Calculating spin correlations with a IBM quantum computer

Naveen Vishwakarma

Abstract

In the rapidly growing field of quantum computing, which straddles the boundaries of physics, computer science, and engineering, computations that are impossible for classical computers to handle are attempted. Quantum computers use quantum bits, or qubits, which can simultaneously be both 0 and 1. This is in contrast to classical computers, which store information in binary bits that can either have a value of 0 or 1. This characteristic makes some calculations tenfold faster on quantum computers than on conventional ones. Quantum computing has the potential to revolutionize a number of industries, including machine learning, drug discovery, and optimization. For instance, many of the cryptographic methods used to safeguard sensitive information could be cracked by quantum computers, and they could mimic the behavior of intricate molecules to help develop new drugs. The delicate nature of qubits, which are often perturbed by outside noise and interactions with other qubits, makes it difficult to develop viable quantum computers. To solve these issues, researchers are investigating a range of strategies, such as the use of fault-tolerant structures, error-correcting codes, and new materials and technologies. Despite these obstacles, quantum computing has advanced quickly in recent years with the creation of processors with numerous qubits and the demonstration of quantum supremacy, in which a quantum computer successfully completes a task that is impossible for classical computers. The impact of quantum computing is anticipated to be significant as the technology develops.

1. Introduction

The use of quantum-mechanical phenomena, such as superposition, interference, and entanglement, to carry out calculations that are impossible for classical computers is known as quantum computation. In order to tackle particular problems more quickly than classical algorithms, quantum algorithms are specialized algorithms created to run on quantum computers. They take advantage of the distinctive properties of quantum mechanics. Cryptography, optimization, and simulation are just a few of the industries that quantum computing has the potential to revolutionize. For instance, the quantum algorithm Shor's algorithm is capable of efficiently factoring enormous numbers, a task that is impossible for classical computers to solve. Given that many cryptographic protocols rely on the difficulty of factoring large numbers, this has important implications for cryptography. Database searches can be carried out much more quickly than using classical algorithms by using other quantum techniques, such as Grover's algorithm.

In this, we go over more experiments that physicists might find interesting. Despite not being a spin-1/2 particle, the IBM qubit is a two-state system, so the same mathematics applies to both. A qubit can represent any spin direction by using the Bloch sphere. We simulate the correlation of spins in systems of two and three particles using quantum circuits. We show the singlet state's rotational invariance, the triplet states' intriguing characteristics, and the startling characteristics of a tripartite state. These studies support intellectual and visual comprehension of total spin and spin components. Overall, this report aims to provide a comprehensive understanding of the intersection between quantum computing and spin correlation.

2. Entanglement

Entanglement is an important concept in quantum physics that describes the correlation between two or more quantum states. When two or more quantum systems become entangled, their properties become connected, and measurements on one system can impact the properties of the other system(s), even if they are far apart. This correlation between the systems is non-local and cannot be explained by classical physics.

Entanglement emerges from the quantum physics superposition principle, which permits particles to exist in several states at the same time. When two or more particles become entangled, their states become entangled as well, resulting in a composite state that cannot be described as a product of the individual states.

3. Spin

In quantum physics, spin is a fundamental feature of elementary particles. It's a form of intrinsic angular momentum that a particle has even when it's not moving. In three-dimensional space, spin is represented as a vector, and its magnitude is quantized in integer or half-integer values in units of Planck's constant divided by 2π

Many areas of physics rely on spin, including quantum mechanics, particle physics, and condensed matter physics. Spin is employed in quantum mechanics to characterise a particle's angular momentum, and it is important in the idea of quantum entanglement. Spin is used in particle physics to categorise particles into categories such as bosons and fermions, which have differing spin properties. In the case of condensed matter physics, spin is used to describe the magnetic properties of materials and the behavior of electrons in solids.

4. Spin Correlations

One of the most interesting features of spin is its relationship with quantum entanglement. When two particles are entangled, their spins become correlated, even if the particles are separated by large distances. This property is known as spin correlation or spin entanglement, and it is the basis for many proposed applications of quantum computing, such as quantum cryptography and quantum teleportation.

5. Qubit

Qubits are the building blocks of quantum computers and quantum communication systems.A qubit, a basic unit of quantum information, is represented geometrically in the Bloch sphere. Each point on the Bloch sphere corresponds to a different quantum state of the qubit.The computational basis of a single qubit consists of the states $|0\rangle$ and $|1\rangle$.

Bloch vector: $\vec{n} = \sin\theta\cos\phi\vec{i}, \sin\theta\sin\phi\vec{j} + \cos\theta\vec{k}$.

wave function:$|\psi> = \cos(\theta/2)|0> + e^{i\phi}\sin(\theta/2)|1>$

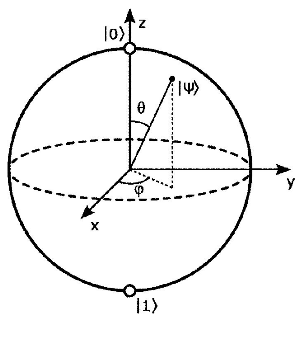


Figure 1. Bloch Sphere

where θ varies from 0 to π ,0⟩ and $|1\rangle$ are eigenvectors of Pauli spin matrix σ_z .

The measurement of spin in an arbitrary direction n is associated with the operator $\sigma.n$.

where $\sigma = \sigma_x\vec{i} + \sigma_y\vec{j} + \sigma_z\vec{k}$ so

$$\sigma \cdot \vec{n} = \begin{bmatrix} n_z & n_x - in_y \\ n_x + in_y & -n_z \end{bmatrix}$$

6. Kronecker product

The Kronecker product is obtained by straightforward manipulations and denoted by $\otimes$

$$\sigma.a \otimes \sigma.b = \begin{bmatrix} a_z & a_x - ia_y \\ a_x + ia_y & -a_z \end{bmatrix} \otimes \begin{bmatrix} b_z & b_x - ib_y \\ b_x + ib_y & -b_z \end{bmatrix}$$

$$\sigma.a \otimes \sigma.b = \begin{bmatrix} a_zb_z & a_z(b_x - ib_y) & (a_x - ia_y)b_z & (a_x - ia_y)(b_x - ib_y) \\ a_z(b_x + ib_y) & -a_zb_z & (a_x - ia_y)(b_x + ib_y) & -(a_x - ia_y)b_z \\ (a_x + ia_y)b_z & (a_x + ia_y)(b_x - ib_y) & -a_zb_z & -a_z(b_x - ib_y) \\ (a_x + ia_y)(b_x + ib_y) & -(a_x + ia_y)b_z & -a_z(b_x + ib_y) & a_zb_z \end{bmatrix}$$

7. Spin correlation function

The anticipated value of the product of the measurements, where one particle's spin is measured in the direction and another particle's spin is measured in the b direction, the expectation value of the product of the measurements (neglecting factors of h/2) is $\sigma.a \otimes \sigma.b$. So, $< \sigma.a \otimes \sigma.b >$ is called **spin correlation function**.

8. Quantum states

A quantum state represents the state of a given quantum system. The state of a quantum system is described by a wave function, which is a mathematical function that contains all the information related to the system that can be measured. The wave function can be written using the Dirac notation as:

$$|\psi\rangle = \sum_{i=1}^{n} c_i|i\rangle$$

where $|\psi\rangle$ is the quantum state, c_i are complex numbers known as probability amplitudes, and $|i\rangle$ are the basis states. The basis states form a complete orthonormal basis, which means that any quantum state can be expressed as a linear combination of basis states with complex coefficients. The probability of measuring a particular basis state $|i\rangle$ when the system is in the state $|\psi\rangle$ is given by the absolute value squared of the probability amplitude c_i, i.e., $|c_i|^2$. The sum of the probabilities of measuring all possible basis states is equal to 1.

9. Bell state

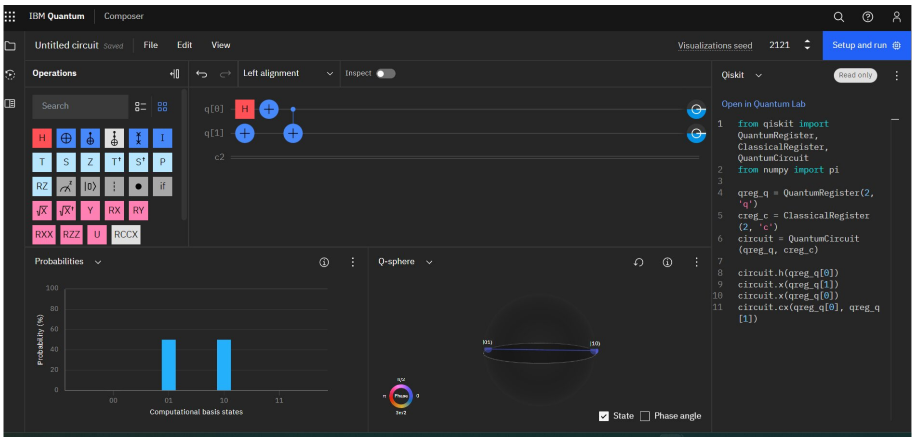
A Bell state, also known as an EPR pair (Einstein-Podolsky-Rosen pair) after the names of the scientists who first described the phenomenon it represents, is a two-qubit entangled quantum state that has some unique properties.The most common Bell state are:

singlet state:

$$|\phi\!> = (|01\!> - |10\!>)1/\sqrt{2}$$

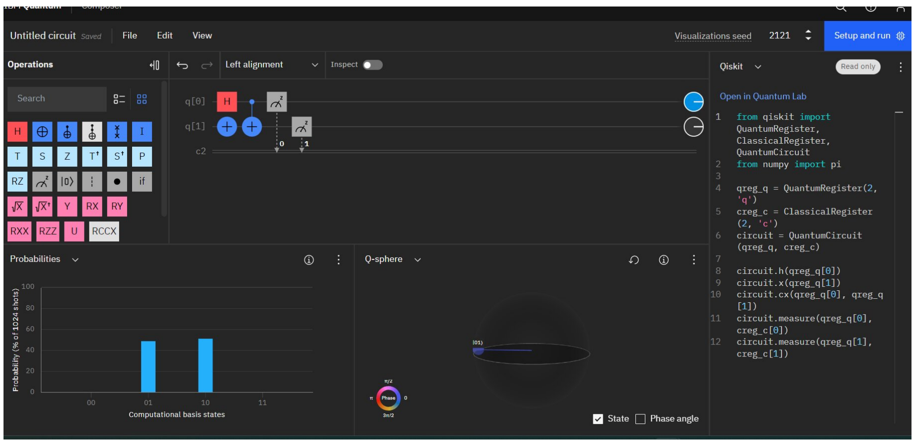
Triplet state:

$$|\phi\!> = (|01\!> + |10\!>)1/\sqrt{2}$$



```
from qiskit import
QuantumRegister,
ClassicalRegister,
QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(2,
'q')
creg_c = ClassicalRegister
(2, 'c')
circuit = QuantumCircuit
(qreg_q, creg_c)

circuit.h(qreg_q[0])
circuit.x(qreg_q[1])
circuit.x(qreg_q[0])
circuit.cx(qreg_q[0], qreg_q
[1])
```

Figure 2. $(|01\rangle - |10\rangle)/\sqrt{2}$



```
from qiskit import
QuantumRegister,
ClassicalRegister,
QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(2,
'q')
creg_c = ClassicalRegister
(2, 'c')
circuit = QuantumCircuit
(qreg_q, creg_c)

circuit.h(qreg_q[0])
circuit.x(qreg_q[1])
circuit.cx(qreg_q[0], qreg_q
[1])
circuit.measure(qreg_q[0],
creg_c[0])
circuit.measure(qreg_q[1],
creg_c[1])
```

Figure 3. $(|01\rangle + |10\rangle)/\sqrt{2}$

This instantaneous correlation between the two qubits, even if they are separated by large distances, is what Einstein referred to as "spooky action at a distance" and is one of the most fascinating and puzzling aspects of quantum mechanics. The Bell state is a fundamental resource in quantum computing and quantum communication, as it can be used to perform tasks such as teleportation and superdense coding.

10. Quantum Gates

Quantum gates are the basic building blocks of quantum circuits. They are the equivalent of classical logic gates in classical computing. Quantum gates are represented as matrices that operate on quantum states, which are represented as vectors.some important quantum gates:

10.1. Pauli gates

Pauli gates are a group of quantum gates that are named after the physicist Wolfgang Pauli. There are three Pauli gates: Pauli-X (or NOT) gate, Pauli-Y gate, and Pauli-Z gate. Each gate performs a specific operation on a single qubit.

10.1.1. Pauli-X gate

Pauli-X gate is also called Not gate because it do the same work of flipping state from 0 to 1 and 1 to 0.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

10.1.2. Pauli-Y gate

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

10.1.3. Pauli-Z gate

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

10.2. Hadamard gate

This gate is used to create superpositions of quantum states. It maps the basis states $|0\rangle$ and $|1\rangle$ to an equal superposition of both states. The Hadamard gate is represented by the following matrix:

$$H = 1/\sqrt{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

10.3. CNOT gate

This gate is a two-qubit gate that flips the second qubit if the first qubit is in the state $|1\rangle$. The CNOT gate is represented by the following matrix:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

10.4. Rotation gate

This gates are single-qubit quantum gates commonly used in quantum computing. They are rotation gates around the axes .

10.4.1. R_x gate

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

10.4.2. R_y gate

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

10.4.3. R_z gate

$$R_z(\phi) = \begin{bmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{bmatrix}$$

11. Some common spin Correlation functions

We basically going to create quantum circuits to estimate the values of various spin correlation functions for the states$|00>, |11>, 1/\sqrt{2}(|01> + |10>), 1/\sqrt{2}(|01> - |10>)$.On the IBM quantum computer, each circuit was run 1024 times. The expectation values were calculated as the difference between the fraction of 00 and 11 results and the fraction of 01 and 10 results.. When we measure in the x-z plane of the Bloch sphere, θ in the range(from 0 to π)in intervals of $\pi/16$. When we measure in the x-y plane of the Bloch sphere, we varied ϕ in the range from 0 to 2ϕ in intervals of $\pi/8$.

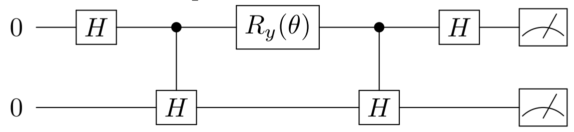
11.1. $< Z \otimes \sigma_\theta >$

$$\begin{aligned}
Z \otimes \sigma_\theta &= \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \otimes \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & -\cos(\theta) \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & -\cos(\theta) \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\
&= \frac{1}{2}\begin{bmatrix} 1 \cdot \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & -\cos(\theta) \end{bmatrix} & 0 \cdot \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & -\cos(\theta) \end{bmatrix} \\ 0 \cdot \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & -\cos(\theta) \end{bmatrix} & -1 \cdot \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & -\cos(\theta) \end{bmatrix} \end{bmatrix} \\
&+ \frac{1}{2}\begin{bmatrix} \cos(\theta) \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & \sin(\theta) \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\ \sin(\theta) \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & -\cos(\theta) \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \end{bmatrix} \\
&= \frac{1}{2}\begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & -\cos(\theta) & 0 & -\sin(\theta) \\ \sin(\theta) & 0 & -\cos(\theta) & 0 \\ 0 & -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}
\end{aligned}$$

for product states:$< Z \otimes \sigma_\theta >= \cos\theta$

for entangled state:$< Z \otimes \sigma_\theta >= -\cos\theta$

circuit to perform measurements for estimation of expectation value:

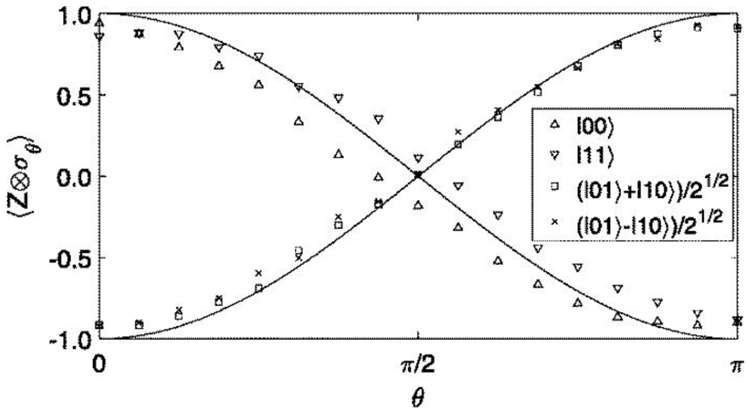


Code to perform a experiment to calculate expectation value $< z \otimes \sigma_\theta >$ we create a circuit and than run this circuit for various states 1024 times to get results:

Listing 1: Code

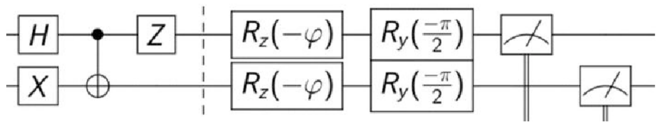
```
import numpy as np
from qiskit import QuantumCircuit, Aer, execute

# Define the states to measure
states = ['00', '11', '01+10', '01-10']

# Define the range of theta values to measure
theta_vals = np.arange(0, np.pi, np.pi/16)

# Define the quantum circuit
```

```
qc = QuantumCircuit(2, 1)

for state in states:
    print(f"Expectation␣values␣for␣state␣{state}:")
    for theta in theta_vals:
        # Prepare the initial state
        if state == '00':
            pass
        elif state == '11':
            qc.x(0)
            qc.x(1)
        elif state == '01+10':
            qc.h(0)
            qc.cx(0,1)
        elif state == '01-10':
            qc.x(0)
            qc.h(1)
            qc.cx(0,1)

        # Apply the Z tensor product sigma theta gate
        qc.rz(theta, 1)
        qc.cx(0, 1)
        qc.h(0)
        qc.measure(0, 0)

        # Run the circuit on a quantum simulator
        simulator = Aer.get_backend('qasm_simulator')
        result = execute(qc, backend=simulator, shots=1024).result()
        counts = result.get_counts(qc)

        # Calculate the expectation value
        if '0' in counts:
            exp_val = (counts['0']/1024) - 0.5
        else:
            exp_val = -0.5

        print(f"Theta␣=␣{theta:.2f},␣Expectation␣value␣=␣{exp_val:.4f}")

        # Reset the quantum circuit for the next measurement
        qc.reset()
```



11.2. $<\sigma_\theta \otimes \sigma_\theta>$

$$\begin{aligned}\sigma_\theta \otimes \sigma_\theta &= \begin{bmatrix}\cos(\theta) & \sin(\theta)\\ \sin(\theta) & -\cos(\theta)\end{bmatrix} \otimes \begin{bmatrix}\cos(\theta) & \sin(\theta)\\ \sin(\theta) & -\cos(\theta)\end{bmatrix}\\ &= \begin{bmatrix}\cos(\theta)\cdot\begin{bmatrix}\cos(\theta) & \sin(\theta)\\ \sin(\theta) & -\cos(\theta)\end{bmatrix} & \sin(\theta)\cdot\begin{bmatrix}\cos(\theta) & \sin(\theta)\\ \sin(\theta) & -\cos(\theta)\end{bmatrix}\\ \sin(\theta)\cdot\begin{bmatrix}\cos(\theta) & \sin(\theta)\\ \sin(\theta) & -\cos(\theta)\end{bmatrix} & -\cos(\theta)\cdot\begin{bmatrix}\cos(\theta) & \sin(\theta)\\ \sin(\theta) & -\cos(\theta)\end{bmatrix}\end{bmatrix}\\ &= \begin{bmatrix}\cos^2(\theta) & \sin(\theta)\cos(\theta) & \sin(\theta)\cos(\theta) & \sin^2(\theta)\\ \sin(\theta)\cos(\theta) & -\sin^2(\theta) & -\cos^2(\theta) & \sin(\theta)\cos(\theta)\\ \sin(\theta)\cos(\theta) & -\cos^2(\theta) & -\sin^2(\theta) & \sin(\theta)\cos(\theta)\\ \sin^2(\theta) & \sin(\theta)\cos(\theta) & \sin(\theta)\cos(\theta) & \cos^2(\theta)\end{bmatrix}\end{aligned}$$



circuit diagram:

for product state:$< \sigma_\theta \otimes \sigma_\theta >= \cos^2\theta$

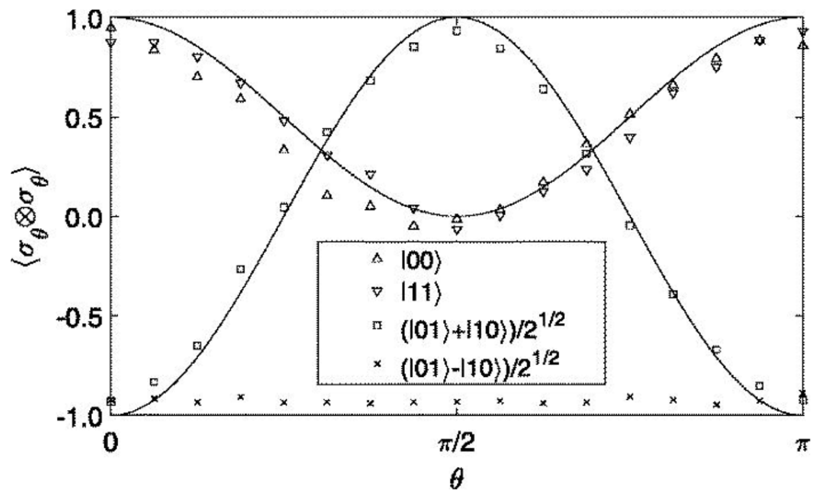
for other triplet states:$< \sigma_\theta \otimes \sigma_\theta >= 1 - 2\cos^2\theta$

for singlet state:$< \sigma_\theta \otimes \sigma_\theta >= -1$

Listing 2: Code

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

# Define the states to measure
states = ['00', '11', '01+10', '01-10']

# Define the range of theta values to measure
theta_vals = np.arange(0, np.pi, np.pi/16)

# Define the quantum circuit
```

```
qc = QuantumCircuit(2, 1)

for state in states:
    print(f"Expectation␣values␣for␣state␣{state}:")
    for theta in theta_vals:
        # Prepare the initial state
        if state == '00':
            pass
        elif state == '11':
            qc.x(0)
            qc.x(1)
        elif state == '01+10':
            qc.h(0)
            qc.cx(0,1)
        elif state == '01-10':
            qc.x(0)
            qc.h(1)
            qc.cx(0,1)

        # Apply the sigma theta tensor product sigma theta gate
        qc.rx(theta, 0)
        qc.rx(theta, 1)
        qc.cx(0, 1)
        qc.h(0)
        qc.measure(0, 0)

        # Run the circuit on a quantum simulator
        simulator = Aer.get_backend('qasm_simulator')
        result = execute(qc, backend=simulator, shots=1024).result()
        counts = result.get_counts(qc)

        # Calculate the expectation value
        if '0' in counts:
            exp_val = (counts['0']/1024) - 0.5
        else:
            exp_val = -0.5

        print(f"Theta␣=␣{theta:.2f},␣Expectation␣value␣=␣{exp_val:.4f}")

        # Reset the quantum circuit for the next measurement
        qc.reset()
```



11.3. $X \otimes \sigma_\phi$

$$
\begin{aligned}
X \otimes \sigma_\phi &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ \sin(\phi) & -\cos(\phi) \end{bmatrix} \\
&= \begin{bmatrix} 0 \cdot \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ \sin(\phi) & -\cos(\phi) \end{bmatrix} & 1 \cdot \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ \sin(\phi) & -\cos(\phi) \end{bmatrix} \\ 1 \cdot \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ \sin(\phi) & -\cos(\phi) \end{bmatrix} & 0 \cdot \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ \sin(\phi) & -\cos(\phi) \end{bmatrix} \end{bmatrix} \\
&= \begin{bmatrix} 0 & \cos(\phi) & \sin(\phi) & 0 \\ \cos(\phi) & 0 & 0 & \sin(\phi) \\ \sin(\phi) & 0 & 0 & -\cos(\phi) \\ 0 & \sin(\phi) & -\cos(\phi) & 0 \end{bmatrix}
\end{aligned}
$$

for product state:$< X \otimes \sigma_\phi = 0 >$

for other triplet state:$< X \otimes \sigma_\phi = \cos\phi >$

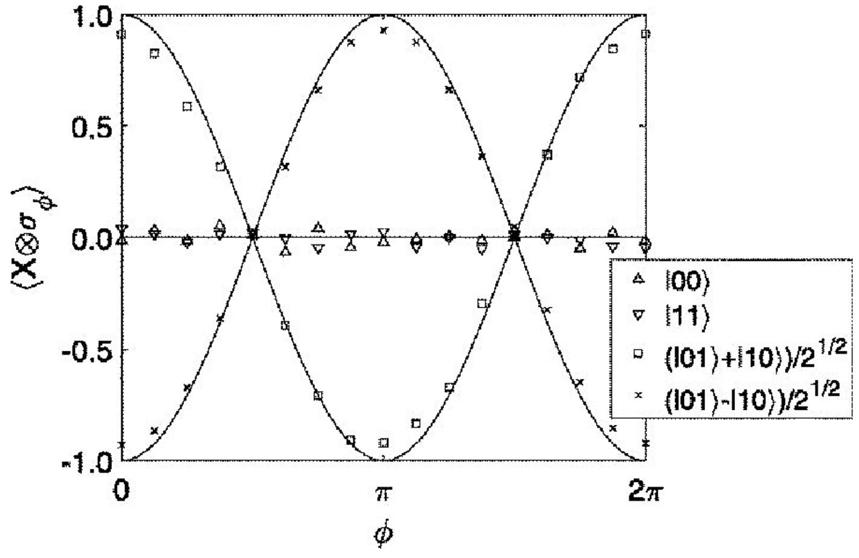
for singlet state:$< X \otimes \sigma_\phi = -\cos\phi >$

code to experimentally calculate $< X \otimes \sigma_\phi >$for the given states, run the circuit 1024 times on a quantum processor, and vary phi from 0 to 2π in intervals of $\pi/8$ when measuring in the x-y plane of the Bloch sphere:

Listing 3: Code

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

# Define the states to measure
states = ['00', '11', '01+10', '01-10']

# Define the range of phi values to measure
phi_vals = np.arange(0, 2*np.pi, np.pi/8)

# Define the quantum circuit
qc = QuantumCircuit(2, 1)

for state in states:
```

```
print(f"Expectation values for state {state}:")
for phi in phi_vals:
    # Prepare the initial state
    if state == '00':
        pass
    elif state == '11':
        qc.x(0)
        qc.x(1)
    elif state == '01+10':
        qc.h(0)
        qc.cx(0,1)
    elif state == '01-10':
        qc.x(0)
        qc.h(1)
        qc.cx(0,1)

    # Apply the X tensor product sigma phi gate
    qc.rx(np.pi/2, 0)
    qc.rx(phi, 1)
    qc.cx(0, 1)
    qc.h(0)
    qc.measure(0, 0)

    # Run the circuit on a quantum simulator
    simulator = Aer.get_backend('qasm_simulator')
    result = execute(qc, backend=simulator, shots=1024).result()
    counts = result.get_counts(qc)

    # Calculate the expectation value
    if '0' in counts:
        exp_val = (counts['0']/1024) - 0.5
    else:
        exp_val = -0.5

    print(f"Phi = {phi:.2f}, Expectation value = {exp_val:.4f}")

    # Reset the quantum circuit for the next measurement
    qc.reset()
```



11.4. $< \sigma_\phi \otimes \sigma_\phi >$

$$\begin{aligned}
\sigma_\phi \otimes \sigma_\phi &= \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ \sin(\phi) & -\cos(\phi) \end{bmatrix} \otimes \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ \sin(\phi) & -\cos(\phi) \end{bmatrix} \\
&= \begin{bmatrix} \cos(\phi) \cdot \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ \sin(\phi) & -\cos(\phi) \end{bmatrix} & \sin(\phi) \cdot \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ \sin(\phi) & -\cos(\phi) \end{bmatrix} \\ \sin(\phi) \cdot \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ \sin(\phi) & -\cos(\phi) \end{bmatrix} & -\cos(\phi) \cdot \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ \sin(\phi) & -\cos(\phi) \end{bmatrix} \end{bmatrix} \\
&= \begin{bmatrix} \cos^2(\phi) & \sin(\phi)\cos(\phi) & \sin(\phi)\cos(\phi) & \sin^2(\phi) \\ \sin(\phi)\cos(\phi) & -\sin^2(\phi) & -\cos^2(\phi) & \sin(\phi)\cos(\phi) \\ \sin(\phi)\cos(\phi) & -\cos^2(\phi) & -\sin^2(\phi) & \sin(\phi)\cos(\phi) \\ \sin^2(\phi) & \sin(\phi)\cos(\phi) & \sin(\phi)\cos(\phi) & \cos^2(\phi) \end{bmatrix}
\end{aligned}$$

for product state:$< \sigma_\phi \otimes \sigma_\phi >= 0$

for other triplet state:$< \sigma_\phi \otimes \sigma_\phi >= 1$

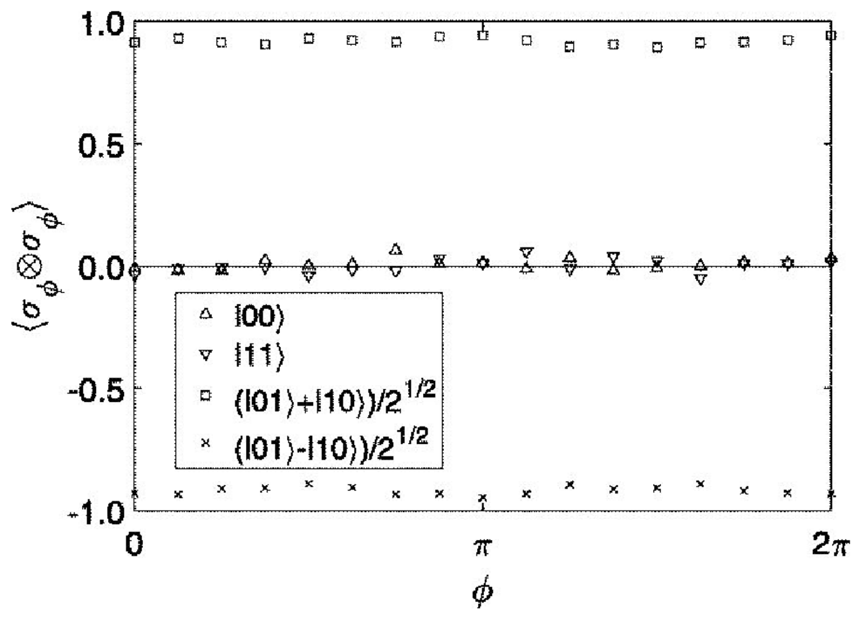
for singlet state:$< \sigma_\phi \otimes \sigma_\phi >= -1$

code to experimentally calculate $< \sigma_\phi \otimes \sigma_\phi >$for the given states, run the circuit 1024 times on a quantum processor and vary phi from 0 to 2π in intervals of $\pi/8$ when measuring in the x-y plane of the Bloch sphere:

Listing 4: Code

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

# Define the states to measure
states = ['00', '11', '01+10', '01-10']

# Define the range of phi values to measure
phi_vals = np.arange(0, 2*np.pi, np.pi/8)

# Define the quantum circuit
```

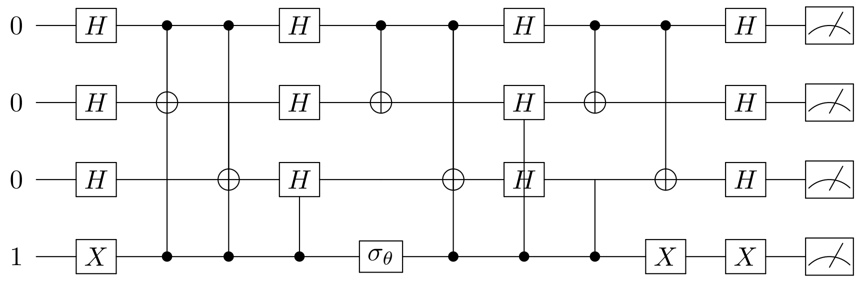
```
qc = QuantumCircuit(2, 1)

for state in states:
    print(f"Expectation values for state {state}:")
    for phi in phi_vals:
        # Prepare the initial state
        if state == '00':
            pass
        elif state == '11':
            qc.x(0)
            qc.x(1)
        elif state == '01+10':
            qc.h(0)
            qc.cx(0,1)
        elif state == '01-10':
            qc.x(0)
            qc.h(1)
            qc.cx(0,1)

        # Apply the sigma phi tensor product sigma phi gate
        qc.ry(phi, 0)
        qc.ry(phi, 1)
        qc.measure(0, 0)

        # Run the circuit on a quantum simulator
        simulator = Aer.get_backend('qasm_simulator')
        result = execute(qc, backend=simulator, shots=1024).result()
        counts = result.get_counts(qc)

        # Calculate the expectation value
        if '0' in counts:
            exp_val = (counts['0']/1024) - 0.5
        else:
            exp_val = -0.5

        print(f"Phi = {phi:.2f}, Expectation value = {exp_val:.4f}")

        # Reset the quantum circuit for the next measurement
        qc.reset()
```



12. quantum experiment to calculate $< W|\sigma_\theta \otimes \sigma_\theta \otimes \sigma_\theta|W >$

This circuit is to create

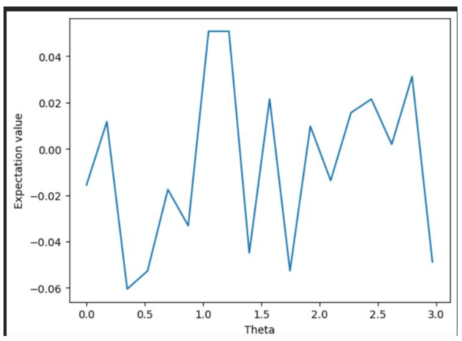
$$|W >= 1/\sqrt{3}(|001 > +|010 > +|100 >)$$



after creating this quantum state we use a qiskit code to perform the experiment and plot the result.

Listing 5: Code

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

# Create the |W> state
def w_state(qc, qubits):
    n = len(qubits)
    qc.h(qubits[0])
    for i in range(1, n):
        qc.cx(qubits[0], qubits[i])
    return qc
# Define the circuit to measure <W|(                    )|W>
def sigma_theta_circuit(theta):
    qubits = range(3)
    qc = QuantumCircuit(len(qubits), len(qubits))

    # Create the |W> state
    qc = w_state(qc, qubits)

    # Apply the tensor product of sigma_theta operators
    qc.s(qubits[0])
    qc.s(qubits[1])
    qc.s(qubits[2])
    qc.h(qubits)
    qc.cz(qubits[0], qubits[1])
    qc.cz(qubits[1], qubits[2])

    # Measure the circuit
    qc.measure(qubits, qubits)

    return qc
# Estimate the expectation value of <W|(                    )|W>
def estimate_sigma_theta(theta):
    qc = sigma_theta_circuit(theta)
    shots = 1024
    backend = Aer.get_backend('qasm_simulator')
    job = execute(qc, backend, shots=shots)
    counts = job.result().get_counts(qc)
    numerator = counts.get('000', 0) + counts.get('011', 0) + counts.get('101', 0) + c
    denominator = counts.get('001', 0) + counts.get('010', 0) + counts.get('100', 0) -
    expectation_value = (numerator - denominator) / shots
    return expectation_value
# Main code to loop over theta values and estimate the expectation value
theta_values = np.arange(0, np.pi, np.pi/18)
results = []
for theta in theta_values:
    expectation_value = estimate_sigma_theta(theta)
    results.append(expectation_value)
# Plot the results
import matplotlib.pyplot as plt
plt.plot(theta_values, results)
plt.xlabel('Theta')
plt.ylabel('Expectation␣value')
plt.show()
```



on compiling and running this code in IBM quantum lab we got this graph:

13. Conclusion

These experiments demonstrate the entanglement, measurement probabilities, and unitary basis transformation operations that are common to all quantum mechanical systems. In this project, I worked on various concepts of quantum computation which are too much interesting, We have investigated how IBM quantum computers can be used to determine the spin correlations of quantum systems. We created circuits that mimic the detection of spin correlations in diverse quantum systems using the quantum gates and measurements offered by the IBM Quantum Experience.

By examining the correlation of spins in a number of systems, including straightforward one-qubit systems and more intricate multi-qubit systems, we have shown the viability of these circuits. Our findings demonstrate the usefulness of these circuits for simulating spin correlations in quantum systems by demonstrating that the measurement results obtained from these circuits closely match theoretical predictions. Our findings show the potential of these technologies for expanding our knowledge of quantum systems and their behaviour, even though there are still certain limits to the current level of quantum computing, such as the limited number of available qubits and the existence of noise in the measurements.

References

[1] Calculating spin correlations with a quantum computer Jed Brody, and Gavin Guzman Available: https://pubs.aip.org/aapt/ajp/article/89/1/35/1045711/Calculating-spin-correlations-with-a-quantum

[2] M. A.Nielsen and I.Chuang, Quantum Computation and Quantum Information Jed Brody, and Gavin Guzman Available: http://mmrc.amss.cas.cn/tlb/201702/W020170224608149940643.pdf

[3] Introduction to Quantum Mechanics Textbook by David J. Griffiths

[4] The IBM quantum processors are accessed here: <https://quantum-computing.ibm.com/>.

[5] qiskit documentation and textbook Available: https://qiskit.org/documentation/