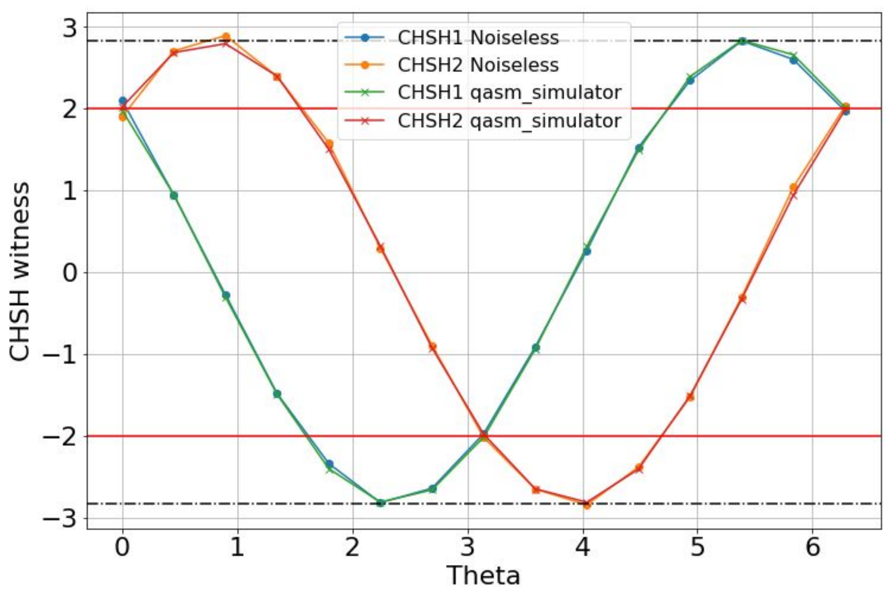
Code

Michelle Ding and Letizia Fazzini

April 28, 2022

Listings

1 Experiment 1




```python
#import qiskit tools
import qiskit
from qiskit import QuantumCircuit, ClassicalRegister,
    QuantumRegister, transpile, Aer, IBMQ
from qiskit.tools.visualization import circuit_drawer
from qiskit.tools.monitor import job_monitor, backend_monitor,
    backend_overview
```

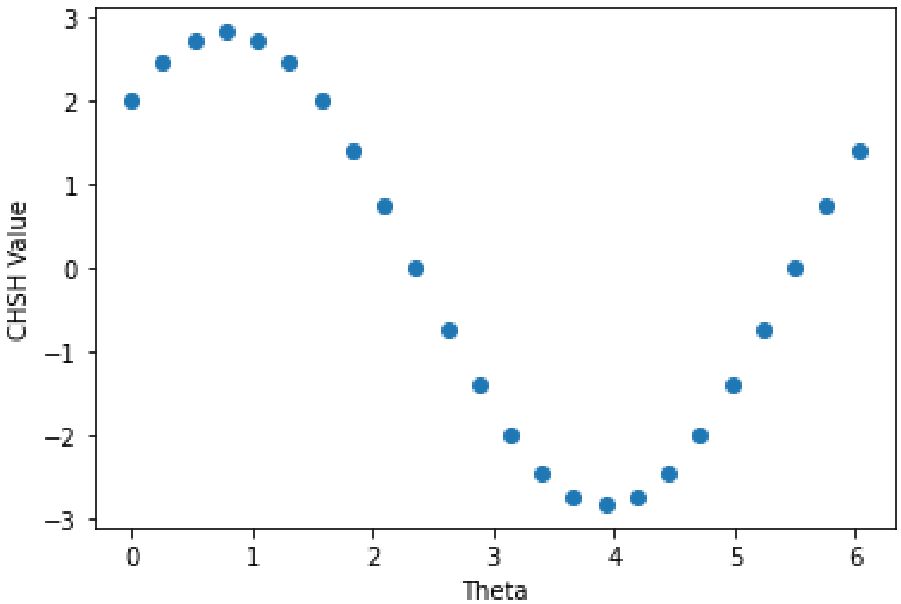
```python
from qiskit.providers.aer import noise

#import python stuff
import matplotlib.pyplot as plt
import numpy as np
import time

# Set devices
IBMQ.load_account()
provider = IBMQ.get_provider('ibm-q')
qasm_simulator = provider.get_backend('ibmq_qasm_simulator')

sim = Aer.get_backend('aer_simulator')

def make_chsh_circuit(theta_vec):
    """Return a list of QuantumCircuits"""
    chsh_circuits = []

    for theta in theta_vec:
        obs_vec = ['00', '01', '10', '11']
        for el in obs_vec:
            qc = QuantumCircuit(2,2)
            qc.h(0)
            qc.cx(0, 1)
            qc.ry(theta, 0)
            for a in range(2):
                if el[a] == '1':
                    qc.h(a)
            qc.measure(range(2),range(2))
            chsh_circuits.append(qc)

    return chsh_circuits

def compute_chsh_witness(counts):
    """Compute expectation values (ZZ,ZX,XZ,XX) """

    CHSH1 = []
    CHSH2 = []
    # Divide the list of dictionaries in sets of 4
    for i in range(0, len(counts), 4):
        theta_dict = counts[i:i + 4]
        zz = theta_dict[0]
        zx = theta_dict[1]
        xz = theta_dict[2]
        xx = theta_dict[3]

        no_shots = sum(xx[y] for y in xx)

        chsh1 = 0
        chsh2 = 0

        for element in zz:
            parity = (-1)**(int(element[0])+int(element[1]))
            chsh1+= parity*zz[element]
            chsh2+= parity*zz[element]

        for element in zx:
```

```
            parity = (-1)**(int(element[0])+int(element[1]))
            chsh1+= parity*zx[element]
            chsh2-= parity*zx[element]

        for element in xz:
            parity = (-1)**(int(element[0])+int(element[1]))
            chsh1-= parity*xz[element]
            chsh2+= parity*xz[element]

        for element in xx:
            parity = (-1)**(int(element[0])+int(element[1]))
            chsh1+= parity*xx[element]
            chsh2+= parity*xx[element]

        CHSH1.append(chsh1/no_shots)
        CHSH2.append(chsh2/no_shots)

    return CHSH1, CHSH2

number_of_thetas = 15
theta_vec = np.linspace(0,2*np.pi,number_of_thetas)
my_chsh_circuits = make_chsh_circuit(theta_vec)

my_chsh_circuits[4].draw()
my_chsh_circuits[5].draw()
my_chsh_circuits[6].draw()
my_chsh_circuits[7].draw()

# Execute and get counts
result_ideal = sim.run(my_chsh_circuits).result()

tic = time.time()
transpiled_circuits = transpile(my_chsh_circuits, qasm_simulator)
job_real = qasm_simulator.run(transpiled_circuits, shots=8192)
job_monitor(job_real)
result_real = job_real.result()
toc = time.time()

print(toc-tic)

# For display
CHSH1_ideal, CHSH2_ideal = compute_chsh_witness(result_ideal.get_counts())
CHSH1_real, CHSH2_real = compute_chsh_witness(result_real.get_counts())

plt.figure(figsize=(12,8))
plt.rcParams.update({'font.size': 22})
plt.plot(theta_vec,CHSH1_ideal,'o-',label = 'CHSH1 Noiseless')
plt.plot(theta_vec,CHSH2_ideal,'o-',label = 'CHSH2 Noiseless')

plt.plot(theta_vec,CHSH1_real,'x-',label = 'CHSH1 qasm_simulator')
plt.plot(theta_vec,CHSH2_real,'x-',label = 'CHSH2 qasm_simulator')

plt.grid(which='major',axis='both')
plt.rcParams.update({'font.size': 16})
plt.legend()
```

```
plt.axhline(y=2, color='r', linestyle='-')
plt.axhline(y=-2, color='r', linestyle='-')
plt.axhline(y=np.sqrt(2)*2, color='k', linestyle='-.')
plt.axhline(y=-np.sqrt(2)*2, color='k', linestyle='-.')
plt.xlabel('Theta')
plt.ylabel('CHSH witness')
```

Listing 1: CHSH IBM Quantum Experience *qasm* Simulation Results

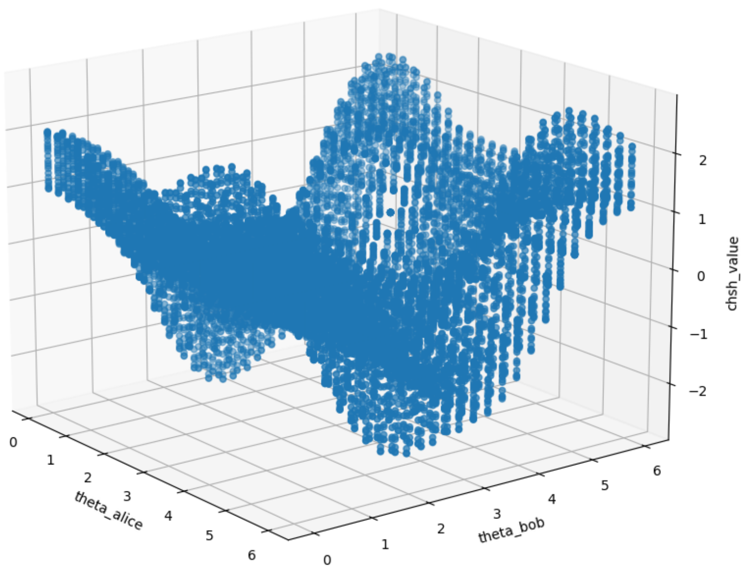
2 Experiment 2



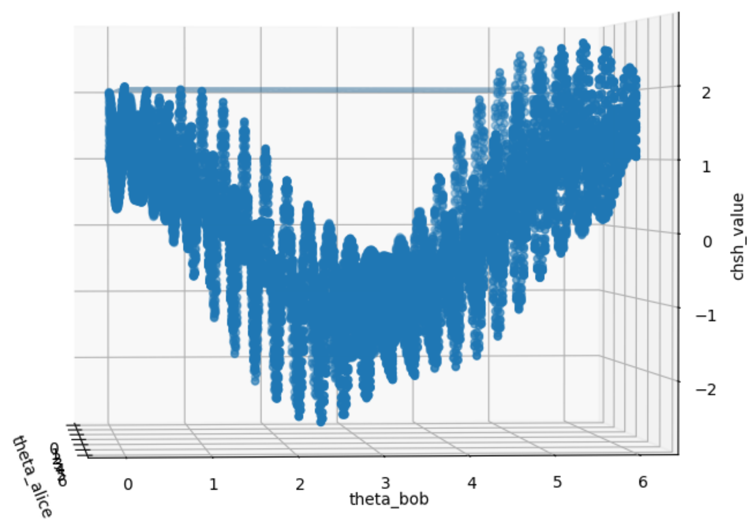

```
import numpy as np
import math
import matplotlib.pyplot as plt

x_axis = [0] * 24
y_axis = [0] * 24
i = 0

for theta in range(0,350,15):

  theta = math.radians(theta)
  # A = Z and a = X
  A = np.array([[1,0],[0,-1]])
  a = np.array([[0,1],[1,0]])

  rotation_arr = np.array([[math.cos(theta),-math.sin(theta)],[math
    .sin(theta),math.cos(theta)]])

  # B and b are rotations of A and a
  B = np.matmul(rotation_arr,A)
  b = np.matmul(rotation_arr,a)

  zero_state = np.array([[1],[0]])
  one_state = np.array([[0],[1]])
  phi_bra = np.array([1/math.sqrt(2),0,0,1/math.sqrt(2)])
  phi_ket = np.array([[1/math.sqrt(2)],[0],[0],[1/math.sqrt(2)]])

  matrix_AB = np.kron(A,B)

  result_AB = np.matmul(phi_bra,matrix_AB)
```

```
    result_AB = np.matmul(result_AB,phi_ket)

    matrix_Ab = np.kron(A,b)

    result_Ab = np.matmul(phi_bra,matrix_Ab)
    result_Ab = np.matmul(result_Ab,phi_ket)

    matrix_aB = np.kron(a,B)

    result_aB = np.matmul(phi_bra,matrix_aB)
    result_aB = np.matmul(result_aB,phi_ket)

    matrix_ab = np.kron(a,b)

    result_ab = np.matmul(phi_bra,matrix_ab)
    result_ab = np.matmul(result_ab,phi_ket)

    result = result_AB - result_Ab + result_aB + result_ab


    print(theta, result)

    x_axis[i] = theta
    y_axis[i] = result[0]

    i += 1

plt.scatter(x_axis,y_axis)
```

Listing 2: CHSH Theoretical Experiment for Fixed Basis Results



(a) front view



(b) side view

3 Experiment 3

```
import numpy as np
import math
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d


# for display
t1 = []
t2 = []
r = []

for phi1 in range(0,360,15):
    for theta1 in range(0,360,15):
        theta1 = math.radians(theta1)
        rotation_arr1 = np.array([[math.cos(theta1),-math.sin(theta1)],[math.sin(theta1),math.cos(theta1)]])
        general_unitary = np.array([[math.cos(theta1/2),-(math.e**(1j))*math.sin(theta1/2)],[(math.e**(phi1*1j))*math.sin(theta1/2),(math.e**(phi1*1j))*math.cos(theta1/2)]])

        # A and a are rotations of Z and X
        A = np.matmul(rotation_arr1,np.array([[1,0],[0,-1]]))
        a = general_unitary

        for theta2 in range(0,360,15):
            theta2 = math.radians(theta2)
            t1.append(theta1)
            t2.append(theta2)
```

```python
            rotation_arr2 = np.array([[math.cos(theta2),-math.sin(theta2)],[math.sin(theta2),math.cos(theta2)]])

            # B and b are rotations of A and a
            B = np.matmul(rotation_arr2,A)
            b = np.matmul(rotation_arr2,a)

            # calculate phi, the entangled wave function
            zero_state = np.array([[1],[0]])
            one_state = np.array([[0],[1]])
            phi_bra = np.array([1/math.sqrt(2),0,0,1/math.sqrt(2)])
            phi_ket = np.array([[1/math.sqrt(2)],[0],[0],[1/math.sqrt(2)]])

            # expected value of AB
            matrix_AB = np.kron(A,B)
            result_AB = np.matmul(phi_bra,matrix_AB)
            result_AB = np.matmul(result_AB,phi_ket)
            # expected value of Ab
            matrix_Ab = np.kron(A,b)
            result_Ab = np.matmul(phi_bra,matrix_Ab)
            result_Ab = np.matmul(result_Ab,phi_ket)
            # expected value of aB
            matrix_aB = np.kron(a,B)
            result_aB = np.matmul(phi_bra,matrix_aB)
            result_aB = np.matmul(result_aB,phi_ket)
            # expected value of ab
            matrix_ab = np.kron(a,b)
            result_ab = np.matmul(phi_bra,matrix_ab)
            result_ab = np.matmul(result_ab,phi_ket)

            # CHSH value (AB - Ab + aB + ab)
            result = result_AB - result_Ab + result_aB + result_ab
            r.append(result)


#    print(theta, result)
# for display
ax = plt.axes(projection='3d')

x = t1
y = t2
z = r

ax.set_xlabel("theta_alice")
ax.set_ylabel("theta_bob")
ax.set_zlabel("chsh_value")

ax.scatter(x, y, z)
plt.show()
```

Listing 3: CHSH Theoretical Experiment for General Basis Results