

Code

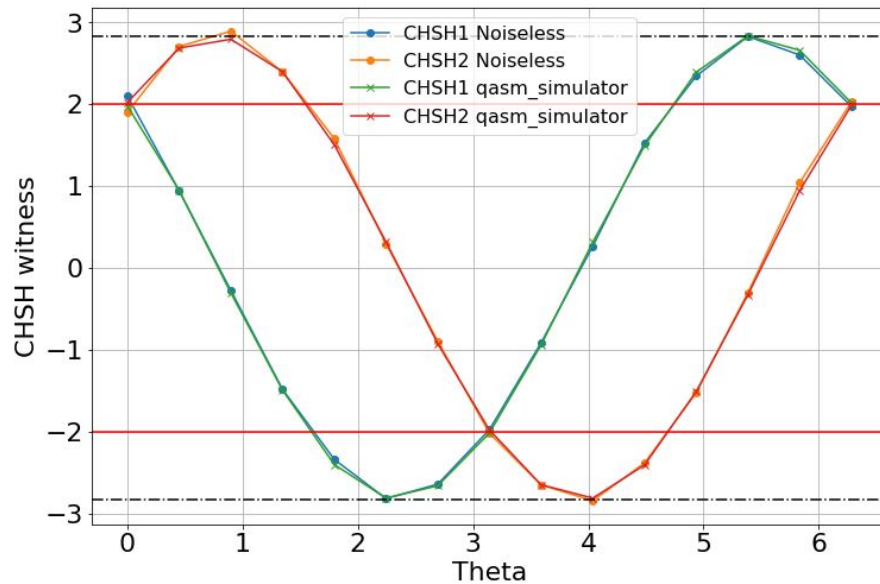
Michelle Ding and Letizia Fazzini

April 28, 2022

Listings

1	CHSH IBM Quantum Experience <i>qasm</i> Simulation Results . . .	1
2	CHSH Theoretical Experiment for Fixed Basis Results	5
3	CHSH Theoretical Experiment for General Basis Results	7

1 Experiment 1



```
1 #import qiskit tools
2 import qiskit
3 from qiskit import QuantumCircuit, ClassicalRegister,
  QuantumRegister, transpile, Aer, IBMQ
4 from qiskit.tools.visualization import circuit_drawer
5 from qiskit.tools.monitor import job_monitor, backend_monitor,
  backend_overview
```

```

6 from qiskit.providers.aer import noise
7
8 #import python stuff
9 import matplotlib.pyplot as plt
10 import numpy as np
11 import time
12
13 # Set devices
14 IBMQ.load_account()
15 provider = IBMQ.get_provider('ibm-q')
16 qasm_simulator = provider.get_backend('ibmq_qasm_simulator')
17
18 sim = Aer.get_backend('aer_simulator')
19
20 def make_chsh_circuit(theta_vec):
21     """Return a list of QuantumCircuits"""
22     chsh_circuits = []
23
24     for theta in theta_vec:
25         obs_vec = ['00', '01', '10', '11']
26         for el in obs_vec:
27             qc = QuantumCircuit(2,2)
28             qc.h(0)
29             qc.cx(0, 1)
30             qc.ry(theta, 0)
31             for a in range(2):
32                 if el[a] == '1':
33                     qc.h(a)
34             qc.measure(range(2),range(2))
35             chsh_circuits.append(qc)
36
37     return chsh_circuits
38
39 def compute_chsh_witness(counts):
40     """Compute expectation values (ZZ,ZX,XZ,XX) """
41
42     CHSH1 = []
43     CHSH2 = []
44     # Divide the list of dictionaries in sets of 4
45     for i in range(0, len(counts), 4):
46         theta_dict = counts[i:i + 4]
47         zz = theta_dict[0]
48         zx = theta_dict[1]
49         xz = theta_dict[2]
50         xx = theta_dict[3]
51
52         no_shots = sum(xx[y] for y in xx)
53
54         chsh1 = 0
55         chsh2 = 0
56
57         for element in zz:
58             parity = (-1)**(int(element[0])+int(element[1]))
59             chsh1+= parity*zz[element]
60             chsh2+= parity*zz[element]
61
62         for element in zx:

```

```

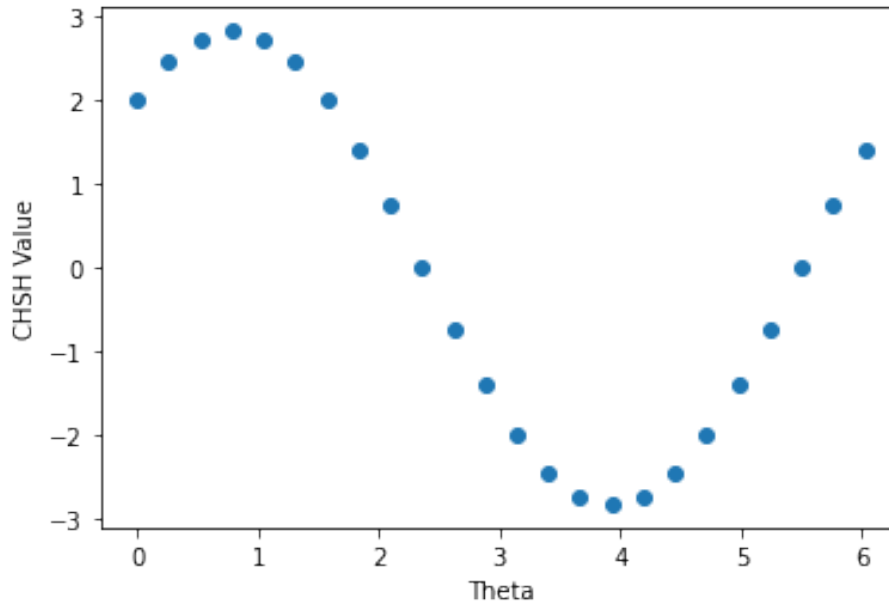
63     parity = (-1)**(int(element[0])+int(element[1]))
64     chsh1+= parity*zx[element]
65     chsh2-= parity*zx[element]
66
67     for element in xz:
68         parity = (-1)**(int(element[0])+int(element[1]))
69         chsh1-= parity*xz[element]
70         chsh2+= parity*xz[element]
71
72     for element in xx:
73         parity = (-1)**(int(element[0])+int(element[1]))
74         chsh1+= parity*xx[element]
75         chsh2+= parity*xx[element]
76
77     CHSH1.append(chsh1/no_shots)
78     CHSH2.append(chsh2/no_shots)
79
80     return CHSH1, CHSH2
81
82 number_of_thetas = 15
83 theta_vec = np.linspace(0,2*np.pi,number_of_thetas)
84 my_chsh_circuits = make_chsh_circuit(theta_vec)
85
86 my_chsh_circuits[4].draw()
87 my_chsh_circuits[5].draw()
88 my_chsh_circuits[6].draw()
89 my_chsh_circuits[7].draw()
90
91 # Execute and get counts
92 result_ideal = sim.run(my_chsh_circuits).result()
93
94 tic = time.time()
95 transpiled_circuits = transpile(my_chsh_circuits, qasm_simulator)
96 job_real = qasm_simulator.run(transpiled_circuits, shots=8192)
97 job_monitor(job_real)
98 result_real = job_real.result()
99 toc = time.time()
100
101 print(toc-tic)
102
103 # For display
104 CHSH1_ideal, CHSH2_ideal = compute_chsh_witness(result_ideal.
105     get_counts())
106 CHSH1_real, CHSH2_real = compute_chsh_witness(result_real.
107     get_counts())
108
109 plt.figure(figsize=(12,8))
110 plt.rcParams.update({'font.size': 22})
111 plt.plot(theta_vec,CHSH1_ideal,'o-',label = 'CHSH1 Noiseless')
112 plt.plot(theta_vec,CHSH2_ideal,'o-',label = 'CHSH2 Noiseless')
113
114 plt.plot(theta_vec,CHSH1_real,'x-',label = 'CHSH1 qasm_simulator')
115 plt.plot(theta_vec,CHSH2_real,'x-',label = 'CHSH2 qasm_simulator')
116
117 plt.grid(which='major',axis='both')
118 plt.rcParams.update({'font.size': 16})
119 plt.legend()

```

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

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

2 Experiment 2



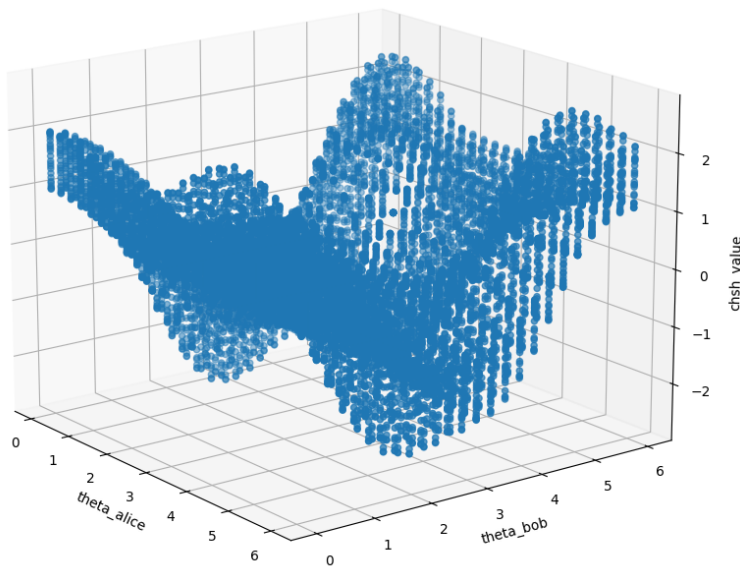
```
1 import numpy as np
2 import math
3 import matplotlib.pyplot as plt
4
5 x_axis = [0] * 24
6 y_axis = [0] * 24
7 i = 0
8
9 for theta in range(0,350,15):
10
11     theta = math.radians(theta)
12     # A = Z and a = X
13     A = np.array([[1,0],[0,-1]])
14     a = np.array([[0,1],[1,0]])
15
16     rotation_arr = np.array([[math.cos(theta),-math.sin(theta)],[math
17         .sin(theta),math.cos(theta)]]
18
19     # B and b are rotations of A and a
20     B = np.matmul(rotation_arr,A)
21     b = np.matmul(rotation_arr,a)
22
23     zero_state = np.array([[1],[0]])
24     one_state = np.array([[0],[1]])
25     phi_bra = np.array([1/math.sqrt(2),0,0,1/math.sqrt(2)])
26     phi_ket = np.array([1/math.sqrt(2)],[0],[0],[1/math.sqrt(2)])
27
28     matrix_AB = np.kron(A,B)
29     result_AB = np.matmul(phi_bra,matrix_AB)
```

```

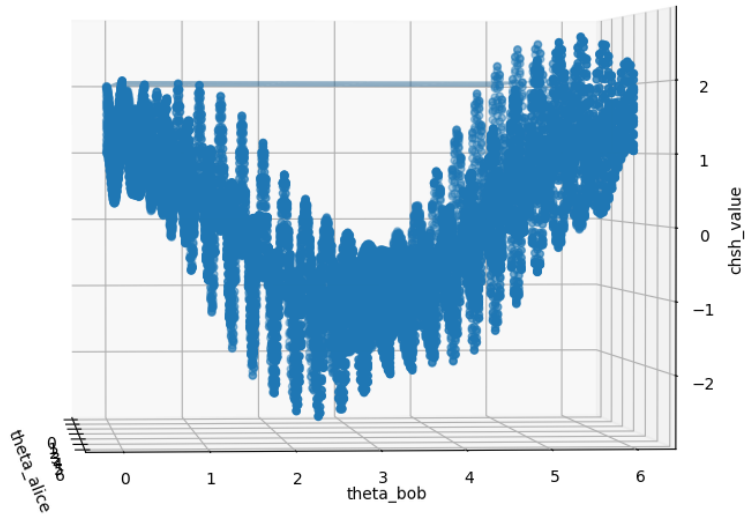
30 result_AB = np.matmul(result_AB, phi_ket)
31
32 matrix_Ab = np.kron(A, b)
33
34 result_Ab = np.matmul(phi_bra, matrix_Ab)
35 result_Ab = np.matmul(result_Ab, phi_ket)
36
37 matrix_aB = np.kron(a, B)
38
39 result_aB = np.matmul(phi_bra, matrix_aB)
40 result_aB = np.matmul(result_aB, phi_ket)
41
42 matrix_ab = np.kron(a, b)
43
44 result_ab = np.matmul(phi_bra, matrix_ab)
45 result_ab = np.matmul(result_ab, phi_ket)
46
47 result = result_AB - result_Ab + result_aB + result_ab
48
49
50 print(theta, result)
51
52 x_axis[i] = theta
53 y_axis[i] = result[0]
54
55 i += 1
56
57 plt.scatter(x_axis, y_axis)

```

Listing 2: CHSH Theoretical Experiment for Fixed Basis Results



(a) front view



(b) side view

3 Experiment 3

```

1 import numpy as np
2 import math
3 import matplotlib.pyplot as plt
4 from mpl_toolkits import mplot3d
5
6
7 # for display
8 t1 = []
9 t2 = []
10 r = []
11
12 for phi1 in range(0,360,15):
13     for theta1 in range(0,360,15):
14         theta1 = math.radians(theta1)
15         rotation_arr1 = np.array([[math.cos(theta1), -math.sin(
16             theta1)], [math.sin(theta1), math.cos(theta1)]])
17         general_unitary = np.array([[math.cos(theta1/2), -(math.e
18             *(1j))*math.sin(theta1/2)], [(math.e**(phi1*1j))*math.sin(
19             theta1/2), (math.e**(phi1*1j))*math.cos(theta1/2)])]
20
21         # A and a are rotations of Z and X
22         A = np.matmul(rotation_arr1, np.array([[1, 0], [0, -1]]))
23         a = general_unitary
24
25     for theta2 in range(0,360,15):
26         theta2 = math.radians(theta2)
27         t1.append(theta1)
28         t2.append(theta2)

```

```

26 rotation_arr2 = np.array([[math.cos(theta2), -math.sin(
theta2)], [math.sin(theta2), math.cos(theta2)]])
27
28 # B and b are rotations of A and a
29 B = np.matmul(rotation_arr2, A)
30 b = np.matmul(rotation_arr2, a)
31
32 # calculate phi, the entangled wave function
33 zero_state = np.array([[1], [0]])
34 one_state = np.array([[0], [1]])
35 phi_bra = np.array([1/math.sqrt(2), 0, 0, 1/math.sqrt(2)])
36 phi_ket = np.array([[1/math.sqrt(2)], [0], [0], [1/math.
sqrt(2)]])
37
38 # expected value of AB
39 matrix_AB = np.kron(A, B)
40 result_AB = np.matmul(phi_bra, matrix_AB)
41 result_AB = np.matmul(result_AB, phi_ket)
42 # expected value of Ab
43 matrix_Ab = np.kron(A, b)
44 result_Ab = np.matmul(phi_bra, matrix_Ab)
45 result_Ab = np.matmul(result_Ab, phi_ket)
46 # expected value of aB
47 matrix_aB = np.kron(a, B)
48 result_aB = np.matmul(phi_bra, matrix_aB)
49 result_aB = np.matmul(result_aB, phi_ket)
50 # expected value of ab
51 matrix_ab = np.kron(a, b)
52 result_ab = np.matmul(phi_bra, matrix_ab)
53 result_ab = np.matmul(result_ab, phi_ket)
54
55 # CHSH value (AB - Ab + aB + ab)
56 result = result_AB - result_Ab + result_aB + result_ab
57 r.append(result)
58
59
60 # print(theta, result)
61 # for display
62 ax = plt.axes(projection='3d')
63
64 x = t1
65 y = t2
66 z = r
67
68 ax.set_xlabel("theta_alice")
69 ax.set_ylabel("theta_bob")
70 ax.set_zlabel("chsh_value")
71
72 ax.scatter(x, y, z)
73 plt.show()

```

Listing 3: CHSH Theoretical Experiment for General Basis Results