Untitled circuit_Oct 17, 2020 8:15 AM

October 17, 2020

[15]:

```
%matplotlib inline
# Importing standard Qiskit libraries and configuring account
from qiskit import *

# Loading your IBM Q account(s)
provider = IBMQ.load_account()
```

```
ibmqfactory.load_account:WARNING:2020-10-17 15:06:37,139: Credentials are
already in use. The existing account in the session will be replaced.
```

[50]:

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

# importing Qiskit
from qiskit import IBMQ, Aer
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister, execute
# import module for repetition
from qiskit.ignis.verification.topological_codes import RepetitionCode
from qiskit.ignis.verification.topological_codes import lookuptable_decoding
from qiskit.ignis.verification.topological_codes import GraphDecoder
# import basic plot tools
from qiskit.visualization import plot_histogram

from qiskit.providers.aer.noise import NoiseModel
from qiskit.providers.aer.noise.errors import pauli_error, depolarizing_error

def get_noise(p_meas,p_gate):
    error_meas = pauli_error([('X',p_meas), ('I', 1 - p_meas)])
    error_gate1 = depolarizing_error(p_gate, 1)
    error_gate2 = error_gate1.tensor(error_gate1)

    noise_model = NoiseModel()
    noise_model.add_all_qubit_quantum_error(error_meas, "measure") #␣
 ↪measurement error is applied to measurements
    noise_model.add_all_qubit_quantum_error(error_gate1, ["x"]) # single qubit␣
 ↪gate error is applied to x gates
```

```
    noise_model.add_all_qubit_quantum_error(error_gate2, ["cx"]) # two qubit␣
 ↪gate error is applied to cx gates

    return noise_model

# with depth D = 2 and with periodic boundary conditions,there are only two pos-
# sible causal cones: a 4-qubit cone enclosing Nb = 3 blocks
# time step tau = 0.1

def get_raw_results(code,noise_model):#=None
    circuits = code.get_circuit_list()
#     raw_results = {}
#     for log in range(2):
#         job = execute( circuits[log], Aer.get_backend('qasm_simulator'),␣
 ↪noise_model=noise_model)
#         raw_results[str(log)] = job.result().get_counts(str(log))
    table_results = {}
    for log in range(2):
        job = execute( circuits[log], Aer.get_backend('qasm_simulator'),␣
 ↪noise_model=noise_model, shots=10000 )
        table_results[str(log)] = job.result().get_counts(str(log))

    P = lookuptable_decoding(raw_results,table_results)
    print('P =',P)
    return [P,table_results]#raw_results

# 1.Cone
tau = 0.1
Nb = 3
Np = Nb/tau
qreg_q = QuantumRegister(4, 'q')
areg_q = QuantumRegister(1,'ancilla_qubit')
creg_c = ClassicalRegister(4, 'c')
qc = QuantumCircuit(qreg_q, areg_q, creg_c)
#for count in range(10):
for count in range(1):
    qc.h(areg_q[0])
    qc.cu1(np.pi/Nb, qreg_q[2], 4)
    qc.x(qreg_q[2])
    qc.cx(qreg_q[2],qreg_q[3])
    qc.cu1(np.pi/Nb, qreg_q[0], 4)
    qc.x(qreg_q[0])
    qc.cx(qreg_q[0],qreg_q[1])
    qc.cu1(np.pi/Nb, qreg_q[1], 4)
    qc.x(qreg_q[1])
    qc.cx(qreg_q[1],qreg_q[2])
    qc.h(areg_q[0])
    for i in range(4):
        qc.measure(qreg_q[i], creg_c[i])
        backend = Aer.get_backend('qasm_simulator')
        shots = 4096
    #    results = execute(qc, backend=backend, shots=shots).result()
    #    answer = results.get_counts()

    #    n = 4
        T = 10

        code = RepetitionCode(2,T)

        noise_model = get_noise(0.05,0.05)

        [P,raw_results] = get_raw_results(code,noise_model)

        plt.figure()
        plt.bar([0,1],[P['0'],P['1']]) #answer
        plt.title(str(i)+'th qubit'+''''s'''+' results')

        results = code.process_results(raw_results)
#        for log in raw_results:
#            print('Logical',log,':',raw_results[log],'\n')
        for log in ['0','1']:
            print('\nLogical ' + log + ':')
            print('raw results       ', {string:raw_results[log][string] for string in raw_results[log] if raw_results[log][string]>=50 })
            print('processed results ', {string:results[log][string] for string in results[log] if results[log][string]>=50 })

            for string in results[log]:
                if len(string) >0 & results[log][string]>=50:
                    plt.figure()
                    plt.bar(results[log][string]) #answer
 #answer
        #print('Results:',answer)
    #qc.measure(areg_q, creg_c)
#P['0'] = [0.1866, 0.2001, 0.1991, 0.1832]
#P['1'] = [0.2176, 0.2164, 0.212, 0.2253]

#plt.figure()
#plt.bar(P) #answer

qc.draw()
```

```
P = {'0': 0.1738, '1': 0.2218}

Logical 0:
raw results        {'00 0 0 0 0 0 0 0 0 0 0': 2060, '00 0 0 0 0 0 0 0 0 0 1':
193, '00 0 0 0 0 0 1 0 0 0 0': 213, '00 0 1 0 0 0 0 0 0 0 0': 199, '00 0 0 0 0 0
0 0 0 1 0': 204, '00 0 0 0 0 1 0 0 0 0 0': 245, '00 1 0 0 0 0 0 0 0 0 0': 218,
'00 0 0 0 0 0 0 0 1 0 0': 223, '00 0 0 0 1 0 0 0 0 0 0': 196, '01 0 0 0 0 0 0 0
0 0 0': 174, '01 1 0 0 0 0 0 0 0 0 0': 75, '01 1 1 0 0 0 0 0 0 0 0': 67, '01 1 1
1 0 0 0 0 0 0 0': 64, '01 1 1 1 1 0 0 0 0 0 0': 55, '01 1 1 1 1 1 0 0 0 0 0':
77, '01 1 1 1 1 1 1 0 0 0 0': 62, '01 1 1 1 1 1 1 1 0 0 0': 56, '01 1 1 1 1 1 1
1 1 0 0': 53, '01 1 1 1 1 1 1 1 1 1 0': 60, '00 0 0 0 0 0 0 1 0 0 0': 195, '00 0
0 1 0 0 0 0 0 0 0': 244, '10 0 0 0 0 0 0 0 0 0 0': 164, '10 1 0 0 0 0 0 0 0 0
0': 83, '10 1 1 0 0 0 0 0 0 0 0': 61, '10 1 1 1 0 0 0 0 0 0 0': 71, '10 1 1 1 1
1 0 0 0 0 0': 68, '10 1 1 1 1 1 1 0 0 0 0': 64, '10 1 1 1 1 1 1 1 0 0 0': 59,
'10 1 1 1 1 1 1 1 1 0 0': 80, '10 1 1 1 1 1 1 1 1 1 0': 74}
processed results  {'0 0  0 0 0 0 0 0 0 0 0 0 0': 2060, '0 0  1 1 0 0 0 0 0 0 0
0 0': 193, '0 0  0 0 0 0 1 1 0 0 0 0 0': 213, '0 0  0 0 0 0 0 0 0 0 1 1 0': 199,
'0 0  0 1 1 0 0 0 0 0 0 0 0': 204, '0 0  0 0 0 0 0 1 1 0 0 0 0': 245, '0 0  0 0
0 0 0 0 0 0 0 1 1': 218, '0 0  0 0 1 1 0 0 0 0 0 0 0': 223, '0 0  0 0 0 0 0 0 1
1 0 0 0': 196, '0 1  0 0 0 0 0 0 0 0 0 0 1': 174, '0 1  0 0 0 0 0 0 0 0 0 1 0':
75, '0 1  0 0 0 0 0 0 0 0 1 0 0': 67, '0 1  0 0 0 0 0 0 0 1 0 0 0': 64, '0 1  0
0 0 0 0 0 1 0 0 0 0': 55, '0 1  0 0 0 0 0 1 0 0 0 0 0': 77, '0 1  0 0 0 0 1 0 0
0 0 0 0': 62, '0 1  0 0 0 1 0 0 0 0 0 0 0': 56, '0 1  0 0 1 0 0 0 0 0 0 0 0':
53, '0 1  0 1 0 0 0 0 0 0 0 0 0': 60, '0 0  0 0 0 1 1 0 0 0 0 0 0': 195, '0 0  0
0 0 0 0 0 0 1 1 0 0': 244, '1 0  0 0 0 0 0 0 0 0 0 0 1': 164, '1 0  0 0 0 0 0 0
0 0 0 1 0': 83, '1 0  0 0 0 0 0 0 0 0 1 0 0': 61, '1 0  0 0 0 0 0 0 0 1 0 0 0':
71, '1 0  0 0 0 0 0 1 0 0 0 0 0': 68, '1 0  0 0 0 0 1 0 0 0 0 0 0': 64, '1 0  0
0 0 1 0 0 0 0 0 0 0': 59, '1 0  0 0 1 0 0 0 0 0 0 0 0': 80, '1 0  0 1 0 0 0 0 0 0
0 0 0 0': 74}

Logical 1:
raw results        {'01 0 0 0 0 0 0 0 0 0 0': 148, '01 1 0 0 0 0 0 0 0 0 0': 83,
'01 1 1 0 0 0 0 0 0 0 0': 75, '01 1 1 1 0 0 0 0 0 0 0': 61, '01 1 1 1 1 0 0 0 0
0 0': 75, '01 1 1 1 1 1 0 0 0 0 0': 62, '01 1 1 1 1 1 1 0 0 0 0': 69, '01 1 1 1
1 1 1 1 0 0 0': 61, '01 1 1 1 1 1 1 1 1 0 0': 64, '01 1 1 1 1 1 1 1 1 1 0': 59,
'01 1 1 1 1 1 1 1 1 1 1': 60, '10 0 0 0 0 0 0 0 0 0 0': 174, '10 1 0 0 0 0 0 0 0
0 0': 63, '10 1 1 0 0 0 0 0 0 0 0': 59, '10 1 1 1 0 0 0 0 0 0 0': 56, '10 1 1 1
1 0 0 0 0 0 0': 52, '10 1 1 1 1 1 0 0 0 0 0': 57, '10 1 1 1 1 1 1 0 0 0 0': 72,
'10 1 1 1 1 1 1 1 1 0 0': 57, '10 1 1 1 1 1 1 1 1 1 0': 70, '10 1 1 1 1 1 1 1 1
1 1': 50, '11 0 0 0 0 0 0 0 0 0 0': 1915, '11 0 0 0 0 0 0 0 0 0 1': 192, '11 0 0
0 0 0 0 0 0 1 0': 205, '11 0 0 0 0 0 0 0 1 0 0': 196, '11 0 0 0 0 0 0 1 0 0 0':
194, '11 0 0 0 0 0 1 0 0 0 0': 197, '11 0 0 0 0 1 0 0 0 0 0': 195, '11 0 0 0 1 0
0 0 0 0 0': 218, '11 0 0 1 0 0 0 0 0 0 0': 215, '11 0 1 0 0 0 0 0 0 0 0': 206,
'11 1 0 0 0 0 0 0 0 0 0': 242}
processed results  {'0 1  0 0 0 0 0 0 0 0 0 0 1': 148, '0 1  0 0 0 0 0 0 0 0 0 1
0': 83, '0 1  0 0 0 0 0 0 0 0 1 0 0': 75, '0 1  0 0 0 0 0 0 0 1 0 0 0': 61, '0 1
0 0 0 0 0 0 1 0 0 0 0': 75, '0 1  0 0 0 0 0 1 0 0 0 0 0': 62, '0 1  0 0 0 0 1 0
0 0 0 0 0': 69, '0 1  0 0 0 1 0 0 0 0 0 0 0': 61, '0 1  0 0 1 0 0 0 0 0 0 0 0':
```

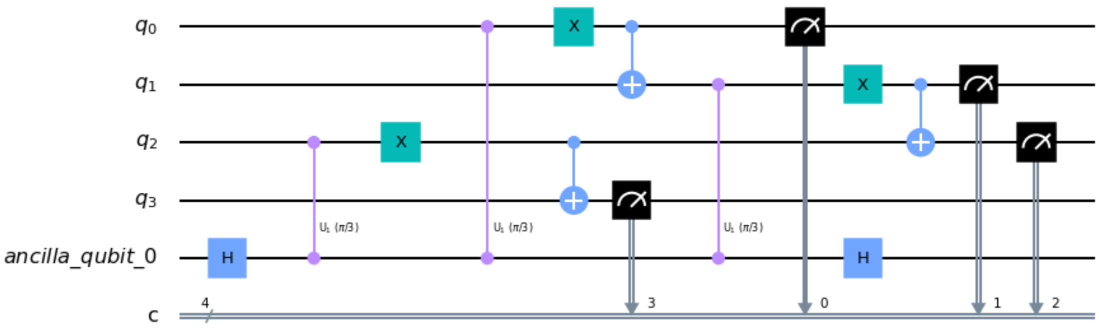
```
64, '0 1  0 1 0 0 0 0 0 0 0 0 0': 59, '0 1  1 0 0 0 0 0 0 0 0 0 0': 60, '1 0  0
0 0 0 0 0 0 0 0 0 1': 174, '1 0  0 0 0 0 0 0 0 0 0 1 0': 63, '1 0  0 0 0 0 0 0 0
0 1 0 0': 59, '1 0  0 0 0 0 0 0 0 1 0 0 0': 56, '1 0  0 0 0 0 0 0 1 0 0 0 0':
52, '1 0  0 0 0 0 0 1 0 0 0 0 0': 57, '1 0  0 0 0 0 1 0 0 0 0 0 0': 72, '1 0  0
0 1 0 0 0 0 0 0 0 0 0': 57, '1 0  0 1 0 0 0 0 0 0 0 0 0': 70, '1 0  1 0 0 0 0 0 0
0 0 0 0': 50, '1 1  0 0 0 0 0 0 0 0 0 0 0': 1915, '1 1  1 1 0 0 0 0 0 0 0 0 0':
192, '1 1  0 1 1 0 0 0 0 0 0 0 0': 205, '1 1  0 0 1 1 0 0 0 0 0 0 0': 196, '1 1
0 0 0 1 1 0 0 0 0 0 0': 194, '1 1  0 0 0 0 1 1 0 0 0 0 0': 197, '1 1  0 0 0 0 0
1 1 0 0 0 0': 195, '1 1  0 0 0 0 0 0 1 1 0 0 0': 218, '1 1  0 0 0 0 0 0 0 1 1 0
0': 215, '1 1  0 0 0 0 0 0 0 0 1 1 0': 206, '1 1  0 0 0 0 0 0 0 0 0 1 1': 242}
P = {'0': 0.1943, '1': 0.2016}

Logical 0:
raw results        {'00 0 0 0 0 0 0 0 0 0 0': 2052, '00 0 0 0 0 0 0 0 0 0 1':
205, '00 0 0 0 0 0 1 0 0 0 0': 221, '00 0 1 0 0 0 0 0 0 0 0': 227, '00 0 0 0 0 0
0 0 0 1 0': 215, '00 0 0 0 0 1 0 0 0 0 0': 186, '00 1 0 0 0 0 0 0 0 0 0': 221,
'00 0 0 0 0 0 0 0 1 0 0': 201, '00 0 0 0 1 0 0 0 0 0 0': 230, '01 0 0 0 0 0 0 0
0 0 0': 183, '01 1 0 0 0 0 0 0 0 0 0': 64, '01 1 1 0 0 0 0 0 0 0 0': 77, '01 1 1
1 0 0 0 0 0 0 0': 64, '01 1 1 1 1 0 0 0 0 0 0': 73, '01 1 1 1 1 1 0 0 0 0 0':
66, '01 1 1 1 1 1 1 0 0 0 0': 62, '01 1 1 1 1 1 1 1 0 0 0': 64, '01 1 1 1 1 1 1
1 1 0 0': 78, '01 1 1 1 1 1 1 1 1 1 0': 55, '00 0 0 0 0 0 0 1 0 0 0': 199, '00 0
0 1 0 0 0 0 0 0 0': 218, '10 0 0 0 0 0 0 0 0 0 0': 168, '10 1 0 0 0 0 0 0 0 0
0': 80, '10 1 1 0 0 0 0 0 0 0 0': 72, '10 1 1 1 0 0 0 0 0 0 0': 79, '10 1 1 1 1
0 0 0 0 0 0': 59, '10 1 1 1 1 1 0 0 0 0 0': 70, '10 1 1 1 1 1 1 0 0 0 0': 65,
'10 1 1 1 1 1 1 1 0 0 0': 73, '10 1 1 1 1 1 1 1 1 0 0': 54, '10 1 1 1 1 1 1 1 1
1 0': 64}
processed results  {'0 0  0 0 0 0 0 0 0 0 0 0 0': 2052, '0 0  1 1 0 0 0 0 0 0 0
0 0': 205, '0 0  0 0 0 0 1 1 0 0 0 0 0': 221, '0 0  0 0 0 0 0 0 0 0 1 1 0': 227,
'0 0  0 1 1 0 0 0 0 0 0 0 0': 215, '0 0  0 0 0 0 0 1 1 0 0 0 0': 186, '0 0  0 0
0 0 0 0 0 0 0 1 1': 221, '0 0  0 0 1 1 0 0 0 0 0 0 0': 201, '0 0  0 0 0 0 0 0 1
1 0 0 0': 230, '0 1  0 0 0 0 0 0 0 0 0 0 1': 183, '0 1  0 0 0 0 0 0 0 0 0 1 0':
64, '0 1  0 0 0 0 0 0 0 0 1 0 0': 77, '0 1  0 0 0 0 0 0 0 1 0 0 0': 64, '0 1  0
0 0 0 0 0 1 0 0 0 0': 73, '0 1  0 0 0 0 0 1 0 0 0 0 0': 66, '0 1  0 0 0 0 1 0 0
0 0 0 0': 62, '0 1  0 0 0 1 0 0 0 0 0 0 0': 64, '0 1  0 0 1 0 0 0 0 0 0 0 0':
78, '0 1  0 1 0 0 0 0 0 0 0 0 0': 55, '0 0  0 0 0 1 1 0 0 0 0 0 0': 199, '0 0  0
0 0 0 0 0 0 1 1 0 0': 218, '1 0  0 0 0 0 0 0 0 0 0 0 1': 168, '1 0  0 0 0 0 0 0
0 0 0 1 0': 80, '1 0  0 0 0 0 0 0 0 0 1 0 0': 72, '1 0  0 0 0 0 0 0 0 1 0 0 0':
79, '1 0  0 0 0 0 0 0 1 0 0 0 0': 59, '1 0  0 0 0 0 0 1 0 0 0 0 0': 70, '1 0  0
0 0 0 1 0 0 0 0 0 0': 65, '1 0  0 0 0 1 0 0 0 0 0 0 0': 73, '1 0  0 0 1 0 0 0 0
0 0 0 0': 54, '1 0  0 1 0 0 0 0 0 0 0 0 0': 64}

Logical 1:
raw results        {'01 0 0 0 0 0 0 0 0 0 0': 151, '01 1 0 0 0 0 0 0 0 0 0': 66,
'01 1 1 0 0 0 0 0 0 0 0': 55, '01 1 1 1 0 0 0 0 0 0 0': 55, '01 1 1 1 1 1 0 0 0
0 0': 56, '01 1 1 1 1 1 1 0 0 0 0': 63, '01 1 1 1 1 1 1 1 0 0 0': 59, '01 1 1 1
1 1 1 1 1 0 0': 67, '01 1 1 1 1 1 1 1 1 1 0': 59, '01 1 1 1 1 1 1 1 1 1 1': 59,
'10 0 0 0 0 0 0 0 0 0 0': 179, '10 1 0 0 0 0 0 0 0 0 0': 51, '10 1 1 0 0 0 0 0 0
0 0': 72, '10 1 1 1 0 0 0 0 0 0 0': 69, '10 1 1 1 1 0 0 0 0 0 0': 52, '10 1 1 1
```

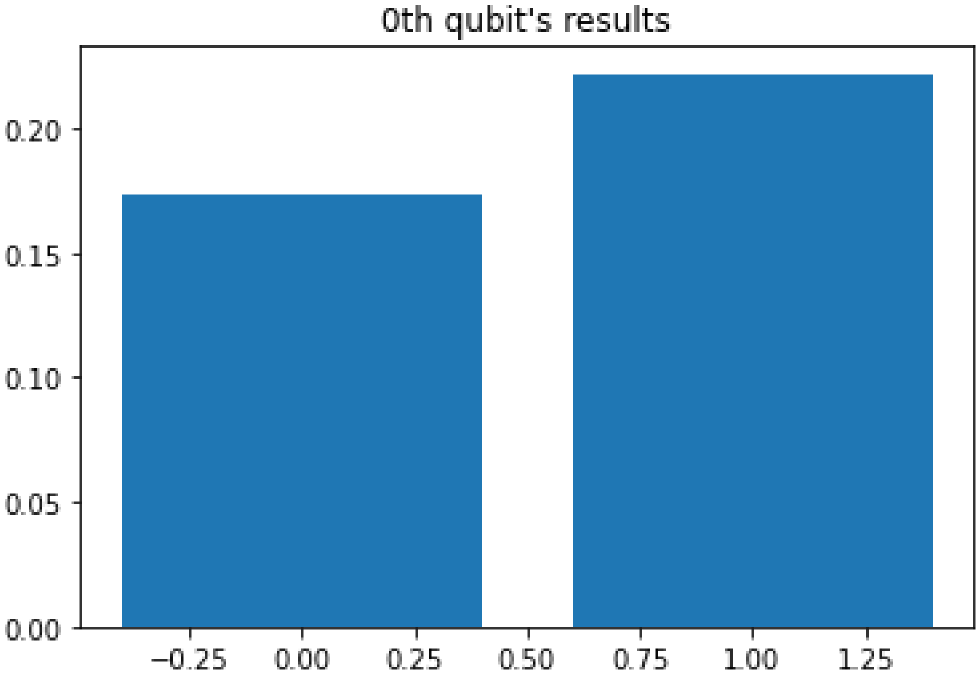
```
1 1 0 0 0 0 0': 56, '10 1 1 1 1 1 1 0 0 0 0': 57, '10 1 1 1 1 1 1 1 0 0 0': 66,
'10 1 1 1 1 1 1 1 1 0 0': 64, '10 1 1 1 1 1 1 1 1 1 0': 57, '10 1 1 1 1 1 1 1 1
1 1': 54, '11 0 0 0 0 0 0 0 0 0 0': 1959, '11 0 0 0 0 0 0 0 0 0 1': 199, '11 0 0
0 0 0 0 0 0 1 0': 218, '11 0 0 0 0 0 0 0 1 0 0': 214, '11 0 0 0 0 0 0 1 0 0 0':
188, '11 0 0 0 0 0 1 0 0 0 0': 213, '11 0 0 0 0 1 0 0 0 0 0': 198, '11 0 0 0 1 0
0 0 0 0 0': 206, '11 0 0 1 0 0 0 0 0 0 0': 208, '11 0 1 0 0 0 0 0 0 0 0': 206,
'11 1 0 0 0 0 0 0 0 0 0': 211}
processed results  {'0 1  0 0 0 0 0 0 0 0 0 0 1': 151, '0 1  0 0 0 0 0 0 0 0 0 1
0': 66, '0 1  0 0 0 0 0 0 0 0 1 0 0': 55, '0 1  0 0 0 0 0 0 0 1 0 0 0': 55, '0 1
0 0 0 0 0 1 0 0 0 0 0': 56, '0 1  0 0 0 0 1 0 0 0 0 0 0': 63, '0 1  0 0 0 1 0 0
0 0 0 0 0': 59, '0 1  0 0 1 0 0 0 0 0 0 0 0': 67, '0 1  0 1 0 0 0 0 0 0 0 0 0':
59, '0 1  1 0 0 0 0 0 0 0 0 0 0': 59, '1 0  0 0 0 0 0 0 0 0 0 0 1': 179, '1 0  0
0 0 0 0 0 0 0 0 1 0': 51, '1 0  0 0 0 0 0 0 0 0 1 0 0': 72, '1 0  0 0 0 0 0 0 0
1 0 0 0': 69, '1 0  0 0 0 0 0 0 1 0 0 0 0': 52, '1 0  0 0 0 0 0 1 0 0 0 0 0':
56, '1 0  0 0 0 0 1 0 0 0 0 0 0': 57, '1 0  0 0 0 1 0 0 0 0 0 0 0': 66, '1 0  0
0 1 0 0 0 0 0 0 0 0': 64, '1 0  0 1 0 0 0 0 0 0 0 0 0': 57, '1 0  1 0 0 0 0 0 0
0 0 0 0': 54, '1 1  0 0 0 0 0 0 0 0 0 0 0': 1959, '1 1  1 1 0 0 0 0 0 0 0 0 0':
199, '1 1  0 1 1 0 0 0 0 0 0 0 0': 218, '1 1  0 0 1 1 0 0 0 0 0 0 0': 214, '1 1
0 0 0 1 1 0 0 0 0 0 0': 188, '1 1  0 0 0 0 1 1 0 0 0 0 0': 213, '1 1  0 0 0 0 0
1 1 0 0 0 0': 198, '1 1  0 0 0 0 0 0 1 1 0 0 0': 206, '1 1  0 0 0 0 0 0 0 1 1 0
0': 208, '1 1  0 0 0 0 0 0 0 0 1 1 0': 206, '1 1  0 0 0 0 0 0 0 0 0 1 1': 211}
P = {'0': 0.1811, '1': 0.2304}

Logical 0:
raw results        {'00 0 0 0 0 0 0 0 0 0 0': 2073, '00 0 0 0 0 0 0 0 0 0 1':
202, '00 0 0 0 0 0 1 0 0 0 0': 220, '00 0 1 0 0 0 0 0 0 0 0': 209, '00 0 0 0 0 0
0 0 0 1 0': 211, '00 0 0 0 0 1 0 0 0 0 0': 211, '00 1 0 0 0 0 0 0 0 0 0': 225,
'00 0 0 0 0 0 0 0 1 0 0': 209, '00 0 0 0 1 0 0 0 0 0 0': 215, '01 0 0 0 0 0 0 0
0 0 0': 156, '01 1 0 0 0 0 0 0 0 0 0': 72, '01 1 1 0 0 0 0 0 0 0 0': 64, '01 1 1
1 0 0 0 0 0 0 0': 59, '01 1 1 1 1 0 0 0 0 0 0': 72, '01 1 1 1 1 1 0 0 0 0 0':
74, '01 1 1 1 1 1 1 0 0 0 0': 72, '01 1 1 1 1 1 1 1 0 0 0': 85, '01 1 1 1 1 1 1
1 1 0 0': 67, '01 1 1 1 1 1 1 1 1 1 0': 52, '00 0 0 0 0 0 0 1 0 0 0': 220, '00 0
0 1 0 0 0 0 0 0 0': 185, '10 0 0 0 0 0 0 0 0 0 0': 187, '10 1 0 0 0 0 0 0 0 0
0': 69, '10 1 1 0 0 0 0 0 0 0 0': 62, '10 1 1 1 0 0 0 0 0 0 0': 60, '10 1 1 1 1
0 0 0 0 0 0': 69, '10 1 1 1 1 1 0 0 0 0 0': 61, '10 1 1 1 1 1 1 0 0 0 0': 62,
'10 1 1 1 1 1 1 1 0 0 0': 71, '10 1 1 1 1 1 1 1 1 0 0': 60, '10 1 1 1 1 1 1 1 1
1 0': 55}
processed results  {'0 0  0 0 0 0 0 0 0 0 0 0 0': 2073, '0 0  1 1 0 0 0 0 0 0 0
0 0': 202, '0 0  0 0 0 0 1 1 0 0 0 0 0': 220, '0 0  0 0 0 0 0 0 0 0 1 1 0': 209,
'0 0  0 1 1 0 0 0 0 0 0 0 0': 211, '0 0  0 0 0 0 0 1 1 0 0 0 0': 211, '0 0  0 0
0 0 0 0 0 0 0 1 1': 225, '0 0  0 0 1 1 0 0 0 0 0 0 0': 209, '0 0  0 0 0 0 0 0 1
1 0 0 0': 215, '0 1  0 0 0 0 0 0 0 0 0 0 1': 156, '0 1  0 0 0 0 0 0 0 0 0 1 0':
72, '0 1  0 0 0 0 0 0 0 0 1 0 0': 64, '0 1  0 0 0 0 0 0 0 1 0 0 0': 59, '0 1  0
0 0 0 0 0 1 0 0 0 0': 72, '0 1  0 0 0 0 0 1 0 0 0 0 0': 74, '0 1  0 0 0 0 1 0 0
0 0 0 0': 72, '0 1  0 0 0 1 0 0 0 0 0 0 0': 85, '0 1  0 0 1 0 0 0 0 0 0 0 0':
67, '0 1  0 1 0 0 0 0 0 0 0 0 0': 52, '0 0  0 0 0 1 1 0 0 0 0 0 0': 220, '0 0  0
0 0 0 0 0 0 1 1 0 0': 185, '1 0  0 0 0 0 0 0 0 0 0 0 1': 187, '1 0  0 0 0 0 0 0
0 0 0 1 0': 69, '1 0  0 0 0 0 0 0 0 0 1 0 0': 62, '1 0  0 0 0 0 0 0 0 1 0 0 0':
```

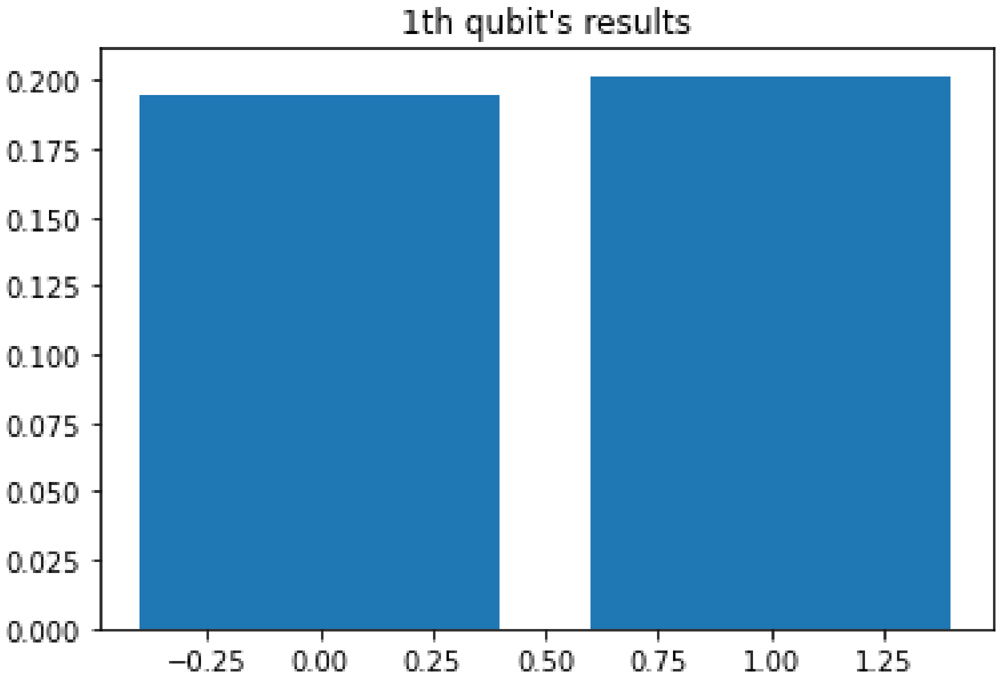
```
60, '1 0  0 0 0 0 0 0 1 0 0 0 0': 69, '1 0  0 0 0 0 0 1 0 0 0 0 0': 61, '1 0  0
0 0 0 1 0 0 0 0 0 0': 62, '1 0  0 0 0 1 0 0 0 0 0 0 0': 71, '1 0  0 0 1 0 0 0 0
0 0 0 0': 60, '1 0  0 1 0 0 0 0 0 0 0 0 0': 55}

Logical 1:
raw results        {'01 0 0 0 0 0 0 0 0 0 0': 182, '01 1 0 0 0 0 0 0 0 0 0': 61,
'01 1 1 0 0 0 0 0 0 0 0': 59, '01 1 1 1 0 0 0 0 0 0 0': 57, '01 1 1 1 1 0 0 0 0
0 0': 73, '01 1 1 1 1 1 0 0 0 0 0': 64, '01 1 1 1 1 1 1 0 0 0 0': 68, '01 1 1 1
1 1 1 1 0 0 0': 63, '01 1 1 1 1 1 1 1 1 0 0': 64, '01 1 1 1 1 1 1 1 1 1 0': 59,
'01 1 1 1 1 1 1 1 1 1 1': 60, '10 0 0 0 0 0 0 0 0 0 0': 150, '10 1 0 0 0 0 0 0 0
0 0': 63, '10 1 1 0 0 0 0 0 0 0 0': 67, '10 1 1 1 0 0 0 0 0 0 0': 59, '10 1 1 1
1 0 0 0 0 0 0': 66, '10 1 1 1 1 1 0 0 0 0 0': 57, '10 1 1 1 1 1 1 0 0 0 0': 70,
'10 1 1 1 1 1 1 1 1 0 0': 54, '10 1 1 1 1 1 1 1 1 1 0': 58, '10 1 1 1 1 1 1 1 1
1 1': 55, '11 0 0 0 0 0 0 0 0 0 0': 1865, '11 0 0 0 0 0 0 0 0 0 1': 203, '11 0 0
0 0 0 0 0 0 1 0': 241, '11 0 0 0 0 0 0 0 1 0 0': 223, '11 0 0 0 0 0 0 1 0 0 0':
204, '11 0 0 0 0 0 1 0 0 0 0': 195, '11 0 0 0 0 1 0 0 0 0 0': 217, '11 0 0 0 1 0
0 0 0 0 0': 186, '11 0 0 1 0 0 0 0 0 0 0': 177, '11 0 1 0 0 0 0 0 0 0 0': 207,
'11 1 0 0 0 0 0 0 0 0 0': 201}
processed results  {'0 1  0 0 0 0 0 0 0 0 0 0 1': 182, '0 1  0 0 0 0 0 0 0 0 0 1
0': 61, '0 1  0 0 0 0 0 0 0 0 1 0 0': 59, '0 1  0 0 0 0 0 0 0 1 0 0 0': 57, '0 1
0 0 0 0 0 0 1 0 0 0 0': 73, '0 1  0 0 0 0 0 1 0 0 0 0 0': 64, '0 1  0 0 0 0 1 0
0 0 0 0 0': 68, '0 1  0 0 0 1 0 0 0 0 0 0 0': 63, '0 1  0 0 1 0 0 0 0 0 0 0 0':
64, '0 1  0 1 0 0 0 0 0 0 0 0 0': 59, '0 1  1 0 0 0 0 0 0 0 0 0 0': 60, '1 0  0
0 0 0 0 0 0 0 0 0 1': 150, '1 0  0 0 0 0 0 0 0 0 0 1 0': 63, '1 0  0 0 0 0 0 0 0
0 1 0 0': 67, '1 0  0 0 0 0 0 0 0 1 0 0 0': 59, '1 0  0 0 0 0 0 0 1 0 0 0 0':
66, '1 0  0 0 0 0 0 1 0 0 0 0 0': 57, '1 0  0 0 0 0 1 0 0 0 0 0 0': 70, '1 0  0
0 1 0 0 0 0 0 0 0 0': 54, '1 0  0 1 0 0 0 0 0 0 0 0 0': 58, '1 0  1 0 0 0 0 0 0
0 0 0 0': 55, '1 1  0 0 0 0 0 0 0 0 0 0 0': 1865, '1 1  1 1 0 0 0 0 0 0 0 0 0':
203, '1 1  0 1 1 0 0 0 0 0 0 0 0': 241, '1 1  0 0 1 1 0 0 0 0 0 0 0': 223, '1 1
0 0 0 1 1 0 0 0 0 0 0': 204, '1 1  0 0 0 0 1 1 0 0 0 0 0': 195, '1 1  0 0 0 0 0
1 1 0 0 0 0': 217, '1 1  0 0 0 0 0 0 1 1 0 0 0': 186, '1 1  0 0 0 0 0 0 0 1 1 0
0': 177, '1 1  0 0 0 0 0 0 0 0 1 1 0': 207, '1 1  0 0 0 0 0 0 0 0 0 1 1': 201}
P = {'0': 0.1802, '1': 0.2231}

Logical 0:
raw results        {'00 0 0 0 0 0 0 0 0 0 0': 2023, '00 0 0 0 0 0 0 0 0 0 1':
218, '00 0 0 0 0 0 1 0 0 0 0': 220, '00 0 1 0 0 0 0 0 0 0 0': 217, '00 0 0 0 0 0
0 0 0 1 0': 229, '00 0 0 0 0 1 0 0 0 0 0': 223, '00 1 0 0 0 0 0 0 0 0 0': 222,
'00 0 0 0 0 0 0 0 1 0 0': 235, '00 0 0 0 1 0 0 0 0 0 0': 210, '01 0 0 0 0 0 0 0
0 0 0': 184, '01 1 0 0 0 0 0 0 0 0 0': 69, '01 1 1 0 0 0 0 0 0 0 0': 63, '01 1 1
1 0 0 0 0 0 0 0': 50, '01 1 1 1 1 0 0 0 0 0 0': 77, '01 1 1 1 1 1 0 0 0 0 0':
68, '01 1 1 1 1 1 1 0 0 0 0': 55, '01 1 1 1 1 1 1 1 0 0 0': 70, '01 1 1 1 1 1 1
1 1 0 0': 72, '01 1 1 1 1 1 1 1 1 1 0': 55, '00 0 0 0 0 0 0 1 0 0 0': 196, '00 0
0 1 0 0 0 0 0 0 0': 211, '10 0 0 0 0 0 0 0 0 0 0': 168, '10 1 0 0 0 0 0 0 0 0
0': 67, '10 1 1 1 0 0 0 0 0 0 0': 54, '10 1 1 1 1 0 0 0 0 0 0': 66, '10 1 1 1 1
1 0 0 0 0 0': 72, '10 1 1 1 1 1 1 0 0 0 0': 67, '10 1 1 1 1 1 1 1 0 0 0': 67,
'10 1 1 1 1 1 1 1 1 0 0': 62}
processed results  {'0 0  0 0 0 0 0 0 0 0 0 0 0': 2023, '0 0  1 1 0 0 0 0 0 0 0
```

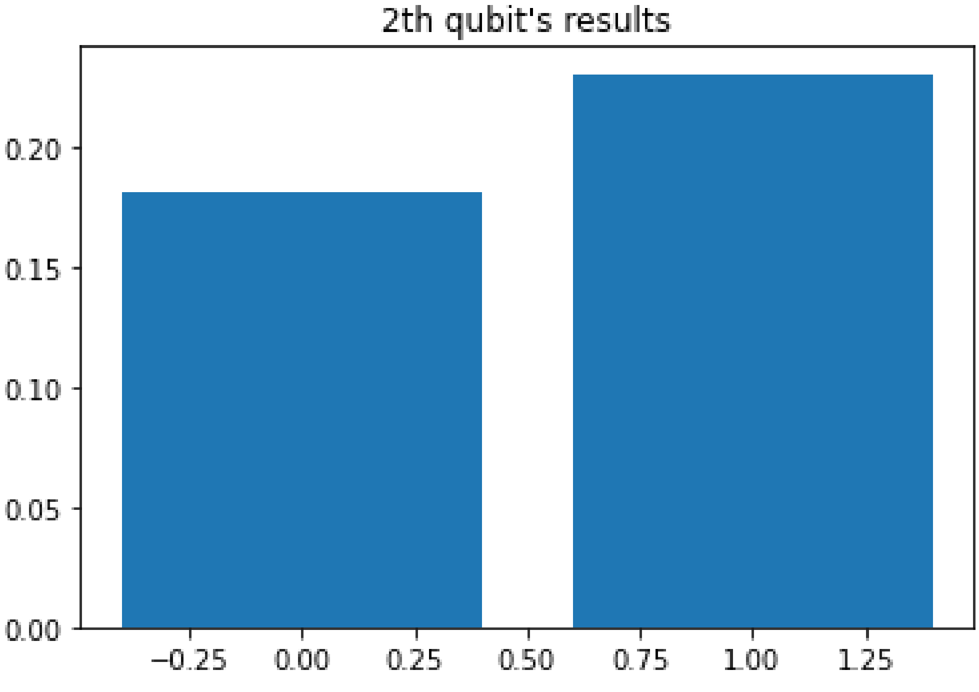
```
0 0': 218, '0 0  0 0 0 0 1 1 0 0 0 0 0': 220, '0 0  0 0 0 0 0 0 0 0 1 1 0': 217,
'0 0  0 1 1 0 0 0 0 0 0 0 0': 229, '0 0  0 0 0 0 0 1 1 0 0 0 0': 223, '0 0  0 0
0 0 0 0 0 0 0 1 1': 222, '0 0  0 0 1 1 0 0 0 0 0 0 0': 235, '0 0  0 0 0 0 0 0 1
1 0 0 0': 210, '0 1  0 0 0 0 0 0 0 0 0 0 1': 184, '0 1  0 0 0 0 0 0 0 0 0 1 0':
69, '0 1  0 0 0 0 0 0 0 0 1 0 0': 63, '0 1  0 0 0 0 0 0 0 1 0 0 0': 50, '0 1  0
0 0 0 0 0 1 0 0 0 0': 77, '0 1  0 0 0 0 0 1 0 0 0 0 0': 68, '0 1  0 0 0 0 1 0 0
0 0 0 0': 55, '0 1  0 0 0 1 0 0 0 0 0 0 0': 70, '0 1  0 0 1 0 0 0 0 0 0 0 0':
72, '0 1  0 1 0 0 0 0 0 0 0 0 0': 55, '0 0  0 0 0 1 1 0 0 0 0 0 0': 196, '0 0  0
0 0 0 0 0 0 1 1 0 0': 211, '1 0  0 0 0 0 0 0 0 0 0 0 1': 168, '1 0  0 0 0 0 0 0
0 0 0 1 0': 67, '1 0  0 0 0 0 0 0 0 1 0 0 0': 54, '1 0  0 0 0 0 0 0 1 0 0 0 0':
66, '1 0  0 0 0 0 0 1 0 0 0 0 0': 72, '1 0  0 0 0 0 1 0 0 0 0 0 0': 67, '1 0  0
0 0 1 0 0 0 0 0 0 0': 67, '1 0  0 0 1 0 0 0 0 0 0 0 0': 62}

Logical 1:
raw results        {'01 0 0 0 0 0 0 0 0 0 0': 173, '01 1 0 0 0 0 0 0 0 0 0': 65,
'01 1 1 0 0 0 0 0 0 0 0': 63, '01 1 1 1 0 0 0 0 0 0 0': 60, '01 1 1 1 1 0 0 0 0
0 0': 67, '01 1 1 1 1 1 0 0 0 0 0': 54, '01 1 1 1 1 1 1 0 0 0 0': 66, '01 1 1 1
1 1 1 1 0 0 0': 69, '01 1 1 1 1 1 1 1 1 0 0': 80, '01 1 1 1 1 1 1 1 1 1 0': 56,
'01 1 1 1 1 1 1 1 1 1 1': 70, '10 0 0 0 0 0 0 0 0 0 0': 147, '10 1 0 0 0 0 0 0 0
0 0': 78, '10 1 1 0 0 0 0 0 0 0 0': 72, '10 1 1 1 0 0 0 0 0 0 0': 55, '10 1 1 1
1 0 0 0 0 0 0': 61, '10 1 1 1 1 1 0 0 0 0 0': 56, '10 1 1 1 1 1 1 0 0 0 0': 64,
'10 1 1 1 1 1 1 1 0 0 0': 54, '10 1 1 1 1 1 1 1 1 0 0': 72, '10 1 1 1 1 1 1 1 1
1 0': 53, '10 1 1 1 1 1 1 1 1 1 1': 60, '11 0 0 0 0 0 0 0 0 0 0': 1955, '11 0 0
0 0 0 0 0 0 0 1': 198, '11 0 0 0 0 0 0 0 0 1 0': 202, '11 0 0 0 0 0 0 0 1 0 0':
177, '11 0 0 0 0 0 0 1 0 0 0': 222, '11 0 0 0 0 0 1 0 0 0 0': 180, '11 0 0 0 0 1
0 0 0 0 0': 187, '11 0 0 0 1 0 0 0 0 0 0': 212, '11 0 0 1 0 0 0 0 0 0 0': 227,
'11 0 1 0 0 0 0 0 0 0 0': 209, '11 1 0 0 0 0 0 0 0 0 0': 232}
processed results  {'0 1  0 0 0 0 0 0 0 0 0 0 1': 173, '0 1  0 0 0 0 0 0 0 0 0 1
0': 65, '0 1  0 0 0 0 0 0 0 0 1 0 0': 63, '0 1  0 0 0 0 0 0 0 1 0 0 0': 60, '0 1
0 0 0 0 0 0 1 0 0 0 0': 67, '0 1  0 0 0 0 0 1 0 0 0 0 0': 54, '0 1  0 0 0 0 1 0
0 0 0 0 0': 66, '0 1  0 0 0 1 0 0 0 0 0 0 0': 69, '0 1  0 0 1 0 0 0 0 0 0 0 0':
80, '0 1  0 1 0 0 0 0 0 0 0 0 0': 56, '0 1  1 0 0 0 0 0 0 0 0 0 0': 70, '1 0  0
0 0 0 0 0 0 0 0 0 0 1': 147, '1 0  0 0 0 0 0 0 0 0 0 1 0': 78, '1 0  0 0 0 0 0 0 0
0 1 0 0': 72, '1 0  0 0 0 0 0 0 1 0 0 0': 55, '1 0  0 0 0 0 0 0 1 0 0 0 0':
61, '1 0  0 0 0 0 0 1 0 0 0 0 0': 56, '1 0  0 0 0 0 1 0 0 0 0 0 0': 64, '1 0  0
0 0 1 0 0 0 0 0 0 0': 54, '1 0  0 0 1 0 0 0 0 0 0 0 0': 72, '1 0  0 1 0 0 0 0 0
0 0 0 0': 53, '1 0  1 0 0 0 0 0 0 0 0 0 0': 60, '1 1  0 0 0 0 0 0 0 0 0 0 0':
1955, '1 1  1 1 0 0 0 0 0 0 0 0 0': 198, '1 1  0 1 1 0 0 0 0 0 0 0 0': 202, '1 1
0 0 1 1 0 0 0 0 0 0 0': 177, '1 1  0 0 0 1 1 0 0 0 0 0 0': 222, '1 1  0 0 0 0 1
1 0 0 0 0 0': 180, '1 1  0 0 0 0 0 1 1 0 0 0 0': 187, '1 1  0 0 0 0 0 0 1 1 0 0
0': 212, '1 1  0 0 0 0 0 0 0 1 1 0 0': 227, '1 1  0 0 0 0 0 0 0 0 1 1 0': 209,
'1 1  0 0 0 0 0 0 0 0 0 1 1': 232}
```

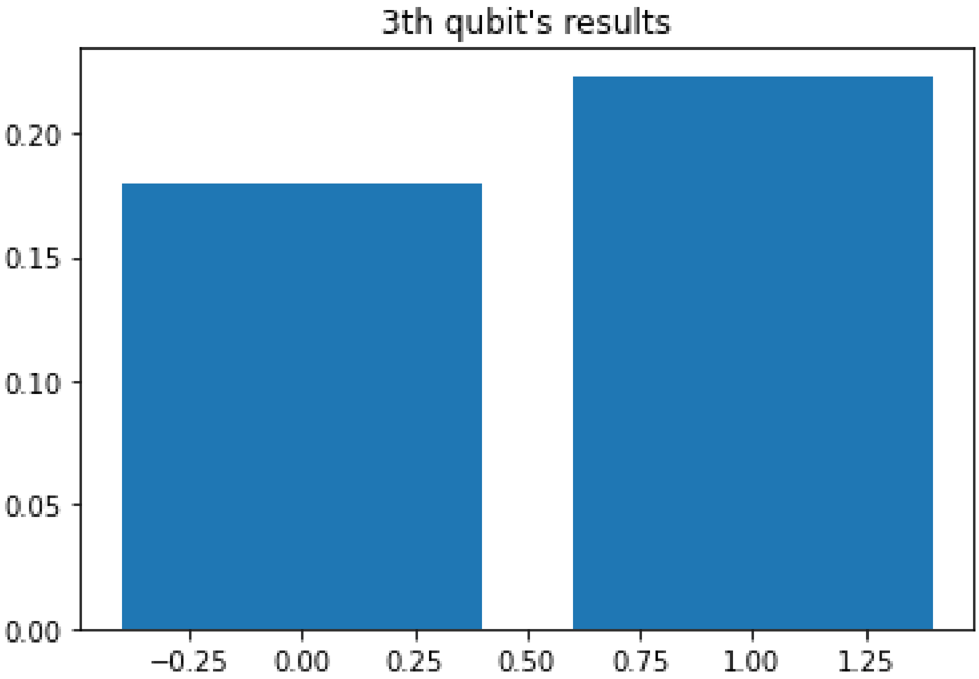
[50]:


q_0
q_1
q_2
q_3
ancilla_qubit_0
c
4
H
U_1 (π/3)
X
X
U_1 (π/3)
U_1 (π/3)
X
H
3
0
1
2


0th qubit's results
0.20
0.15
0.10
0.05
0.00
−0.25
0.00
0.25
0.50
0.75
1.00
1.25


1th qubit's results
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
−0.25
0.00
0.25
0.50
0.75
1.00
1.25


2th qubit's results
0.20
0.15
0.10
0.05
0.00
−0.25
0.00
0.25
0.50
0.75
1.00
1.25


3th qubit's results
0.00
0.05
0.10
0.15
0.20
−0.25
0.00
0.25
0.50
0.75
1.00
1.25

```
[ ]:
```