IX. SUPPLEMENTARY INFORMATION: QISKIT CODES FOR OPTIMIZATION.

For Energy estimation Z_0I, IZ_1 and Z_0Z_1 terms of Hamiltonian respectively are coded then optimized through scipy.optimize which contain 'powell', 'nelder-mead' or 'cobyla'. The QASM code for the same is as follows:

Code for Z1(Z_0I):

```
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 13 20:04:16 2019

@author: Rahul
"""

from scipy.optimize import minimize
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister
import numpy as np
from qiskit import execute
from qiskit import BasicAer
backend = BasicAer.get_backend('qasm_simulator')
T=8192
def Z1(theta):
    # Create a Quantum Register called "q" with 3 qubits
    q = QuantumRegister(2)

    # Create a Classical Register called "c" with 3 bits
    c = ClassicalRegister(2)
    qc = QuantumCircuit(q,c)

    qc.u1(theta[0],q[0])
    qc.u3(theta[1],-np.pi/2,np.pi/2,q[0])
    qc.u1(theta[2],q[0])
    qc.cx(q[0], q[1])
    qc.u1(theta[3],q[1])
    qc.u3(theta[4],-np.pi/2,np.pi/2,q[1])
    qc.u1(theta[5],q[1])
    qc.cx(q[1], q[0])
    qc.u1(theta[6],q[0])
    qc.u1(theta[7],q[1])
    qc.u3(theta[8],-np.pi/2,np.pi/2,q[0])
    qc.u3(theta[9],-np.pi/2,np.pi/2,q[1])
    qc.u1(theta[10],q[0])
    qc.u1(theta[11],q[1])
    qc.z(q[0])

    qc.measure(q[0], c[0])
    qc.measure(q[1], c[1])

    #print(qc)
    #print(i)
    shots = T       # Number of shots to run the program (experiment); maximum is 8192 shots.
    max_credits = 3          # Maximum number of credits to spend on executions.

    job_hpc = execute(qc, backend=backend, shots=shots, max_credits=max_credits)
    result_hpc = job_hpc.result()
    counts11 = result_hpc.get_counts(qc)
    #print(counts11)
    Z=0
    if '00' in list(counts11):
        Z=Z+counts11['00']/T
        #print(Z)
    if '01' in list(counts11):
        Z=Z+counts11['01']/T
        #print(Z)
    if '10' in list(counts11):
        Z=Z-counts11['10']/T
        #print(Z)
    if '11' in list(counts11):
        Z=Z-counts11['11']/T
        #print(Z)

    return Z
theta0=[0,np.pi/2,0,0,np.pi/2,0,0,0,np.pi/2,np.pi/2,0,0]
res = minimize(Z1, theta0, method='powell', options={'xtol': 1e-8 , 'disp': True})
print(res,'Z1')
#x=Z1([ 1.53637770e-03,  1.55027283e+00, 8.09473120e-04,  1.66340534e-04,-8.32420733e-03,  -2.28166637e-05,  -8.94140209e-04, 5.61023344e-04,1.77675920e+00,  1.58425510e+00,  1.09455026e-03,  8.51956374e-04])
#print(x)
```

Code for Z2(IZ_1):

```
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 13 20:04:17 2019

@author: Rahul
"""

from scipy.optimize import minimize
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister
import numpy as np
from qiskit import execute
from qiskit import BasicAer
backend = BasicAer.get_backend('qasm_simulator')
T=8192
def Z2(theta):
    # Create a Quantum Register called "q" with 3 qubits
    q = QuantumRegister(2)

    # Create a Classical Register called "c" with 3 bits
    c = ClassicalRegister(2)
    qc = QuantumCircuit(q,c)

    qc.u1(theta[0],q[0])
    qc.u3(theta[1],-np.pi/2,np.pi/2,q[0])
    qc.u1(theta[2],q[0])
    qc.cx(q[0], q[1])
    qc.u1(theta[3],q[1])
    qc.u3(theta[4],-np.pi/2,np.pi/2,q[1])
    qc.u1(theta[5],q[1])
    qc.cx(q[1], q[0])
    qc.u1(theta[6],q[0])
    qc.u1(theta[7],q[1])
    qc.u3(theta[8],-np.pi/2,np.pi/2,q[0])
    qc.u3(theta[9],-np.pi/2,np.pi/2,q[1])
    qc.u1(theta[10],q[0])
    qc.u1(theta[11],q[1])
    qc.z(q[1])

    qc.measure(q[0], c[0])
    qc.measure(q[1], c[1])

    #print(qc)
```

```
    #print(i)
    shots = T           # Number of shots to run the program (experiment); maximum is 8192 shots.
    max_credits = 3         # Maximum number of credits to spend on executions.

    job_hpc = execute(qc, backend=backend, shots=shots, max_credits=max_credits)
    result_hpc = job_hpc.result()
    counts22 = result_hpc.get_counts(qc)

    Z=0
    if '00' in list(counts22):
        Z=Z+counts22['00']/T
        print(Z)
    if '01' in list(counts22):
        Z=Z-counts22['01']/T
        print(Z)
    if '10' in list(counts22):
        Z=Z+counts22['10']/T
        print(Z)
    if '11' in list(counts22):
        Z=Z-counts22['11']/T
        print(Z)

    return Z
theta0=[0,np.pi/2,0,0,np.pi/2,0,0,0,np.pi/2,np.pi/2,0,0]
res = minimize(Z2, theta0, method='nelder-mead', options={'xtol': 1e-8 , 'disp': True})
print(res,'Z2')
#z=Z2([-5.64200351e-01,  1.61554986e+00,  1.56821823e+00,  1.61219634e-03, 1.81902336e+00,  5.96777406e+00, -6.66574506e-03,  4.63725496e+00, 1.32156756e+00,  3.78827977e-01,  9.59049221e+00,  3.90466495e+00])
#print(z)
```

Code for Z3($Z_0 Z_I$):

```
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 13 20:04:15 2019

@author: Rahul
"""

from scipy.optimize import minimize
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister
import numpy as np
from qiskit import execute
from qiskit import IBMQ
backend =IBMQ.get_backend('ibmqx4')
T=8192
def Z3(theta):
    # Create a Quantum Register called "q" with 3 qubits
    q = QuantumRegister(2)

    # Create a Classical Register called "c" with 3 bits
    c = ClassicalRegister(2)
    qc = QuantumCircuit(q,c)

    qc.u1(theta[0],q[0])
    qc.u3(theta[1],-np.pi/2,np.pi/2,q[0])
    qc.u1(theta[2],q[0])
    qc.cx(q[0], q[1])
    qc.u1(theta[3],q[1])
    qc.u3(theta[4],-np.pi/2,np.pi/2,q[1])
    qc.u1(theta[5],q[1])
    qc.cx(q[1], q[0])
    qc.u1(theta[6],q[0])
    qc.u1(theta[7],q[1])
    qc.u3(theta[8],-np.pi/2,np.pi/2,q[0])
    qc.u3(theta[9],-np.pi/2,np.pi/2,q[1])
    qc.u1(theta[10],q[0])
    qc.u1(theta[11],q[1])
    qc.z(q[0])
    qc.z(q[1])

    qc.measure(q[0], c[0])
    qc.measure(q[1], c[1])

    #print(qc)
    #print(i)
    shots= T           # Number of shots to run the program (experiment); maximum is 8192 shots.
    max_credits = 3         # Maximum number of credits to spend on executions.

    job_hpc = execute(qc, backend=backend, shots=shots, max_credits=max_credits)
    result_hpc = job_hpc.result()
    counts12 = result_hpc.get_counts(qc)

    Z=0
    if '00' in list(counts12):
        Z=counts12['00']/T
        print(Z)
    if '01' in list(counts12):
        Z=Z-counts12['01']/T
        print(Z)
    if '10' in list(counts12):
        Z=Z-counts12['10']/T
        print(Z)
    if '11' in list(counts12):
        Z=Z+counts12['11']/T
        print(Z)
    return Z
theta0=[0,np.pi/2,0,0,np.pi/2,0,0,0,np.pi/2,np.pi/2,0,0]
res = minimize(Z3, theta0, method='nelder-mead', options={'xtol': 1e-8 , 'disp': True})
print(res,'Z3')
#y=Z3([-7.63609374, -0.31633082,  2.15909817,  4.31092763,  1.57470161, 1.6111563 , -0.13041641,  0.08880943,  1.58865914,  1.50475982, 0.59323763,  2.86816974])
#print(y)
```

The tabulated data of convergence is as shown below.

TABLE I. For Nelder Mead

Iterations	Theoretical	Experimental
1	-0.52952	-0.3658118514
2	-0.52952	-0.3687388385
3	-0.52952	-0.364456499
4	-0.52952	-0.3640980715
5	-0.52952	-0.3735459878
6	-0.52952	-0.3547868431
7	-0.52952	-0.3642996161
8	-0.52952	-0.3652836663
9	-0.52952	-0.3730938032
10	-0.52952	-0.3646086323
11	-0.52952	-0.3657974241
12	-0.52952	-0.3712501318
13	-0.52952	-0.3667064972
14	-0.52952	-0.3660282661
15	-0.52952	-0.3702449038
16	-0.52952	-0.3702913827
17	-0.52952	-0.3308180563
18	-0.52952	-0.3653252632
19	-0.52952	-0.3770401351
20	-0.52952	-0.3867578684
21	-0.52952	-0.3746037324
22	-0.52952	-0.3745729229
23	-0.52952	-0.375111539
24	-0.52952	-0.3759054584
25	-0.52952	-0.3819807294
26	-0.52952	-0.3751555337
27	-0.52952	-0.3774191599
28	-0.52952	-0.3790968918
29	-0.52952	-0.3830969824
30	-0.52952	-0.3842554535
31	-0.52952	-0.3821377446
32	-0.52952	-0.3852416368
33	-0.52952	-0.387955228
34	-0.52952	-0.3997053027
35	-0.52952	-0.3925481822
36	-0.52952	-0.3912610618
37	-0.52952	-0.3925515116
38	-0.52952	-0.39142132
39	-0.52952	-0.3948359592
40	-0.52952	-0.3993956196
41	-0.52952	-0.3929185069
42	-0.52952	-0.3984600897
43	-0.52952	-0.3981955975
44	-0.52952	-0.3995114402
45	-0.52952	-0.3982939261
46	-0.52952	-0.4039865324
47	-0.52952	-0.4040763391
48	-0.52952	-0.4061567579
49	-0.52952	-0.4068460868
50	-0.52952	-0.4075465596

Iterations	Theoretical	Experimental
51	-0.52952	-0.4042369078
52	-0.52952	-0.4167492066
53	-0.52952	-0.4100107529
54	-0.52952	-0.4121749407
55	-0.52952	-0.4123393275
56	-0.52952	-0.4153778284
57	-0.52952	-0.4201981629
58	-0.52952	-0.416993412
59	-0.52952	-0.4075684502
60	-0.52952	-0.4216554432
61	-0.52952	-0.4229979668
62	-0.52952	-0.4196147123
63	-0.52952	-0.4253503764
64	-0.52952	-0.4249917301
65	-0.52952	-0.4299162527
66	-0.52952	-0.4330937475
67	-0.52952	-0.4257880472
68	-0.52952	-0.4038551852
69	-0.52952	-0.424561212
70	-0.52952	-0.4276609296
71	-0.52952	-0.4244919108
72	-0.52952	-0.4281787243
73	-0.52952	-0.4255140557
74	-0.52952	-0.4291707672
75	-0.52952	-0.4413052833
76	-0.52952	-0.4357576628
77	-0.52952	-0.4398903991
78	-0.52952	-0.4376459041
79	-0.52952	-0.4434627664
80	-0.52952	-0.4494898276
81	-0.52952	-0.445895972
82	-0.52952	-0.4480489295
83	-0.52952	-0.448329137
84	-0.52952	-0.4582959119
85	-0.52952	-0.4535441657
86	-0.52952	-0.4562919823
87	-0.52952	-0.458321483
88	-0.52952	-0.4572042524
89	-0.52952	-0.4542379797
90	-0.52952	-0.4615284535
91	-0.52952	-0.4637654447
92	-0.52952	-0.4654706567
93	-0.52952	-0.4647630364
94	-0.52952	-0.4646672379
95	-0.52952	-0.4587078335
96	-0.52952	-0.4644073631
97	-0.52952	-0.458071464
98	-0.52952	-0.4648104928
99	-0.52952	-0.455790524
100	-0.52952	-0.4643548465

Iterations	Theoretical	Experimental
101	-0.52952	-0.4597123945
102	-0.52952	-0.4614880122
103	-0.52952	-0.4640232781
104	-0.52952	-0.4631510063
105	-0.52952	-0.4633038065
106	-0.52952	-0.4607956644
107	-0.52952	-0.4640445423
108	-0.52952	-0.4648928415
109	-0.52952	-0.464328522
109	-0.52952	-0.4654582708
110	-0.52952	-0.4653556353
111	-0.52952	-0.4637653123
112	-0.52952	-0.4627551564
113	-0.52952	-0.4616840943
114	-0.52952	-0.4639483874
115	-0.52952	-0.4658633096
116	-0.52952	-0.4652171301
117	-0.52952	-0.4662359862
118	-0.52952	-0.4657969867
119	-0.52952	-0.4654209807
120	-0.52952	-0.4644860719
121	-0.52952	-0.4655182454
122	-0.52952	-0.4639068364
123	-0.52952	-0.4650023139
124	-0.52952	-0.4634304145
125	-0.52952	-0.4670682595

TABLE II. For Cobyla on real Chip

Iterations	Theoretical	Experimental
1	-0.52952	-0.3242458015
2	-0.52952	-0.312141158
3	-0.52952	-0.3000890929
4	-0.52952	-0.2012450742
5	-0.52952	-0.3431368385
6	-0.52952	-0.3354519405
7	-0.52952	-0.3605707275
8	-0.52952	-0.3092321975
9	-0.52952	-0.3421126117
10	-0.52952	-0.382317455
11	-0.52952	-0.365973851
12	-0.52952	-0.3070352964
13	-0.52952	-0.378163368
14	-0.52952	-0.372308508
15	-0.52952	-0.3760271897
16	-0.52952	-0.2783118205
17	-0.52952	-0.3703690787
18	-0.52952	-0.3461754311
19	-0.52952	-0.3659191554
20	-0.52952	-0.3757416573
21	-0.52952	-0.3401756666
22	-0.52952	-0.3805747747
23	-0.52952	-0.3638238258
24	-0.52952	-0.3789247876
25	-0.52952	-0.3807448887
26	-0.52952	-0.3913352043
27	-0.52952	-0.3779779816
28	-0.52952	-0.3963448298
29	-0.52952	-0.3759803137
30	-0.52952	-0.3827794955
31	-0.52952	-0.3805431712
32	-0.52952	-0.3928037149
33	-0.52952	-0.385364926
34	-0.52952	-0.3918327581
35	-0.52952	-0.3848496156
36	-0.52952	-0.3892537216
37	-0.52952	-0.3883246723
38	-0.52952	-0.3732819491
39	-0.52952	-0.3917883664
40	-0.52952	-0.3926158442
41	-0.52952	-0.3891585442
42	-0.52952	-0.3859512631
43	-0.52952	-0.3947456284
44	-0.52952	-0.3852036904
45	-0.52952	-0.3942924663
46	-0.52952	-0.3859246281
47	-0.52952	-0.3890963958
48	-0.52952	-0.4008480231
49	-0.52952	-0.4013480612
50	-0.52952	-0.3944512178

Iterations	Theoretical	Experimental
51	-0.52952	-0.3884351222
52	-0.52952	-0.3884351222
53	-0.52952	-0.3884351222
54	-0.52952	-0.3907311105
55	-0.52952	-0.3980456027
56	-0.52952	-0.3906448114
57	-0.52952	-0.4059641886
58	-0.52952	-0.388795591
59	-0.52952	-0.3965149438
60	-0.52952	-0.4046238846
61	-0.52952	-0.4025665527
62	-0.52952	-0.3938826373
63	-0.52952	-0.3762519993
64	-0.52952	-0.395794006
65	-0.52952	-0.3963359515
66	-0.52952	-0.4041835107
67	-0.52952	-0.4079657663
68	-0.52952	-0.3994302984
69	-0.52952	-0.3969173201
70	-0.52952	-0.4045947654
71	-0.52952	-0.3955795003
72	-0.52952	-0.396464158
73	-0.52952	-0.3954576879
74	-0.52952	-0.3922859202
75	-0.52952	-0.3903440066
76	-0.52952	-0.3997641322
77	-0.52952	-0.3953649947
78	-0.52952	-0.3912133919
79	-0.52952	-0.4025246453
80	-0.52952	-0.4028140875
81	-0.52952	-0.3936148616
82	-0.52952	-0.4070786244
83	-0.52952	-0.3972980299
84	-0.52952	-0.401852009
85	-0.52952	-0.4013366986
86	-0.52952	-0.4078261972
87	-0.52952	-0.3935246526
88	-0.52952	-0.4035197528
89	-0.52952	-0.4009673514
90	-0.52952	-0.4068310896
91	-0.52952	-0.4011399496
92	-0.52952	-0.407983523
93	-0.52952	-0.4008302665
94	-0.52952	-0.4089342389
95	-0.52952	-0.404329474
96	-0.52952	-0.4098480158
97	-0.52952	-0.4068971478
98	-0.52952	-0.3970568892
99	-0.52952	-0.4050681684
100	-0.52952	-0.4176714243

Iterations	Theoretical	Experimental
101	-0.52952	-0.4127125846
102	-0.52952	-0.4028407225
103	-0.52952	-0.397501173
104	-0.52952	-0.4120374641
105	-0.52952	-0.4085077118
106	-0.52952	-0.4095649677
107	-0.52952	-0.4097617167
108	-0.52952	-0.4122036682
109	-0.52952	-0.4182641555