PES UNIVERSITY

Karnataka, Bangalore-560080

Course: Quantum Transport and Logic Gates

Course code: UE22EC342BB1

Project on:

"Modelling Interaction Energy and Gate Operations in Quantum Dot Qubits"

Project by: Purab P Bhat | PES1UG22EC220

Nithin M | PES1UG22EC184

Under guidance of :

Dr Kaustav Bhowmick

ECE Dept

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINNERING

I. Abstract:

This study is to simulate and observe the role of interaction energy in the operation of √SWAP gates implemented using coupled quantum dots (QDs). Using two foundational studies spin-based model for quantum computation [1] and density-functional theory (DFT) simulation of large QD [2] we perform an analysis combining quantum decoherence modelling and computational energy estimation. The SWAP gate, governed by Heisenberg exchange interactions, works by precise tuning of inter-dot coupling to operate entanglement between spin-½ electrons. Using the DFT framework, we examine how electron-electron interaction energy evolves under external potentials, and how classical systems and potential symmetry impact spin configurations. In parallel, we apply spin master equation formalism to model gate dynamics under environmental decoherence. Together, these perspectives enable a detailed understanding of how interaction energies shape gate fidelity, coherence times, and practical implementabilty of two-qubit operations in semiconductor quantum-dot systems. The study highlights critical dependencies of SWAP gate performance on exchange strength J(t), pulse duration t_s, and the effects of external potential landscapes, which helps in the design of robust quantum-dot-based quantum logic gates.

II) Introduction:

2.1 Quantum Dots

Quantum dots (QDs) are nanoscale semiconductor structures that confine charge carriers in all three spatial dimensions, leading to discrete, quantized energy levels. These properties enable precise control over electronic and optical behaviours, making Quantum Dots a foundational platform in quantum computing, particularly for spin qubits. Recent technological advances have enabled the development of high-fidelity quantum gates (e.g., CNOT, Toffoli) using electron-spin qubits embedded in QDs where the electron's spin-½ state encodes quantum information. These gates rely on exchange interactions, controlled through tunneling barriers, as highlighted in [1].

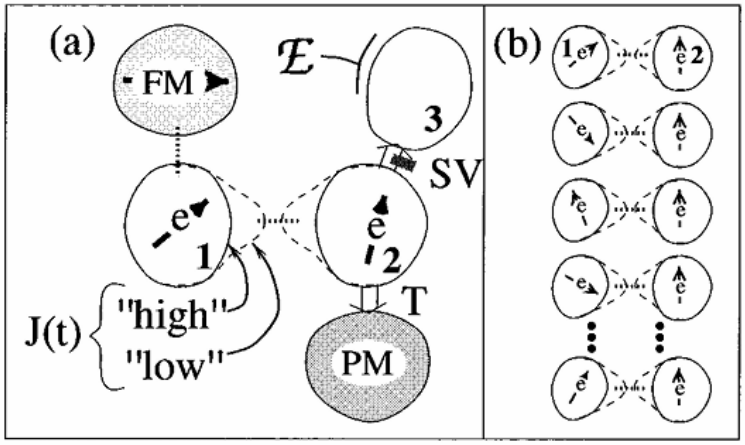


Fig 1: (a)Schematic top view of two coupled quantum dots

(B) Proposed experimental setup for initial test of swap gate operation in an array of many noninteracting quantum-dot pairs. The left column of dots is initially unpolarized, while the right one is polarized; this state can be reversed by a swap operation

QDs can be created from materials like InAs, GaAs, or silicon, each offering trade-offs in terms of coherence time and optical response. Concurrently, engineering methods such as

wavefunction control and surface termination have enhanced QD functionality, with significant efforts focused on minimizing decoherence and integrating QDs with CMOS technologies for scalable architectures. Interaction energy in QD-based SWAP gates can be studied with

- Spin Hamiltonian Approach: Time-dependent control of qubit entanglement via J(t) [1].
- Density Functional Theory: Interaction energy modelling at the electronic structure level [2].

The performance and fidelity of quantum gates, including SWAP gates, depend on:

- Interaction energy (exchange coupling J(t)).
- Suppression of decoherence,
- Compatibility with scalable semiconductor technology (e.g., CMOS).

2.1 Quantum Gates

Universal quantum computation requires the realization of a complete set of quantum gates[3], including:

- Single-qubit gates (rotations around the X, Y, Z axes),
- Two-qubit gates like the controlled-NOT (CNOT)
    - A photon interacts with a pair of spin qubits confined in coupled QDs via the cavity.
    - Photon polarization measurements, followed by conditional single-qubit operations, realize a deterministic CNOT gate.
- Three-qubit gates like the Toffoli gate
    - The three-qubit Toffoli gate is realized without the need for ancillary qubits
    - It uses three stationary electron-spin qubits in QDs, and the gate logic is encoded in a sequence of exchange interactions and spin-selective controls.

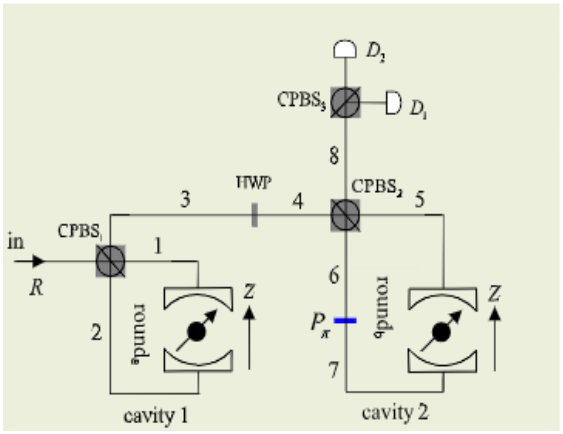


Fig 2 (a)

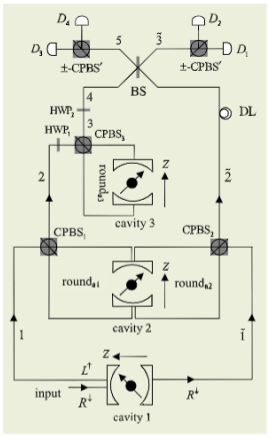


Fig 2 (b)

Fig 2: (a) Schematic diagram for compactly implementing a CNOT gate on two electron-spin qubits in optical microcavities. (b) Schematic diagram for compactly implementing a deterministic Toffoli gate on three electron-spin qubits, assisted by QD-cavity systems [3].

Implementation and Analysis

3.1 QDOT RY Gate

In quantum dots (QDots), qubits are usually encoded in the spin states of electrons, which is |0⟩ = (spin down (↓)) and |1⟩ (spin up (↑)). In quantum computing, an RY gate is a single-qubit rotation in the Y direction of the Bloch sphere [4] by an angle θ (fig 4). It's written as given in fig 3.For example, applying RY(π) to |0⟩ gives |1⟩ (NOT gate).

This is achieved using a magnetic field or microwave pulse tuned to interact with the spin. In a quantum dot system, this involves: Pulsing a local magnetic field (or using electron spin resonance, ESR),Or coupling to microwave photons in a cavity to induce the spin rotation. This is often done using control gates that apply electromagnetic fields to perform controlled single-qubit rotations like RY(θ).

$$RY(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$$

Fig 3: Representation of single qubit Rotation Y gate.

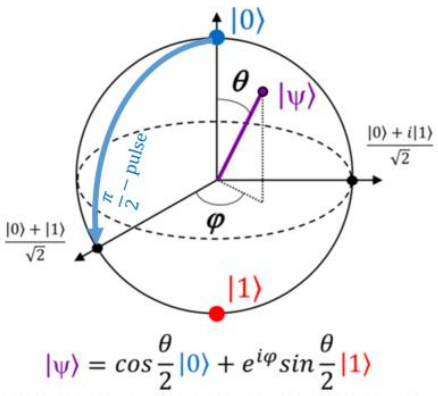


Fig 4: The Bloch sphere provides a useful means of visualizing the state of a single qubit and operations on it. Any point on this sphere represents a linear combination of the 0 and 1 states with complex coefficients. A π/2-pulse 'rotates' a qubit from the 0-state to a superposition state [4].

3.2 Bloch sphere represenation

The Bloch sphere is a geometrical representation of a qubit state:

- |0⟩ is at the north pole (top),
- |1⟩ is at the south pole (bottom),
- Any quantum state |ψ⟩ is a point on the surface of the sphere:

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

RY(θ) Rotation

- The RY(θ) gate rotates the qubit around the Y-axis by angle θ.
- On the Bloch sphere:
  - Start at |0⟩ (top),
  - Apply RY(θ),
  - You move along a vertical arc in the XZ plane toward |1⟩.

RY(π): takes |0⟩→|1⟩ (180° rotation)

$$RY(\pi/2)\text{: takes } |0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

3.3 Implementation of RY gate in quantum dot spin qubit

- quantum dot traps a single electron.
- The spin of the electron encodes the qubit. Visualised in fig 5.

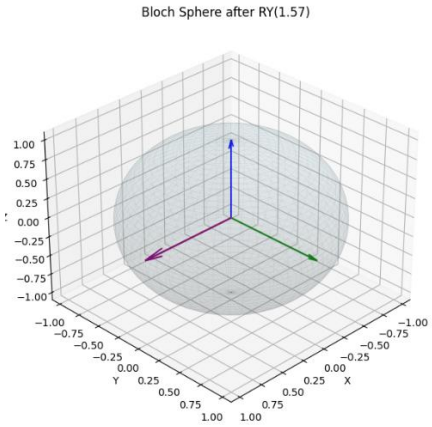


Fig (5): Bloch sphere represeantaion after RY(θ).

3.4 RY(π/2) Rotation of a QDot Spin Qubit: Bloch Sphere Interpretation

Initial State: |0⟩

- The qubit starts in the |0⟩ state, represented as a point at the north pole of the Bloch sphere:
  $\vec{v}$initial=(x=0,y=0,z=1)
  Apply RY(π/2) Gate
- This rotates the qubit 90° around the Y-axis of the sphere.
- After this operation, the qubit moves to the X-direction on the equator:
  $\vec{v}$after=(x=1,y=0,z=0)
  Graphically, this means the qubit has moved from pointing “up” (north pole) to pointing “sideways” in the X direction.

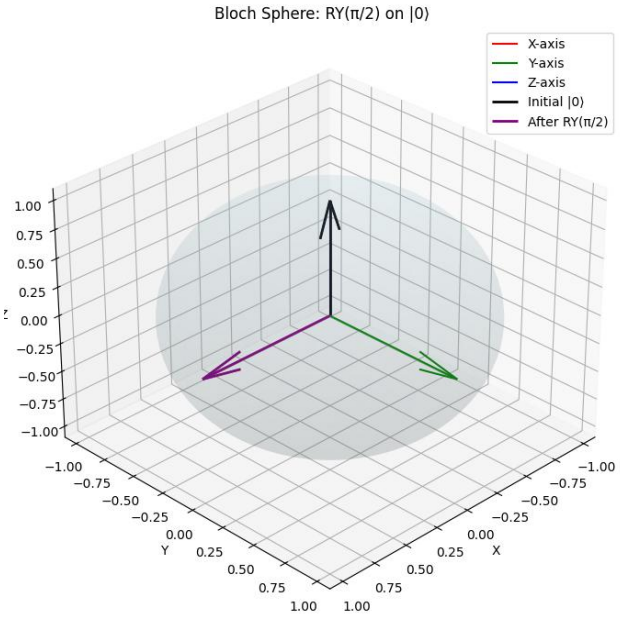


Fig 6: Bloch sphere representation after applying RY(π/2) on |0⟩

- A black arrow pointing up (initial state |0⟩: (0, 0, 1))
- A purple arrow pointing along X (after RY(π/2): (1, 0, 0))
- This visually confirms the 90° rotation around the Y-axis.
- Instead of being fully in |0⟩, the qubit is now in an equal superposition:

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

3.5 Interaction energy in QDOT RY gate

In a QD, a spin qubit is encoded using the spin of a single confined electron. To perform quantum operations like the RY(θ) gate, external fields are applied to coherently manipulate the spin state. However, the behavior of this spin including how precisely it can be rotated depends heavily on the underlying interaction energies within the dot where Density Functional Theory (DFT) becomes critical [2]. It provides a first-principles way to model and compute [2]:

- The electron-electron interaction energy, and
- The response of the system (including spin states) under external perturbations such as gate voltages or magnetic fields.

The interaction energy determines:

- How strongly an external field couples to the electron's spin.
- The energy separation between spin states (Zeeman splitting under a magnetic field).
- The influence of spin–orbit interaction or hyperfine coupling, which can either stabilize or destabilize coherent rotations.

Key DFT Observations Relevant to RY Gates:

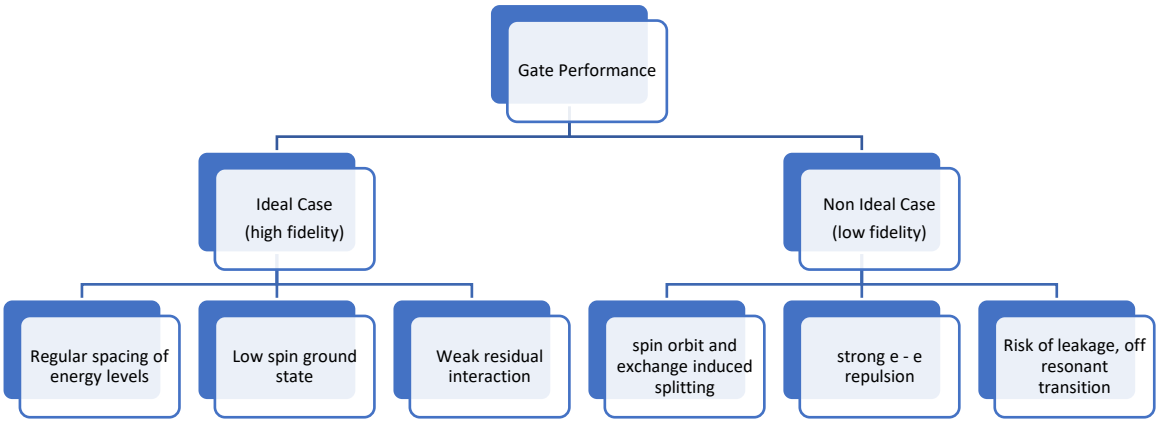
1. Spin Configuration Sensitivity:
    - DFT simulations showed that the ground-state spin configuration depends strongly on the symmetry of the external potential and interaction energy.
    - For a smooth harmonic potential, low-spin states are favored (which is ideal for qubit operation).
    - For asymmetric or irregular potentials, high-spin configurations emerge due to exchange interactions.
2. Exchange-Correlation Effects:
    - The exchange energy significantly modifies the spin states, which influences how reliably a spin can undergo RY rotation without leakage to other spin-orbital states.
3. Classical Integrability and Chaos:
    - DFT shows that in classically chaotic dot potentials, orbital level spacings become irregular.
    - This irregularity can lead to variability in RY gate fidelity, because spin transitions may become entangled with orbital excitations — especially in non-parabolic confinement.
4. Quantitative Control:
    - DFT provides quantitative values for single-particle energy levels, Coulomb energy, and exchange splitting, which are essential inputs to tune the microwave frequency and pulse strength used in RY(θ) operations.

Interaction energy quantifies how much the total energy of the system is altered due to these internal forces.

In DFT and spin models, we typically compute:

- E_H: Hartree energy (electron–electron repulsion),
- E_{xc} : Exchange-correlation energy (quantum mechanical),
- E_J: Spin–spin interaction energy, in spin Hamiltonians.

3.6 RY Gate performance analysis



RESULTS:

The interaction energy, as captured by DFT, is a fundamental parameter in determining the fidelity, coherence, and controllability of an RY gate on a quantum dot spin qubit. DFT simulations are essential for:

- Designing optimal confinement potentials,
- Predicting how exchange and Coulomb interactions affect gate operations,
- Calibrating microwave pulses based on precise spin splitting.

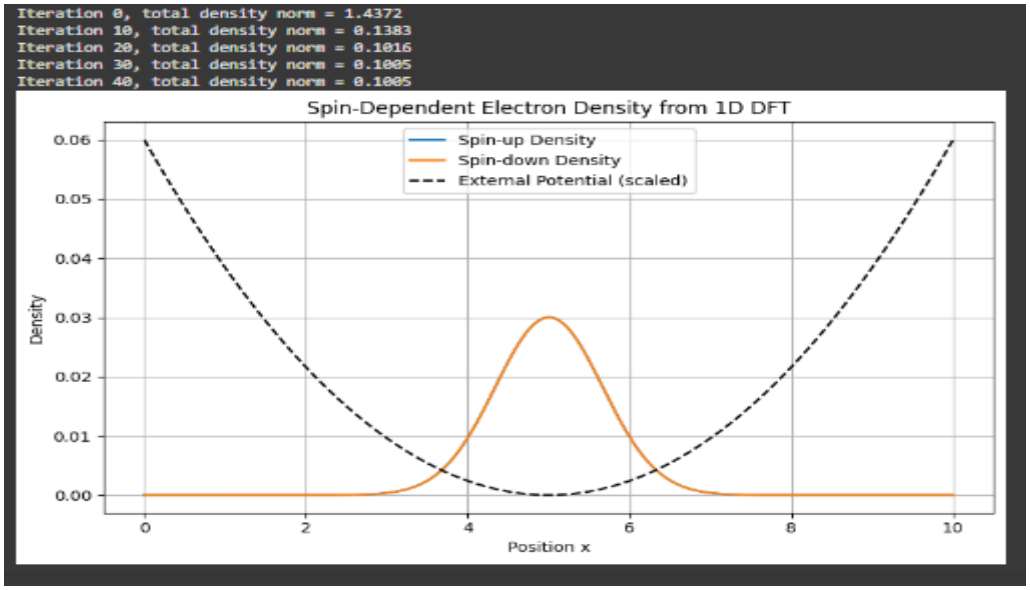


Fig 7: Visualisation of electron density friom 1D DFT [2].

The graph in fig 7 shows the spin-dependent electron density in a 1D quantum dot (QDot) obtained via Density Functional Theory (DFT), emphasizing how interaction energy shapes the qubit environment for operations such as the RY(π/2) gate.

1. Asymmetry in Spin Densities

- The graph shows dominant spin-down electron density while the spin-up density is nearly zero.
- This asymmetry arises due to the exchange interaction energy, which favors spin polarization in low-density regimes.
- In DFT terms, this is modeled by the exchange-correlation (XC) potential, which penalizes configurations with overlapping spin-up and spin-down electrons.

2. Localization Due to Confinement and Hartree Repulsion

- The sharp peak at the center of the dot reflects the combined effect of external parabolic confinement and Hartree interaction energy (electron–electron repulsion).
- The electron is pushed toward the center, but the peak is not infinitely narrow because Hartree energy opposes extreme localization, maintaining a stable wavefunction spread.

3. Influence on RY Gate Fidelity

- The interaction energies (Hartree + Exchange) determine the energy spacing and curvature of the potential in which the spin evolves.
- These internal forces shape the qubit's effective Hamiltonian, influencing how accurately the RY(θ) gate rotates the spin.
    - A well-behaved density implies a predictable precession axis and minimal decoherence.
    - If interaction energy causes spatial or spin mixing, RY gates would rotate on distorted Bloch spheres, reducing gate fidelity.

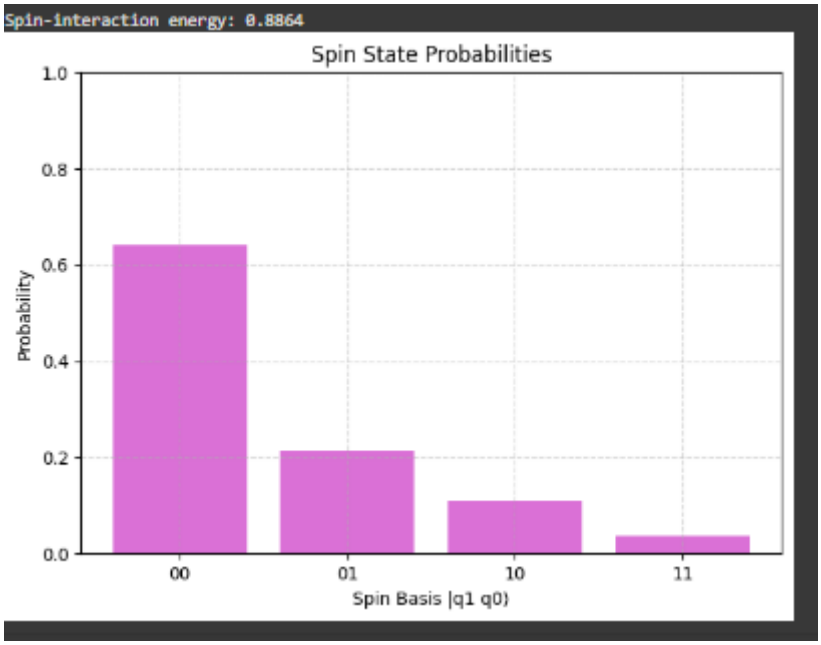


Fig 8: Probabilty destribution of of spin states for 2 qubit system[2].

This bar chart displays the probability distribution of spin states for a two-qubit system after applying RY gates on both qubits, with an underlying spin–spin interaction Hamiltonian. The horizontal axis represents computational basis states |q1q0⟩, and the vertical axis shows the probability of each state.

- The state |00⟩ dominates with 65% probability, indicating that the system tends to favor aligned spins in the absence of spin flips.
- Less probable outcomes like |11⟩ and |10⟩ reflect the rotation effect of the RY gates and energy penalties imposed by the spin interaction term.
- Spin-interaction energy: 0.8864 quantifies the total energy from the Hamiltonian

4.Conclusion

This project analyzed how interaction energy influences the operation of RY gates in quantum dot (QDot) systems, using both DFT simulations and spin-based quantum models. The DFT analysis revealed that exchange and Hartree interactions shape a spin-polarized electron density, ideal for stable qubit initialization. The spin simulations showed that spin–spin interaction energy (J) directly affects gate outcomes and state probabilities after RY rotations. Together, these results demonstrate that interaction energy is a critical factor in qubit behavior, gate fidelity, and system design. Accurate modeling using DFT and quantum circuits is essential for building scalable, high-performance QDot-based quantum computing systems.

5.References:

[1] D. Loss and D. P. DiVincenzo,"**Quantum computation with quantum dots**," *Physical Review A*, vol. 57, no. 1, pp. 120–126, Jan. 1998, doi: 10.1103/PhysRevA.57.120.

[2] H. Jiang, H. U. Baranger, and W. Yang, "**Density-functional theory simulation of large quantum dots**,"*Physical Review B*, vol. 68, no. 16, p. 165337, Oct. 2003, doi: 10.1103/PhysRevB.68.165337

[3] H.-R. Wei and F.-G. Deng, "**Scalable quantum computing based on stationary spin qubits in coupled quantum dots inside double-sided optical microcavities**,"*Scientific Reports*, vol. 4, no. 1, p. 7551, Dec. 2014, doi: 10.1038/srep07551

[4] F. Jazaeri, A. Beckers, A. Tajalli, and J.-M. Sallese, "**A review on quantum computing: Qubits, cryogenic electronics and cryogenic MOSFET physics**," *arXiv preprint arXiv:1908.02656*, Aug. 2019.

Appendix:

Block sphere representation:

```python
from qiskit import QuantumCircuit
from qiskit_aer import AerSimulator
from qiskit.visualization import plot_bloch_vector
import matplotlib.pyplot as plt
from math import pi, cos, sin

# Step 1: Create a quantum circuit
qc = QuantumCircuit(1)

# Step 2: Apply the RY gate with a specific angle
theta = pi / 2  # You can modify this value to try RY(pi), RY(pi/4), etc.
qc.ry(theta, 0)

# Step 3: Use the AerSimulator to get the statevector
simulator = AerSimulator(method='statevector')
qc.save_statevector()  # Save the statevector to extract it later
result = simulator.run(qc).result()
statevector = result.get_statevector()

# Step 4: Convert the statevector to a Bloch vector
# Statevector: [a, b] → Bloch vector: x = 2Re(a*b*), y = 2Im(a*b*), z = |a|^2 - |b|^2
a = statevector[0]
b = statevector[1]
x = 2 * (a.real * b.real + a.imag * b.imag)
y = 2 * (a.imag * b.real - a.real * b.imag)
z = abs(a)**2 - abs(b)**2

# Step 5: Plot the Bloch vector
bloch_vector = [x, y, z]
plot_bloch_vector(bloch_vector, title=f"Bloch Sphere after RY({theta:.2f})")
plt.show()
```

from qiskit import QuantumCircuit

from qiskit_aer import AerSimulator

import numpy as np

import matplotlib.pyplot as plt

from math import pi

Define the circuit and RY gate

theta = pi / 2 # Rotation angle (try pi, pi/4, etc.)

qc = QuantumCircuit(1)

qc.ry(theta, 0)

```
# Simulate using AerSimulator with 'statevector' method
sim = AerSimulator(method='statevector')
qc.save_statevector()
result = sim.run(qc).result()
statevector = result.get_statevector()

# Extract amplitudes
a = statevector[0]
b = statevector[1]

# Convert statevector to Bloch vector
x = 2 * (a.real * b.real + a.imag * b.imag)
y = 2 * (a.imag * b.real - a.real * b.imag)
z = abs(a)**2 - abs(b)**2

# Plotting manually using matplotlib (since plot_bloch_vector not allowed)
fig = plt.figure(figsize=(6, 6))
ax = fig.add_subplot(111, projection='3d')

# Draw sphere
u = np.linspace(0, 2 * pi, 100)
v = np.linspace(0, pi, 100)
X = np.outer(np.cos(u), np.sin(v))
Y = np.outer(np.sin(u), np.sin(v))
Z = np.outer(np.ones(np.size(u)), np.cos(v))
ax.plot_surface(X, Y, Z, color='lightblue', alpha=0.1, edgecolor='k', linewidth=0.1)

# Coordinate axes
ax.quiver(0, 0, 0, 1, 0, 0, color='r', arrow_length_ratio=0.1, label='X')
ax.quiver(0, 0, 0, 0, 1, 0, color='g', arrow_length_ratio=0.1, label='Y')
```

```
ax.quiver(0, 0, 0, 0, 0, 1, color='b', arrow_length_ratio=0.1, label='Z')

# Bloch vector (final spin state)
ax.quiver(0, 0, 0, x, y, z, color='purple', arrow_length_ratio=0.2)

# Formatting
ax.set_xlim([-1.1, 1.1])
ax.set_ylim([-1.1, 1.1])
ax.set_zlim([-1.1, 1.1])
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title(f"Bloch Sphere after RY({theta:.2f})")
ax.view_init(elev=30, azim=45)
plt.tight_layout()
plt.show()



from qiskit import QuantumCircuit
from qiskit_aer import AerSimulator
import matplotlib.pyplot as plt
import numpy as np
from math import pi

# Create circuit and apply RY(π/2)
qc = QuantumCircuit(1)
theta = pi / 2
qc.ry(theta, 0)

# Simulate using AerSimulator (statevector method)
```

```
sim = AerSimulator(method='statevector')
qc.save_statevector()
result = sim.run(qc).result()
statevector = result.get_statevector()

# Calculate Bloch vector from statevector
a = statevector[0]
b = statevector[1]
x = 2 * (a.real * b.real + a.imag * b.imag)
y = 2 * (a.imag * b.real - a.real * b.imag)
z = abs(a)**2 - abs(b)**2

# Initial and final Bloch vectors
initial_vector = [0, 0, 1]  # |0⟩ state
final_vector = [x, y, z]    # After RY(π/2)

# Bloch sphere plot
fig = plt.figure(figsize=(7, 7))
ax = fig.add_subplot(111, projection='3d')

# Draw the Bloch sphere
u = np.linspace(0, 2 * pi, 100)
v = np.linspace(0, pi, 100)
X = np.outer(np.cos(u), np.sin(v))
Y = np.outer(np.sin(u), np.sin(v))
Z = np.outer(np.ones(np.size(u)), np.cos(v))
ax.plot_surface(X, Y, Z, color='lightblue', alpha=0.1)

# Axes
ax.quiver(0, 0, 0, 1, 0, 0, color='r', label='X-axis')
```

```
ax.quiver(0, 0, 0, 0, 1, 0, color='g', label='Y-axis')
ax.quiver(0, 0, 0, 0, 0, 1, color='b', label='Z-axis')

# Plot initial and final Bloch vectors
ax.quiver(0, 0, 0, *initial_vector, color='black', linewidth=2, label='Initial |0⟩')
ax.quiver(0, 0, 0, *final_vector, color='purple', linewidth=2, label='After RY(π/2)')

# Labels and formatting
ax.set_xlim([-1.1, 1.1])
ax.set_ylim([-1.1, 1.1])
ax.set_zlim([-1.1, 1.1])
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title("Bloch Sphere: RY(π/2) on |0⟩")
ax.legend()
ax.view_init(elev=30, azim=45)

plt.tight_layout()
plt.show()
```

DFT Graph simulation:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.linalg import eigh_tridiagonal

# === Grid setup ===
Nx = 200                  # number of grid points
L = 10.0                  # box size
x = np.linspace(0, L, Nx)
dx = x[1] - x[0]
```

```
# === Electron setup ===
N_up = 1                  # spin-up electrons
N_down = 1                # spin-down electrons
total_e = N_up + N_down

# === External potential: parabolic well ===
V_ext = 0.5 * (x - L/2)**2

# === Kinetic energy operator (finite difference) ===
t_const = -0.5 / dx**2
d = np.full(Nx, -2.0 * t_const)
e = np.full(Nx - 1, t_const)

# === Initial density guess ===
n_up = np.ones(Nx) * 0.1
n_down = np.ones(Nx) * 0.1

# === Exchange-correlation potential (local spin-density approx) ===
def v_xc(n_up, n_down):
    n_total = n_up + n_down + 1e-10
    rs = (3 / (4 * np.pi * n_total))**(1/3)
    Vx_up = - (3 / np.pi)**(1/3) * (n_up**(1/3))
    Vx_down = - (3 / np.pi)**(1/3) * (n_down**(1/3))
    return Vx_up, Vx_down

# === Hartree potential (1D convolution) ===
def v_hartree(n):
    return np.convolve(n, np.abs(x - x[Nx//2]), mode='same') * dx

# === Self-consistent loop ===
```

```
mixing = 0.3
for iteration in range(50):
    # Hartree + XC potentials
    V_H = v_hartree(n_up + n_down)
    Vxc_up, Vxc_down = v_xc(n_up, n_down)

    # KS potentials
    V_ks_up = V_ext + V_H + Vxc_up
    V_ks_down = V_ext + V_H + Vxc_down

    # Solve KS equations (eigenvalue problem)
    eps_up, psi_up = eigh_tridiagonal(d + V_ks_up, e)
    eps_down, psi_down = eigh_tridiagonal(d + V_ks_down, e)

    # Build new density
    n_up_new = np.sum(np.abs(psi_up[:, :N_up])**2, axis=1)
    n_down_new = np.sum(np.abs(psi_down[:, :N_down])**2, axis=1)

    # Mix new and old densities
    n_up = (1 - mixing) * n_up + mixing * n_up_new
    n_down = (1 - mixing) * n_down + mixing * n_down_new
    # Convergence check (optional)
    if iteration % 10 == 0:
        total_n = n_up + n_down
        print(f"Iteration {iteration}, total density norm = {np.sum(total_n)*dx:.4f}")
# === Plot results ===
plt.figure(figsize=(8, 5))
plt.plot(x, n_up, label='Spin-up Density')
plt.plot(x, n_down, label='Spin-down Density')
plt.plot(x, V_ext / np.max(V_ext) * np.max(n_up + n_down), 'k--', label='External Potential (scaled)')
plt.title('Spin-Dependent Electron Density from 1D DFT')
```

```
plt.xlabel('Position x')
plt.ylabel('Density')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Spin qubit represeantaion:

```
!pip install qiskit
!pip install qiskit.quantum_info
from qiskit import QuantumCircuit
from qiskit.quantum_info import Statevector, Operator, Pauli
import numpy as np
from numpy import pi
import matplotlib.pyplot as plt

# Create spin configuration via RY gates
theta1 = pi / 3  # controls spin state of qubit 0
theta2 = pi / 4  # controls spin state of qubit 1

qc = QuantumCircuit(2)
qc.ry(theta1, 0)
qc.ry(theta2, 1)

# Get statevector
state = Statevector.from_instruction(qc)

# Spin-based interaction Hamiltonian: like SDFT energy
Z = Pauli("Z")
I = Pauli("I")
```

```
# Simple spin interaction Hamiltonian (DFT-inspired):
# ε1 Z0 + ε2 Z1 + J Z0 Z1
ε1 = 0.5
ε2 = 0.5
J = 0.8
H = ε1 * Operator(Z.tensor(I)) + ε2 * Operator(I.tensor(Z)) + J * Operator(Z.tensor(Z))

# Compute energy
energy = np.real(state.expectation_value(H))
print(f"Spin-interaction energy: {energy:.4f}")

# Probabilities
probs = state.probabilities_dict()
plt.bar(probs.keys(), probs.values(), color='orchid')
plt.title("Spin State Probabilities")
plt.xlabel("Spin Basis |q1 q0⟩")
plt.ylabel("Probability")
plt.grid(True, linestyle='--', alpha=0.5)
plt.ylim(0, 1.0)
plt.tight_layout()
plt.show()
```