QHO_5

February 21, 2021

[ ]:

```
from qiskit import*
from qiskit.tools.visualization import*
from numpy import*
from matplotlib.pyplot import*
```

[2]:

```
qr=QuantumRegister(2,name='qr')
anci=QuantumRegister(3,name='anci')
cr=ClassicalRegister(2,name='cr')
```

[3]:

```
def circuit(a,qc):
    qc.cx(qr[0],anci[0])
    qc.x(anci[0])
    qc.cu(a,-pi/2,pi/2,0,anci[0],qr[0])
    qc.x(anci[0])
    qc.cx(anci[0],qr[0])
    qc.cx(qr[0],anci[2])
    qc.x(anci[2])
    qc.h(anci[2])
    qc.h(qr[1])
    qc.cx(qr[1],anci[2])
    qc.h(qr[1])
    qc.h(anci[2])
    qc.x(anci[2])
    qc.x(anci[0])
    qc.cu(a,-pi/2,pi/2,0,anci[0],qr[1])
    qc.x(anci[0])
    qc.cx(qr[0],qr[1])
    qc.h(anci[2])
    qc.cx(anci[2],qr[1])
    qc.cx(anci[2],qr[0])
    qc.h(anci[2])
    qc.ch(anci[2],qr[0])
    qc.barrier()
    qc.measure(qr[0],cr[0])
    qc.measure(qr[1],cr[1])
```

[4]:

```
t=arange(0,0.005,0.001)
w=1
phi=0
m=1
def f(t):
    A=1
    return((A*cos(w*t + phi))/sqrt(2*m))
```

[5]:

```
p=[]
q=[]
r=[]
s=[]
theta=[b*f(b) for b in t]
a0=theta[0]
a1=theta[1]
a2=theta[2]
a3=theta[3]
a4=theta[4]
```

[4]:

```
IBMQ.load_account()
provider=IBMQ.get_provider(hub='ibm-q')
backend=provider.get_backend('ibmq_quito')
```

```
/home/nav/anaconda3/lib/python3.8/site-
packages/qiskit/providers/ibmq/ibmqfactory.py:192: UserWarning: Timestamps in
IBMQ backend properties, jobs, and job results are all now in local time instead
of UTC.
  warnings.warn('Timestamps in IBMQ backend properties, jobs, and job results '
```

[7]:

```
qc1=QuantumCircuit(qr,anci,cr)
circuit(a0,qc1)
job_exp = execute(qc1, backend=backend, shots=1024)

from qiskit.tools.monitor import job_monitor
job_monitor(job_exp)
```

```
Job Status: job has successfully run
```

[8]:

```
exp_result = job_exp.result()
counts = exp_result.get_counts(qc1)
p.append(counts['00']/1024)
q.append(counts['01']/1024)
s.append(counts['10']/1024)
r.append(counts['11']/1024)
```

[9]:

```
qc2=QuantumCircuit(qr,anci,cr)
circuit(a1,qc2)
```

```
job_exp = execute(qc2, backend=backend, shots=1024)

from qiskit.tools.monitor import job_monitor
job_monitor(job_exp)
```

```
Job Status: job has successfully run
```

[10]:

```
exp_result = job_exp.result()
counts = exp_result.get_counts(qc2)
p.append(counts['00']/1024)
q.append(counts['01']/1024)
s.append(counts['10']/1024)
r.append(counts['11']/1024)
```

[11]:

```
qc3=QuantumCircuit(qr,anci,cr)
circuit(a2,qc3)
job_exp = execute(qc3, backend=backend, shots=1024)

from qiskit.tools.monitor import job_monitor
job_monitor(job_exp)
```

```
Job Status: job has successfully run
```

[12]:

```
exp_result = job_exp.result()
counts = exp_result.get_counts(qc3)
p.append(counts['00']/1024)
q.append(counts['01']/1024)
s.append(counts['10']/1024)
r.append(counts['11']/1024)
```

[13]:

```
qc4=QuantumCircuit(qr,anci,cr)
circuit(a3,qc4)
job_exp = execute(qc4, backend=backend, shots=1024)

from qiskit.tools.monitor import job_monitor
job_monitor(job_exp)
```

```
Job Status: job has successfully run
```

[14]:

```
exp_result = job_exp.result()
counts = exp_result.get_counts(qc4)
p.append(counts['00']/1024)
q.append(counts['01']/1024)
s.append(counts['10']/1024)
r.append(counts['11']/1024)
```

[15]:

```
qc5=QuantumCircuit(qr,anci,cr)
circuit(a4,qc5)
job_exp = execute(qc5, backend=backend, shots=1024)

from qiskit.tools.monitor import job_monitor
job_monitor(job_exp)
```

```
Job Status: job has successfully run
```

[16]:

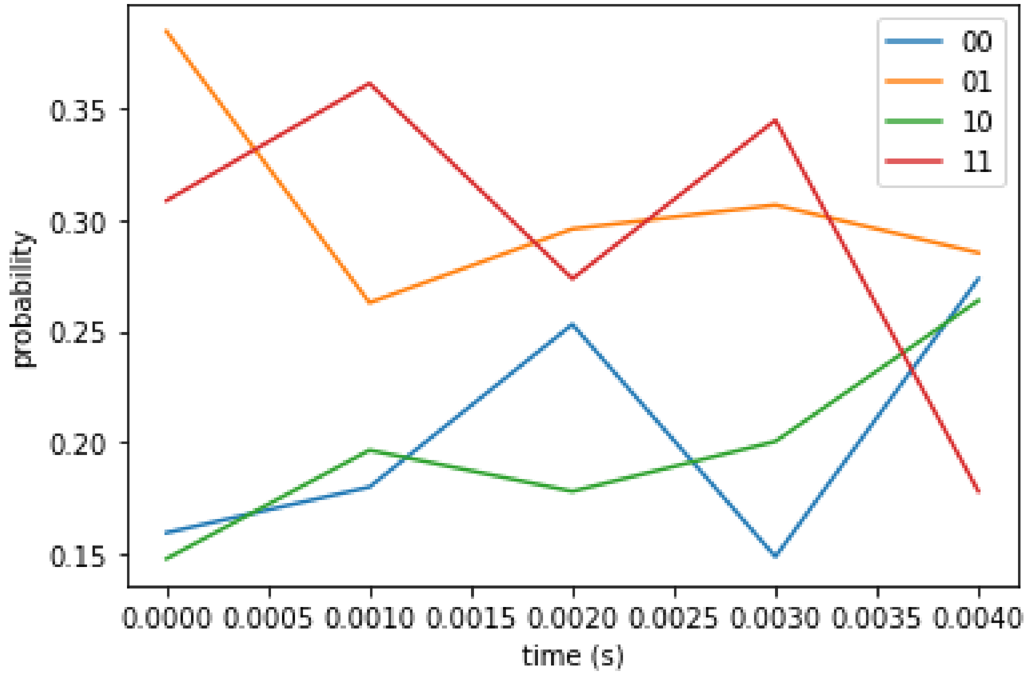
```
exp_result = job_exp.result()
counts = exp_result.get_counts(qc5)
p.append(counts['00']/1024)
q.append(counts['01']/1024)
s.append(counts['10']/1024)
r.append(counts['11']/1024)
```

[17]:

```
plot(t,p,label='00')
plot(t,q,label='01')
plot(t,r,label='10')
plot(t,s,label='11')
xlabel('time (s)')
ylabel('probability')
legend()
```

[17]:

```
<matplotlib.legend.Legend at 0x7f47eb896820>
```



[18]:

```
t=arange(0,0.005,0.0005)
w=1
phi=0
m=1
def f(t):
    A=1
    return((A*cos(w*t + phi))/sqrt(2*m))

p=[]
q=[]
r=[]
s=[]
theta=[b*f(b) for b in t]
qc=QuantumCircuit(qr,anci,cr)
```

[20]:

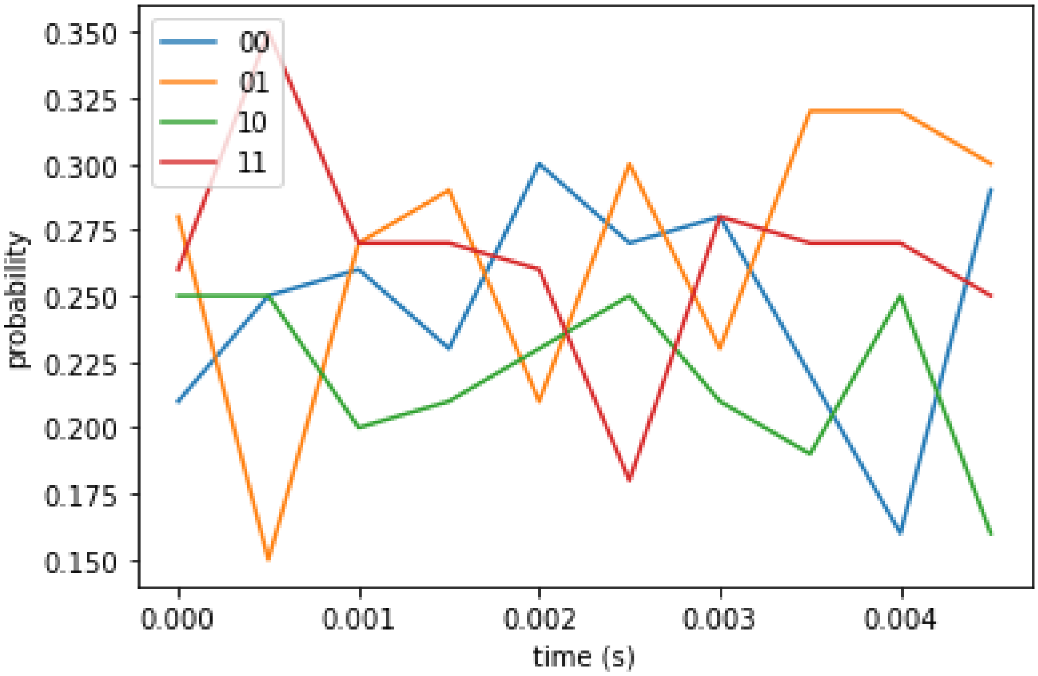
```
for a in theta:
    circuit(a,qc)
    job=execute(qc,backend,shots=100)
    results=job.result()
    counts=results.get_counts()
    p.append(counts['00']/100)
    q.append(counts['01']/100)
    s.append(counts['10']/100)
    r.append(counts['11']/100)
    qc.reset(qr)
    qc.reset(anci)
```

[21]:

```
plot(t,p,label='00')
plot(t,q,label='01')
plot(t,r,label='10')
plot(t,s,label='11')
xlabel('time (s)')
ylabel('probability')
legend()
```

[21]:

```
<matplotlib.legend.Legend at 0x7f47eb7abf40>
```



[5]:

```
t=arange(0,0.005,0.0001)
w=1
phi=0
m=1
def f(t):
    A=1
    return((A*cos(w*t + phi))/sqrt(2*m))

p=[]
q=[]
r=[]
s=[]
theta=[b*f(b) for b in t]
qc=QuantumCircuit(qr,anci,cr)
```

[ ]:

```
for a in theta:
    circuit(a,qc)
    job=execute(qc,backend,shots=100)
    results=job.result()
    counts=results.get_counts()
    p.append(counts['00']/100)
    q.append(counts['01']/100)
    s.append(counts['10']/100)
    r.append(counts['11']/100)
    qc.reset(qr)
    qc.reset(anci)
```

[ ]:

```
plot(t,p,label='00')
plot(t,q,label='01')
plot(t,r,label='10')
plot(t,s,label='11')
xlabel('time (s)')
ylabel('probability')
legend()
```

[ ]:

```

```