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
num_reps = 15
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, 0])
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]
if p0 > p1:
    ptr_state = 0
else:
    ptr_state = 1
ptr_st_list.append(ptr_state)
print('ptr_state=', ptr_state)
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))
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-3-c0c1d6128eff> in <module>()
     14 # simulate circuit
     15 init_st_vec = SEO_simulator.get_standard_basis_st([0, 0])
---> 16 sim = SEO_simulator(file_prefix, num_bits, init_st_vec)
     17 sim.describe_fin_st(print_st_vec=True, do_pp=True, omit_zero_amps=True, show_probs=True)
     18

/home/bram/workspace/python/qubiter-master2/qubiter-master/SEO_simulator.py in __init__(self, file_prefix, n
um_bits, init_st_vec, verbose)
    107         self.verbose = verbose
    108 
--> 109         SEO_reader.__init__(self, file_prefix, num_bits, verbose)
    110 
    111     @staticmethod

/home/bram/workspace/python/qubiter-master2/qubiter-master/SEO_reader.py in __init__(self, file_prefix, num_
bits, verbose)
     87 
     88         while not self.english_in.closed:
---> 89             self.next_line()
     90 
     91         self.write_log()

/home/bram/workspace/python/qubiter-master2/qubiter-master/SEO_reader.py in next_line(self)
    192             tar_bit_pos = int(self.split_line[2])
    193             controls = self.read_TF_controls(self.split_line[4:])
--> 194             self.use_HAD2(tar_bit_pos, controls)
    195 
    196         elif line_name == "LOOP":

/home/bram/workspace/python/qubiter-master2/qubiter-master/SEO_simulator.py in use_HAD2(self, tar_bit_pos, c
ontrols)
    522         """
    523         gate = OneBitGates.had2()
--> 524         self.evolve_by_controlled_one_bit_gate(tar_bit_pos, controls, gate)
    525 
    526     def use_MEAS(self, tar_bit_pos, kind):

/home/bram/workspace/python/qubiter-master2/qubiter-master/SEO_simulator.py in evolve_by_controlled_one_bit_
gate(self, tar_bit_pos, controls, one_bit_gate)
    457         # br = branch
    458         for br in range(len(self.cur_st_vec_list)):
--> 459             vec = self.cur_st_vec_list[br][vec_slicex]
    460             # Axes 1 of one_bit_gate and new_tar of vec are summed over. Axis
    461             #  0 of one_bit_gate goes to the front of all the axes of new vec.
```

```
IndexError: too many indices for array
```

In [ ]: