity of qubit 0 (reduced): {purity_q0:.6f}")
print(f"Purity of qubit 1 (reduced): {purity_q1:.6f}")
```

```
Purity of full 2-qubit system: 1.000000+0.000000j
Purity of qubit 0 (reduced): 0.500000+0.000000j
Purity of qubit 1 (reduced): 0.500000+0.000000j
```

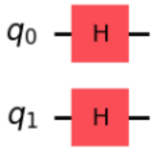
Explanation:

- Purity 1.0 for the combined system indicates a pure state.
- Purity 0.5 for each reduced single qubit indicates a maximally mixed single-qubit state — hallmark of maximal entanglement.

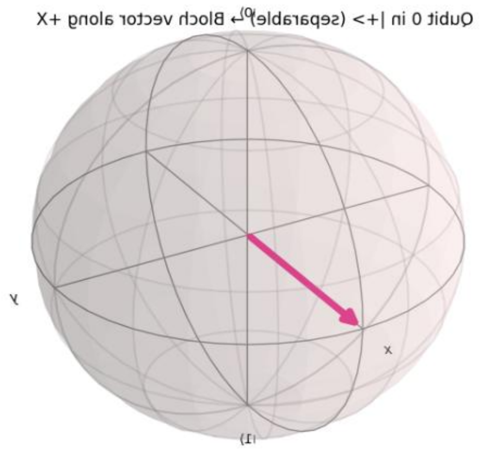
 # Cell 9: Create a separable product state |++> = H⊗H |00> for comparison

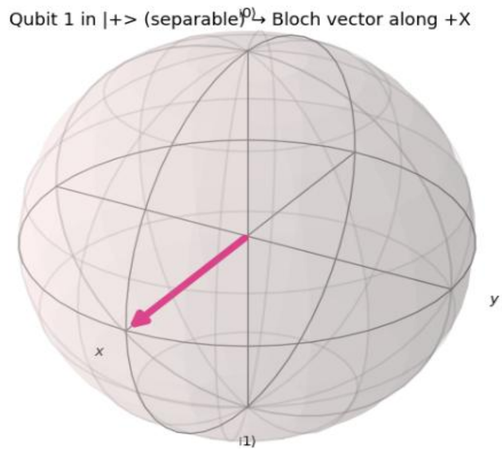
```
qc_sep = QuantumCircuit(2)
qc_sep.h(0)
qc_sep.h(1)

display(qc_sep.draw('mpl'))

state_sep = Statevector.from_instruction(qc_sep)
rho_sep = DensityMatrix(state_sep)
rho_q0_sep = partial_trace(rho_sep, [1])
rho_q1_sep = partial_trace(rho_sep, [0])

bloch_q0_sep = get_bloch_vector(rho_q0_sep)
bloch_q1_sep = get_bloch_vector(rho_q1_sep)

print("Separable |++> Bloch vectors (expected ~ +X):")
print("Qubit 0:", bloch_q0_sep)
print("Qubit 1:", bloch_q1_sep)

fig_sep0 = plot_bloch_vector(bloch_q0_sep)
fig_sep0.suptitle("Qubit 0 in |+> (separable) → Bloch vector along +X", fontsize=12)
```

```
display(fig_sep0)

fig_sep1 = plot_bloch_vector(bloch_q1_sep)
fig_sep1.suptitle("Qubit 1 in |+> (separable) → Bloch vector along +X", fontsize=12)
display(fig_sep1)
```




```
Separable |++> Bloch vectors (expected ~ +X):
Qubit 0: [0.9999999999999996, 0.0, 0.0]
Qubit 1: [0.9999999999999996, 0.0, 0.0]
```





Explanation:
This contrasts entangled vs separable: for $|++\rangle$ the single-qubit Bloch vectors are nonzero (pointing +X), unlike the Bell case where they are centered.

Note :

- The Hadamard gate creates superposition on qubit 0:
  $|0\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
- The CNOT gate correlates qubit 1 with qubit 0, forming:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

- Measurement of one qubit determines the outcome of the other — this is entanglement in action.

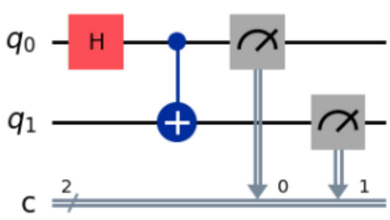
7. Measurement Correlation Example

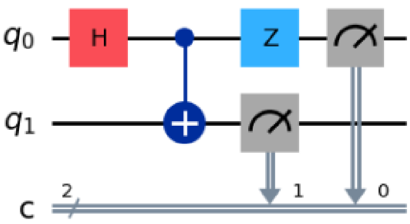
If both qubits are measured in the computational basis:

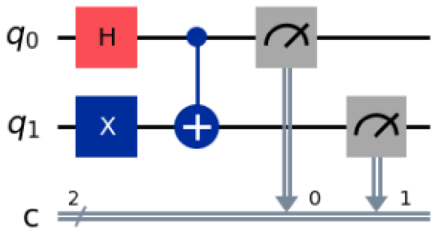
- 50% of the time → both yield 0
- 50% of the time → both yield 1
  They are **perfectly correlated**.

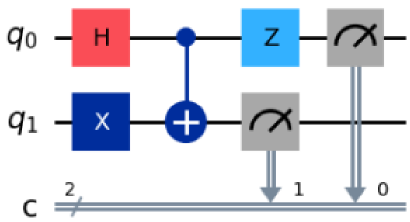
8. Quantum Circuit for All Four Bell States

```
from qiskit import QuantumCircuit
from qiskit.visualization import plot_histogram
from qiskit_aer import AerSimulator

sim = AerSimulator()

# Define a function to create Bell states
def create_bell(index):
    qc = QuantumCircuit(2, 2)
    qc.h(0)
    if index in [2, 3]:  # for Ψ states, add an X on qubit 1
        qc.x(1)
    qc.cx(0, 1)
    if index in [1, 3]:  # for negative phases
        qc.z(0)
    qc.measure([0,1], [0,1])
    return qc

# Execute and display measurement distributions
bell_labels = ["Φ+", "Φ−", "Ψ+", "Ψ−"]

for i in range(4):
    qc = create_bell(i)
    result = sim.run(qc, shots=1000).result()
    counts = result.get_counts()
    display(qc.draw('mpl'))
    plot_histogram(counts, title=f"Bell State |{bell_labels[i]}⟩ Measurements")
```









9. Real-Time Applications of Entanglement

a) Quantum Communication

- **Quantum Teleportation**: Transfer of quantum states between distant qubits using shared entanglement.
- **Superdense Coding**: Sending two classical bits of information using only one entangled qubit.

b) Quantum Cryptography

- **Quantum Key Distribution (QKD)**, e.g., **BB84 protocol**, uses entanglement to guarantee secure communication — any eavesdropper disturbs the entanglement and can be detected.

c) Quantum Computing

- Entanglement is the resource enabling **quantum parallelism**.
- Algorithms like **Shor's** and **Grover's** exploit entanglement to correlate computational pathways.

d) Quantum Sensors & Metrology

- Entangled qubits improve measurement precision (quantum-enhanced sensing).
- Used in atomic clocks, gravitational wave detectors, and magnetometers.

e) Quantum Internet

- Entanglement swapping and **quantum repeaters** form the foundation of **quantum networks** connecting remote quantum processors.

---

10. Key Insights

Concept	Classical View	Quantum (Entangled) View
State of particles	Independent	Correlated, non-separable
Measurement	Reveals pre-existing property	Collapses joint state
Information	Local	Non-local (shared instantaneously)
Mathematical form	Product of states	Superposition in joint Hilbert space

11. Summary

Entanglement reveals the **non-local interconnectedness** of quantum systems. Two qubits, once entangled, behave as a single entity described by a shared quantum state, even across vast distances.

It is the cornerstone of **quantum computing**, **quantum communication**, and **quantum security** — enabling tasks impossible for classical systems. Understanding and harnessing entanglement is what truly unlocks the *quantum advantage*.