Quantum Phase Estimation, Iterative

To economize in qubits, the References below advocate using the so called iterative Quantum Phase Estimation (iterative qPE). Whereas the usual qPE uses multiple pointer qubits and gives the answer in one shot (passage through a single circuit), the iterative qPE uses only a single pointer qubit but requires passage through multiple circuits, with the parameters of each circuit depending on the final pointer measurement of the previous circuit. This works because the kickback phases which each power of U sends to the pointers in the nomal qPE are cummulative: the k'th pointer gets a kickback phase which includes the kickback phases accrued by all previous pointer qubits.

In this example, we use

$U = e^{i*rads*\sigma_Z}$

for some Real number $rads$ and we use initial state $|0\rangle$, so e^{i*rads} is the eigenvalue we seek.

Here are some of the equations used in the code below

```
for  k in range(num_reps):

    |           H
    |           exp(i*alpha(k)*sigz)
    U^(2^k)-----@
    |           H
    |           measure n(k) here
```

$\alpha(0) = n(0) = 0$

$\alpha(k+1) = 2\alpha(k) + \frac{\pi}{2}n(k)$

$\alpha(k) = \pi 2^{k-2} \sum_{b=0}^{k-1} \frac{n(b)}{2^b}$

$rads = \frac{\alpha(num\_reps-1)}{2^{num\_reps-2}}$

References

1. https://arxiv.org/abs/1512.06860 (https://arxiv.org/abs/1512.06860) by Google team
2. https://arxiv.org/abs/1605.03590 (https://arxiv.org/abs/1605.03590) by Microsoft team

First change your working directory to the qubiter directory in your computer, and add its path to the path environment variable.

In [1]:
```
import os
import sys
print(os.getcwd())
os.chdir('../')
print(os.getcwd())
sys.path.insert(0,os.getcwd())
```

```
/home/bram/workspace/python/qubiter-master2/qubiter-master/jupyter-notebooks
/home/bram/workspace/python/qubiter-master2/qubiter-master
```

In [2]:
```
from SEO_writer import *
from SEO_simulator import *
import numpy as np
```

In [3]:
```
rads = 2*np.pi*(1/16 + 1/8 + 1e-8)
z_axis = 3
num_bits = 8

file_prefix = 'chemistry/chem_io_folder/H2_ground_state'

emb = CktEmbedder(num_bits, num_bits)

alpha = 0
ptr_state = 0
ptr_st_list = []


# simulate circuit
init_st_vec = SEO_simulator.get_standard_basis_st([0]*num_bits)
sim = SEO_simulator(file_prefix, num_bits, init_st_vec)
sim.describe_fin_st(print_st_vec=True, do_pp=True, omit_zero_amps=True, show_probs=True)
# find final state of pointer qubit
fin_st_vec = sim.cur_st_vec_list[0]
# dictionary with key=qubit, value=final (P(0), P(1))
bit_to_probs = sim.get_bit_probs(fin_st_vec)
p0, p1 = bit_to_probs[0]

#print('ptr_0=', p0)
print('---------------------')
print('timeline of bit 0 measurements', ptr_st_list)
#print("rads, alpha(num_reps-1)/2^(num_reps-2)", rads, alpha/(1 << num_reps-2))
```

```
final state vector
ZF convention (Zero bit First in state tuple)
(0, 0, 0, 0, 0, 0, 0, 0)ZF (0.0583963151708+0.0450825037509j) , prob= 0.00544256176998
(0, 0, 1, 0, 0, 0, 0, 0)ZF (-0.136810489501+0.177213504106j) , prob= 0.0501217360752
(0, 1, 0, 0, 0, 0, 0, 0)ZF (0.0100139594435+0.0778314085473j) , prob= 0.00615800754018
(0, 1, 1, 0, 0, 0, 0, 0)ZF (0.154336626428-0.0198572882919j) , prob= 0.0242141061553
(1, 0, 0, 0, 0, 0, 0, 0)ZF (0.0349518232873+0.0609515415675j) , prob= 0.00493672037056
(1, 0, 1, 0, 0, 0, 0, 0)ZF (0.81795239116-0.469043549975j) , prob= 0.889047965977
(1, 1, 0, 0, 0, 0, 0, 0)ZF (-0.0280931411744+0.103625229279j) , prob= 0.0115274127241
(1, 1, 1, 0, 0, 0, 0, 0)ZF (0.0892525151865+0.024196651016j) , prob= 0.00855148938751
total probability of final state vector (=one if no measurements)= 1.0
dictionary with key=qubit, value=final (P(0), P(1))
```

```
{0: (0.085936411540670235, 0.91406358845932978),
 1: (0.94954898419283329, 0.05045101580716671),
 2: (0.028064702404855722, 0.9719352975951443),
 3: (1.0, 0.0),
 4: (1.0, 0.0),
 5: (1.0, 0.0),
 6: (1.0, 0.0),
 7: (1.0, 0.0)}
---------------------
timeline of bit 0 measurements []
```

In [ ]: