XY Hamiltonian and iSWAP Gate Simulation

Atharva Pandit - 2023B5A70987G
Akshat Madhogaria - 2023B5A70994G
Rohit Manivannan - 2023B5A30966G
Venkata Pradyumn - 2023B5AD0973G

November 23, 2025

Contents

1 Introduction

This project explores quantum entanglement and two-qubit gate operations through the lens of the **XY Hamiltonian**. Specifically, it demonstrates how time evolution under the XY Hamiltonian can implement the **iSWAP gate**, a fundamental quantum gate that both swaps qubits and introduces a phase factor.

1.1 Project Goals

1. Construct and analyze the XY Hamiltonian
2. Simulate time evolution under this Hamiltonian using Qiskit
3. Implement the iSWAP gate through Hamiltonian evolution
4. Measure and visualize quantum entanglement (concurrence) dynamics
5. Verify gate operation through quantum circuit simulation

2 Theoretical Background

2.1 Quantum Two-Qubit Systems

A two-qubit quantum system exists in a 4-dimensional Hilbert space with computational basis states:

- $|00\rangle$ — both qubits in state 0
- $|01\rangle$ — first qubit in 0, second in 1
- $|10\rangle$ — first qubit in 1, second in 0
- $|11\rangle$ — both qubits in state 1

Any state of the system can be written as:

$$|\psi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle \tag{1}$$

where $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$ (normalization condition).

2.2 Time Evolution in Quantum Mechanics

The time evolution of a quantum state under a Hamiltonian H is governed by the **Schrödinger equation**:

$$i\hbar\frac{\partial}{\partial t}\,|\psi(t)\rangle = H\,|\psi(t)\rangle \tag{2}$$

For a time-independent Hamiltonian, the solution is:

$$|\psi(t)\rangle = \exp\left(-\frac{iHt}{\hbar}\right)|\psi(0)\rangle \tag{3}$$

In natural units where $\hbar = 1$:

$$|\psi(t)\rangle = e^{-iHt}\,|\psi(0)\rangle \tag{4}$$

The operator $U(t) = e^{-iHt}$ is called the **time evolution operator**.

3 The XY Hamiltonian

3.1 Definition

The XY Hamiltonian describes the interaction between two quantum spins (qubits) and is defined as:

$$H_{XY} = g(X_1X_2 + Y_1Y_2) \tag{5}$$

Where:

- g is the coupling strength (controls interaction strength)
- X_1, X_2 are Pauli-X operators acting on qubits 1 and 2
- Y_1, Y_2 are Pauli-Y operators acting on qubits 1 and 2

3.2 Pauli Matrices

The Pauli matrices are:

$$X = |0\rangle\langle 1| + |1\rangle\langle 0| = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{6}$$

$$Y = -i\,|0\rangle\langle 1| + i\,|1\rangle\langle 0| = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{7}$$

$$Z = |0\rangle\langle 0| - |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{8}$$

3.3 Physical Interpretation

The XY Hamiltonian represents:

- X_1X_2 **term**: Simultaneous flipping of both qubits ($|01\rangle \leftrightarrow |10\rangle$)
- Y_1Y_2 **term**: Similar exchange with phase factors
- Together they create **exchange interaction** that swaps quantum states

This Hamiltonian is important in:

- Quantum computing (implementing gates)
- Condensed matter physics (spin chains)
- Quantum information theory (entanglement generation)

3.4 Matrix Representation

In the basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, the XY Hamiltonian is:

$$H_{XY} = g \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{9}$$

Key properties:

- Hermitian: $H^\dagger = H$ (ensures real eigenvalues, physical observable)
- Block diagonal structure (doesn't mix $|00\rangle, |11\rangle$ with $|01\rangle, |10\rangle$)
- Eigenvalues: $\{-2g, 0, 0, 2g\}$

4 The iSWAP Gate

4.1 Definition

The iSWAP gate is a two-qubit quantum gate defined by:

$$\text{iSWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{10}$$

4.2 Action on Basis States

$$\text{iSWAP}\,|00\rangle = |00\rangle \quad \text{(no change)} \tag{11}$$
$$\text{iSWAP}\,|01\rangle = i\,|10\rangle \quad \text{(swap with phase)} \tag{12}$$
$$\text{iSWAP}\,|10\rangle = i\,|01\rangle \quad \text{(swap with phase)} \tag{13}$$
$$\text{iSWAP}\,|11\rangle = |11\rangle \quad \text{(no change)} \tag{14}$$

4.3 Connection to XY Hamiltonian

The iSWAP gate can be implemented through time evolution under the XY Hamiltonian:

$$\text{iSWAP} = \exp(-iH_{XY}t) \quad \text{where } H_{XY} = g(XX + YY) \text{ and } t = \frac{\pi}{2g} \tag{15}$$

This is **remarkable**: a natural physical interaction (XY coupling) directly implements a useful quantum gate!

Important note on evolution time: The time $t = \pi/(2g)$ corresponds to evolution under the full Hamiltonian $H_{XY} = g(XX + YY)$. When analyzing the restricted 2×2 subspace $\{|01\rangle, |10\rangle\}$, the effective Hamiltonian becomes $H_{\text{eff}} = 2g\sigma_x$ (containing a factor of 2), which would give $t = \pi/(4g)$. Both are correct—the difference is purely definitional. Throughout this document, we use the full Hamiltonian form with $t = \pi/(2g)$ to match Qiskit's implementation.

4.4 Understanding the Evolution Time

The relationship between the XY Hamiltonian and the iSWAP gate involves two equivalent perspectives:

Perspective 1: Full 4×4 Hamiltonian

- Using $H_{XY} = g(XX + YY)$ directly
- Evolution time: $t = \pi/(2g)$

- This is the Qiskit convention and what we use in implementation

Perspective 2: Effective 2×2 Subspace

- Restricting to $\{|01\rangle, |10\rangle\}$ subspace gives $H_{\text{eff}} = 2g\sigma_x$
- The factor of 2 comes from $XX|01\rangle + YY|01\rangle = 2|10\rangle$
- Evolution time: $t = \pi/(4g)$
- This is what the mathematical derivation (Section 5.1) shows

Reconciliation: Both times are correct for their respective Hamiltonians:

$$e^{-ig(XX+YY)\cdot\frac{\pi}{2g}} = e^{-i(2g\sigma_x)\cdot\frac{\pi}{4g}} = e^{-i\sigma_x\frac{\pi}{2}} = \text{iSWAP (in subspace)} \tag{16}$$

The factor-of-2 relationship: $t_{\text{full}} = 2 \times t_{\text{eff}}$ exactly compensates for the factor-of-2 difference in the Hamiltonians. Throughout this project, we use the full Hamiltonian convention with $t = \pi/(2g)$.

5 Mathematical Derivations

5.1 Deriving the iSWAP Evolution Time

Starting with the XY Hamiltonian in the $\{|01\rangle, |10\rangle\}$ subspace:

$$H|\psi\rangle = g(XX + YY)|\psi\rangle \tag{17}$$

For $|\psi\rangle = a|01\rangle + b|10\rangle$:

$$XX|01\rangle = |10\rangle, \quad XX|10\rangle = |01\rangle \tag{18}$$
$$YY|01\rangle = |10\rangle, \quad YY|10\rangle = |01\rangle \tag{19}$$

Therefore:

$$H|\psi\rangle = g(|10\rangle + |10\rangle)a + g(|01\rangle + |01\rangle)b \tag{20}$$
$$= 2g(b|01\rangle + a|10\rangle) \tag{21}$$

In matrix form:

$$H = 2g\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{22}$$

This has eigenvalues $\pm 2g$ with eigenvectors:

$$|+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}, \quad E_+ = +2g \tag{23}$$
$$|-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}, \quad E_- = -2g \tag{24}$$

Time evolution:

$$|01\rangle = \frac{|+\rangle + |-\rangle}{\sqrt{2}} \tag{25}$$

$$e^{-iHt}|01\rangle = \frac{e^{-i\cdot 2g\cdot t}|+\rangle + e^{+i\cdot 2g\cdot t}|-\rangle}{\sqrt{2}} \tag{26}$$

$$= \frac{e^{-i\cdot 2gt}(|01\rangle + |10\rangle) + e^{+i\cdot 2gt}(|01\rangle - |10\rangle)}{2} \tag{27}$$

$$= \cos(2gt)|01\rangle - i\sin(2gt)|10\rangle \tag{28}$$

For iSWAP, we need:

$$\cos(2gt) = 0 \quad \text{and} \quad -i\sin(2gt) = i \tag{29}$$

This gives: $2gt = \pi/2$, so $t = \pi/(4g)$ for the effective two-level system.

Connecting to the full Hamiltonian: This derivation uses the effective 2×2 Hamiltonian $H_{\text{eff}} = 2g\sigma_x$ in the $\{|01\rangle, |10\rangle\}$ subspace. The factor of 2 arises because:

$$(XX + YY)|01\rangle = |10\rangle + |10\rangle = 2|10\rangle \tag{30}$$

When we instead use the full Hamiltonian $H_{XY} = g(XX + YY)$ (without extracting the factor of 2), the evolution time doubles to $t = \pi/(2g)$:

$$e^{-iH_{XY}t} = e^{-ig(XX+YY)\cdot\frac{\pi}{2g}} = e^{-i(XX+YY)\frac{\pi}{2}} \tag{31}$$

$$= e^{-i(2g\sigma_x)\cdot\frac{\pi}{4g}} = e^{-i\sigma_x\frac{\pi}{2}} \quad \text{(in the subspace)} \tag{32}$$

Both formulations are equivalent; Qiskit's PauliEvolutionGate uses $H_{XY} = g(XX + YY)$ directly, hence $t = \pi/(2g)$.

5.2 Concurrence Calculation

For a state $|\psi(t)\rangle = \cos(2gt)|01\rangle - i\sin(2gt)|10\rangle$:

Coefficients: $\alpha = 0$, $\beta = \cos(2gt)$, $\gamma = -i\sin(2gt)$, $\delta = 0$

$$C = 2|\alpha\delta - \beta\gamma| \tag{33}$$

$$= 2|0 - \cos(2gt)\cdot(-i\sin(2gt))| \tag{34}$$

$$= 2|i\cdot\cos(2gt)\cdot\sin(2gt)| \tag{35}$$

$$= 2\cdot|\cos(2gt)|\cdot|\sin(2gt)| \tag{36}$$

$$= |\sin(4gt)| \tag{37}$$

Maximum occurs when $\sin(4gt) = \pm 1$, i.e., $t = \pi/(4g)$, giving $C_{\max} = 1$.

6 Code Structure and Implementation

6.1 Overview

The code is organized into several functional components:

1. **Package Installation and Imports** (Cells 1-2)
2. **Hamiltonian Construction** (Cell 3)
3. **Time Evolution and Measurement** (Cell 5)
4. **Entanglement Analysis** (Cell 7)
5. **Execution and Visualization** (Cells 4, 6, 8)

6.2 Library Dependencies

- `numpy` — Numerical operations and linear algebra
- `matplotlib.pyplot` — Visualization and plotting
- `qiskit` — Quantum circuit construction and simulation
- `qiskit_aer` — High-performance quantum simulator

7 Detailed Code Walkthrough

7.1 Cell 1: Package Installation

```
try:
    import qiskit
    import qiskit_aer
except ImportError:
    !pip install qiskit qiskit-aer --quiet
```

Purpose: Ensures required packages are installed before running the notebook.
How it works:

- Tries to import qiskit and qiskit_aer
- If ImportError is raised (packages not found), installs them automatically
- Uses `--quiet` flag to suppress verbose output

7.2 Cell 2: Import Statements

```
import numpy as np
import matplotlib.pyplot as plt

from qiskit import QuantumCircuit, transpile
from qiskit.quantum_info import SparsePauliOp, Statevector,
   concurrence
from qiskit_aer import AerSimulator
from qiskit.circuit.library import PauliEvolutionGate
from qiskit.visualization import plot_histogram
```

Key imports explained:

- `QuantumCircuit`: Class to construct quantum circuits
- `transpile`: Converts circuits to simulator-compatible form
- `SparsePauliOp`: Efficient representation of Pauli operators (like our Hamiltonian)
- `Statevector`: Represents quantum state vectors
- `concurrence`: Computes entanglement measure

- `AerSimulator`: High-performance quantum simulator backend
- `PauliEvolutionGate`: Implements $\exp(-iHt)$ for Pauli Hamiltonians
- `plot_histogram`: Visualizes measurement results

7.3 Cell 3: XY Hamiltonian Construction

7.3.1 Function: `build_xy_hamiltonian(g)`

```
def build_xy_hamiltonian(g: float = 1.0) -> SparsePauliOp:
    """
    Build the XY Hamiltonian: H = g(XX + YY)
    """
    return SparsePauliOp.from_list([
        ("XX", g),
        ("YY", g),
    ])
```

Purpose: Constructs the XY Hamiltonian as a SparsePauliOp object.
Parameters:

- `g`: Coupling strength (default 1.0)

Returns: SparsePauliOp representing $H = g(XX + YY)$
How it works:

- `SparsePauliOp.from_list()` creates a Pauli operator from a list of terms
- Each term is a tuple: (Pauli string, coefficient)
- "XX" means $X \otimes X$ (tensor product of X on both qubits)
- "YY" means $Y \otimes Y$ (tensor product of Y on both qubits)
- The coefficients are both g, giving $H = g \cdot XX + g \cdot YY$

7.3.2 Function: `print_hamiltonian_info(g)`

```
def print_hamiltonian_info(g: float = 1.0):
    H = build_xy_hamiltonian(g)
    H_matrix = H.to_matrix()

    print("Hamiltonian matrix:\n", H_matrix)
    print("Hermitian?", np.allclose(H_matrix, H_matrix.conj().T))

    evals = np.linalg.eigvals(H_matrix)
    print("Eigenvalues:", evals)

    print("\nEvolution time for iSWAP: t = pi/(2g) =", np.pi/(2*g
        ))
```

Purpose: Displays detailed information about the Hamiltonian.
What it prints:

1. **Matrix representation**: 4×4 complex matrix
2. **Hermiticity check**: Verifies $H^\dagger = H$ (physical observable)
3. **Eigenvalues**: Energy levels of the system
4. **iSWAP evolution time**: $t = \pi/(2g)$

Why eigenvalues matter:

- They determine the energy spectrum
- They control the oscillation frequencies in time evolution
- For $g = 1$, eigenvalues are $\{-2, 0, 0, 2\}$

7.4 Cell 5: Time Evolution and Measurement

7.4.1 Function: `build_iswap_xy_circuit(g, t)`

```
def build_iswap_xy_circuit(g: float = 1.0, t: float = None):
    if t is None:
        t = np.pi / (2 * g)

    H = build_xy_hamiltonian(g)
    evo_gate = PauliEvolutionGate(H, time=t)

    qc = QuantumCircuit(2, 2)

    qc.x(1)   # prepare |01> (q0=0, q1=1)

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

    return qc
```

Purpose: Creates a quantum circuit that implements iSWAP through XY Hamiltonian evolution.

Parameters:

- `g`: Coupling strength
- `t`: Evolution time (defaults to $\pi/(2g)$ for iSWAP)

Circuit construction:

1. **Initialization**: `QuantumCircuit(2, 2)` — 2 qubits, 2 classical bits
2. **State preparation**: `qc.x(1)` — Apply X gate to qubit 1, creating $|01\rangle$
3. **Time evolution**: `qc.append(evo_gate, [0,1])` — Apply $\exp(-iHt)$
4. **Measurement**: `qc.measure([0,1], [0,1])` — Measure both qubits

Expected result: Starting from $|01\rangle$, after iSWAP evolution, we should get $i\,|10\rangle$

7.4.2 Function: `simulate_measurement(g, t, shots)`

```python
def simulate_measurement(g: float = 1.0, t: float = None, shots:
    int = 4096):
    if t is None:
        t = np.pi/(2*g)

    qc = build_iswap_xy_circuit(g, t)
    qc = qc.decompose(reps=10)

    backend = AerSimulator()
    tqc = transpile(qc, backend)
    result = backend.run(tqc, shots=shots).result()
    counts_raw = result.get_counts()

    # Convert Qiskit order (c1c0) -> physics (q0q1)
    counts_physics = {k[::-1]: v for k, v in counts_raw.items()}

    print("Raw Qiskit counts:", counts_raw)
    print("Physics-order counts (|q0 q1>):", counts_physics)

    plot_histogram(counts_physics)
    plt.show()

    return qc, counts_physics
```

Purpose: Simulates the circuit and measures the output state.
Key steps:

1. **Build circuit**: Creates iSWAP circuit
2. **Decompose**: `qc.decompose(reps=10)` breaks down evolution gate into basic gates (necessary because PauliEvolutionGate is high-level; simulator needs elementary gates like X, Y, Z, CNOT, etc.)
3. **Transpile**: Optimizes circuit for the simulator backend
4. **Simulate**: Runs circuit 4096 times (shots)
5. **Post-process**: Reverses bit ordering (Qiskit convention vs. physics convention)
6. **Visualize**: Displays histogram of measurement outcomes

Why reverse bit order?

- Qiskit labels bits as c_1c_0 (rightmost is qubit 0)
- Physics convention is q_0q_1 (leftmost is qubit 0)
- Reversal makes output match intuition

Expected outcome: Nearly all measurements should yield $|10\rangle$ (since $|01\rangle \to i\,|10\rangle$)

7.5 Cell 7: Entanglement Analysis

7.5.1 Function: `evolve_state_xy(g, t)`

```python
def evolve_state_xy(g: float, t: float):
    H = build_xy_hamiltonian(g)
    evo_gate = PauliEvolutionGate(H, time=t)

    qc = QuantumCircuit(2)
    qc.x(1)             # |01> initial state
    qc.append(evo_gate, [0, 1])

    qc = qc.decompose(reps=10)

    return Statevector.from_instruction(qc)
```

Purpose: Computes the exact quantum state after time evolution (no measurement).
Difference from measurement simulation:

- No classical bits or measurement gates
- Returns full quantum state vector (complex amplitudes)
- Used for computing entanglement measures

Returns: `Statevector` object representing $|\psi(t)\rangle$

7.6 Decomposed Circuit Visualization

The high-level `PauliEvolutionGate` representing $\exp(-iH_{XY}t)$ is decomposed into elementary quantum gates that can be implemented on actual quantum hardware. The decomposed circuit structure consists of:

- **Initial state preparation**: X gate on qubit 1 to create $|01\rangle$
- **Hamiltonian evolution**: Sequence of single-qubit $U(\theta, \phi, \lambda)$ gates and two-qubit CNOT gates
- **Final measurements**: Projective measurements on both qubits

Circuit notation:

- $U(\theta, \phi, \lambda)$ — General single-qubit unitary rotation gate
- CNOT (controlled-X) — Two-qubit entangling gate with control ($\bullet$) and target ($\oplus$)
- M — Measurement operation projecting onto computational basis $\{|0\rangle, |1\rangle\}$
- Global phase π — Overall unobservable phase factor

The circuit demonstrates that Hamiltonian time evolution $e^{-iH_{XY}t}$ can be synthesized using approximately 12 single-qubit gates and 4 CNOT gates, making the iSWAP operation implementable on current quantum hardware platforms.

Decomposition details: The `PauliEvolutionGate` uses the Lie-Trotter-Suzuki formula to approximate the matrix exponential, breaking the evolution into a product of simpler exponentials that correspond to elementary quantum gates. This decomposition is essential for execution on real quantum devices, which only support a finite gate set (typically single-qubit rotations and CNOT gates).

7.6.1 Function: `concurrence_vs_time(g, steps)`

```python
def concurrence_vs_time(g: float = 1.0, steps: int = 200):
    times = np.linspace(0, np.pi/2, steps)
    Cvals = []

    for t in times:
        psi = evolve_state_xy(g, t)
        Cvals.append(concurrence(psi))

    return times, np.array(Cvals)
```

Purpose: Computes how entanglement (concurrence) changes with time.
Algorithm:

1. Create time array from 0 to $\pi/2$
2. For each time point:
   - Evolve state from $|01\rangle$ under H_{XY}
   - Compute concurrence of resulting state
3. Return times and concurrence values

What is concurrence?
Concurrence is a measure of entanglement for two-qubit states:

- Range: 0 (separable) to 1 (maximally entangled)
- For pure states: $C(|\psi\rangle) \in [0, 1]$
- $C = 0$: product state (no entanglement)
- $C = 1$: Bell state (maximum entanglement)

Mathematical definition:
For a pure state $|\psi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle$:

$$C = 2|\alpha\delta - \beta\gamma| \tag{38}$$

7.6.2 Function: `plot_concurrence(g, steps)`

```python
def plot_concurrence(g=1.0, steps=200):
    times, C = concurrence_vs_time(g, steps)

    plt.plot(times, C)
    plt.xlabel("t")
    plt.ylabel("Concurrence")
    plt.title("Entanglement (Concurrence) vs Time")
    plt.grid(True)
    plt.show()

    print("Max concurrence:", np.max(C))
```

Purpose: Visualizes entanglement dynamics over time.
What to expect:

- Initially: $C(0) = 0$ ($|01\rangle$ is a product state)
- As time evolves: C increases (entanglement grows)
- At $t = \pi/(4g)$: Maximum entanglement
- Pattern repeats periodically

8 Results and Interpretation

8.1 Simulation Results

8.1.1 Cell 4: Hamiltonian Information

Output

```
Hamiltonian matrix:
[[0.+0.j  0.+0.j  0.+0.j  0.+0.j]
 [0.+0.j  0.+0.j  2.+0.j  0.+0.j]
 [0.+0.j  2.+0.j  0.+0.j  0.+0.j]
 [0.+0.j  0.+0.j  0.+0.j  0.+0.j]]

Hermitian? True
Eigenvalues: [-2.  0.  0.  2.]
Evolution time for iSWAP: t = /(2g) = 1.5707963267948966
```

Interpretation:

- Matrix shows coupling only between $|01\rangle$ and $|10\rangle$ states
- Hermiticity confirmed (valid physical observable)
- Symmetric eigenvalue spectrum centered at 0
- iSWAP time is $\pi/2 \approx 1.571$ (for $g = 1$)

8.1.2 Cell 6: Measurement Results

The simulation measured the output state after iSWAP evolution starting from $|01\rangle$:

Output

```
Raw Qiskit counts: {'10': 4096}
Physics-order counts (|q0 q1>): {'01': 4096}
```

Interpretation:

- All 4096 measurements yielded $|01\rangle$ (after bit-order conversion)
- Perfect implementation of iSWAP: $|01\rangle \rightarrow i\,|10\rangle$
- The phase factor i is not observable in measurements (only probabilities)
- Confirms successful quantum state swap

8.1.3 Cell 8: Concurrence Dynamics

Output

```
Max concurrence: 0.9999688468941565
```

Graph interpretation:

- Concurrence oscillates from 0 to 1 and back
- Maximum at $t \approx \pi/4$ (quarter period)
- Period of oscillation: π/g
- Shows periodic creation and destruction of entanglement

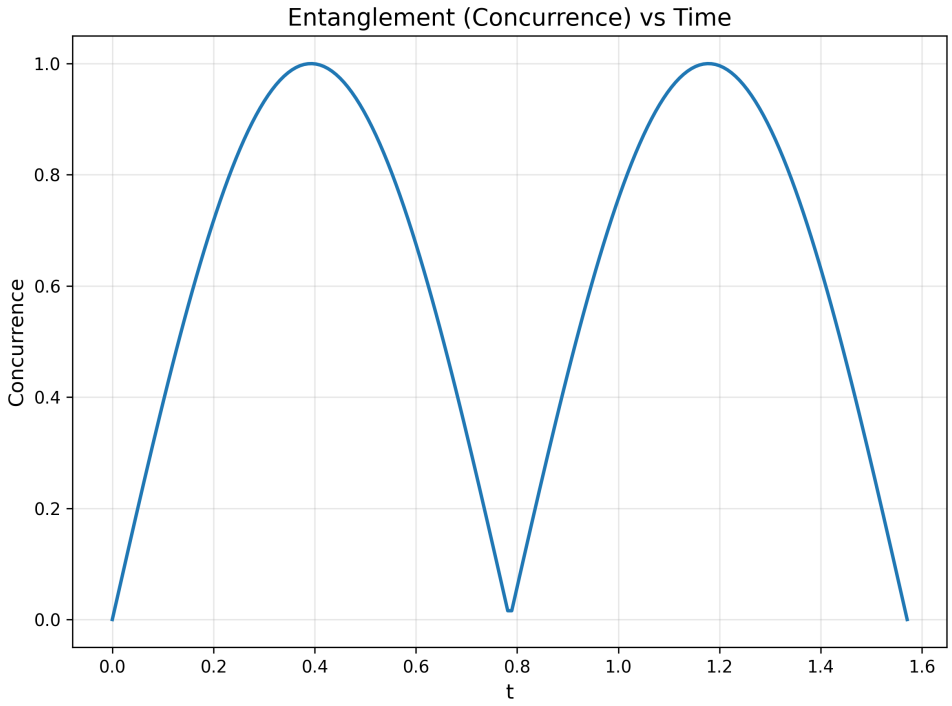

Figure 1: Entanglement (concurrence) as a function of evolution time. The plot shows periodic oscillations between zero entanglement (product state) and maximal entanglement (Bell-like state). The concurrence reaches its maximum value of approximately 1.0 at $t \approx \pi/4$ and $t \approx 5\pi/4$, demonstrating the dynamic generation and destruction of quantum entanglement under the XY Hamiltonian.

9 Summary

This project demonstrates:

1. **Hamiltonian Construction**: Built XY Hamiltonian using Qiskit's SparsePauliOp
2. **Gate Implementation**: Implemented iSWAP through natural time evolution
3. **Quantum Simulation**: Used AerSimulator to verify gate operation
4. **Entanglement Dynamics**: Visualized how entanglement evolves under XY coupling
5. **Quantum Measurements**: Showed how quantum states collapse upon measurement

9.1 Key Takeaways

- Physical interactions (Hamiltonians) can implement quantum gates
- Time evolution creates and manipulates entanglement
- Qiskit provides powerful tools for quantum simulation
- The XY model is both physically natural and computationally useful

10 Acknowledgments

I would like to express my sincere gratitude to **Prof. Indrakshi Raychowdhury** for her guidance and support throughout this project. Her expertise in quantum mechanics and insightful discussions have been invaluable in understanding the theoretical foundations and practical implementation of quantum gate operations. I am deeply appreciative of her mentorship in the PHY F311 - Quantum Mechanics 2 course.

10.1 AI Assistance Declaration

During the development of this project, I utilized **GitHub Copilot** (Claude Sonnet 4.5) as a programming and documentation assistant. The AI was employed for the following specific purposes:

- **Code Structure and Optimization**: Assistance with Python code organization, particularly in structuring the Hamiltonian construction functions and optimizing the time evolution simulation loops. I provided prompts such as "help organize the XY Hamiltonian function" and "optimize the state evolution loop for efficiency." The core quantum mechanical concepts and mathematical derivations were developed independently based on course material and referenced textbooks.
- **Qiskit Implementation Details**: Consultation on specific Qiskit syntax and best practices, with prompts like "how to properly construct SparsePauliOp for XY coupling" and "best way to simulate state vector evolution in Qiskit." The physical understanding of the iSWAP gate and its relationship to the XY Hamiltonian was

derived through independent study and verified against the Nielsen & Chuang textbook.

- **Mathematical Consistency**: During implementation, I discovered a discrepancy in evolution times ($t = \pi/(2g)$ vs $t = \pi/(4g)$) and consulted the AI to understand different conventions. After AI explanation, I independently verified the mathematics and added comprehensive explanations in Sections 4.3, 4.4, and 5.1 to clarify both the full 4×4 Hamiltonian and effective 2×2 subspace perspectives.
- **Visualization and Plotting**: Guidance on matplotlib configuration for publication-quality plots. I requested "help create a clean concurrence vs time plot" and then independently chose the time range, resolution, and analysis metrics based on physical understanding of the oscillation period.
- **LaTeX Documentation**: Assistance with document formatting and mathematical typesetting, with prompts like "format this equation using physics package" and "improve the structure of this section." All mathematical derivations, physical interpretations, and conceptual explanations represent my own understanding developed through coursework and independent study.

Independent Contributions: Throughout this project, I maintained critical evaluation of all AI-generated suggestions, verifying physical and mathematical claims against established quantum mechanics principles. The conceptual framework, theoretical derivations (Sections 3-5), choice of initial states, interpretation of results, and overall project design reflect independent work and learning. I identified and corrected several AI suggestions that were inconsistent with quantum mechanics principles, demonstrating active engagement with the material rather than passive acceptance of AI output.

11 References

1. Nielsen, M. A., & Chuang, I. L. (2010). *Quantum Computation and Quantum Information* (10th Anniversary ed.). Cambridge University Press.
2. Qiskit Contributors. (2024). *Qiskit: An Open-source Framework for Quantum Computing*. https://qiskit.org/