Physics 245 Homework #7

Nathan Solomon

November 26, 2024

Problem 0.1.

See the Jupyter notebook at the end of this doc for the graphs. For parts (a) and (b), since g is small, the Jaynes-Cummings Hamitlonian is almost exactly correct. In part (d), g is higher, so the time evolution predicted by the JC Hamiltonian deviates a bit more from the actual time evolution. For part (c), that deviation is greater.

Problem 0.2.

This Hamiltonian predicts the states will evolve very little over one period of oscillation. That's also exactly what we would expect from the Hamiltonian we derived in class:

$$H = -\frac{g^2\hbar}{4\delta}\left(\sigma_+ \otimes \sigma_- \otimes I + \sigma_- \otimes \sigma_+ \otimes I\right).$$

Problem 0.3.

(a)

$$\begin{aligned}
\frac{H_{eff}}{\hbar} &= \frac{g}{2}\sigma_x \otimes \sigma_x = \frac{g}{2} \cdot \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \\
U(t) &= \exp\left(-\frac{iH_{eff}t}{\hbar}\right) \\
&= \sum_{n=0}^{\infty}\left(\frac{(-itg/2)^{2n+1}}{(2n+1)!}\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\right) + \sum_{n=0}^{\infty}\frac{(-itg/2)^{2n}}{(2n)!} \cdot I \\
&= \cos\left(\frac{tg}{2}\right) I \otimes I - i\sin\left(\frac{tg}{2}\right)\sigma_x \otimes \sigma_x
\end{aligned}$$

(b)

$$[RY(-v\pi/2)\otimes I]\cdot[RX(-s\pi/2)\otimes RX(-vs\pi/2)]\cdot[XX(s\pi/4)]\cdot[RY(v\pi/2)\otimes I]$$

$$= \left(\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}\right)\left(\frac{1}{2}\begin{bmatrix} 1 & i & i & -1 \\ i & 1 & -1 & i \\ i & -1 & 1 & i \\ -1 & i & i & 1 \end{bmatrix}\right)\left(\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & 0 & -i \\ 0 & 1 & -i & 0 \\ 0 & -i & 1 & 0 \\ -i & 0 & 0 & 1 \end{bmatrix}\right)\left(\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}\right)$$

$$= \frac{1}{4\sqrt{2}}\begin{bmatrix} 1+i & -1+i & 1+i & -1+i \\ -1+i & 1+i & -1+i & 1+i \\ -1+i & -1-i & 1-i & 1+i \\ -1-i & -1+i & 1+i & 1-i \end{bmatrix}\begin{bmatrix} 1 & -i & -1 & -i \\ -i & 1 & -i & -1 \\ 1 & -i & 1 & i \\ -i & 1 & i & 1 \end{bmatrix}$$

$$= \frac{1}{4\sqrt{2}}\begin{bmatrix} 4+4i & 0 & 0 & 0 \\ 0 & 4+4i & 0 & 0 \\ 0 & 0 & 0 & 4+4i \\ 0 & 0 & 4+4i & 0 \end{bmatrix}$$

$$= e^{i\pi/4}\cdot CNOT$$

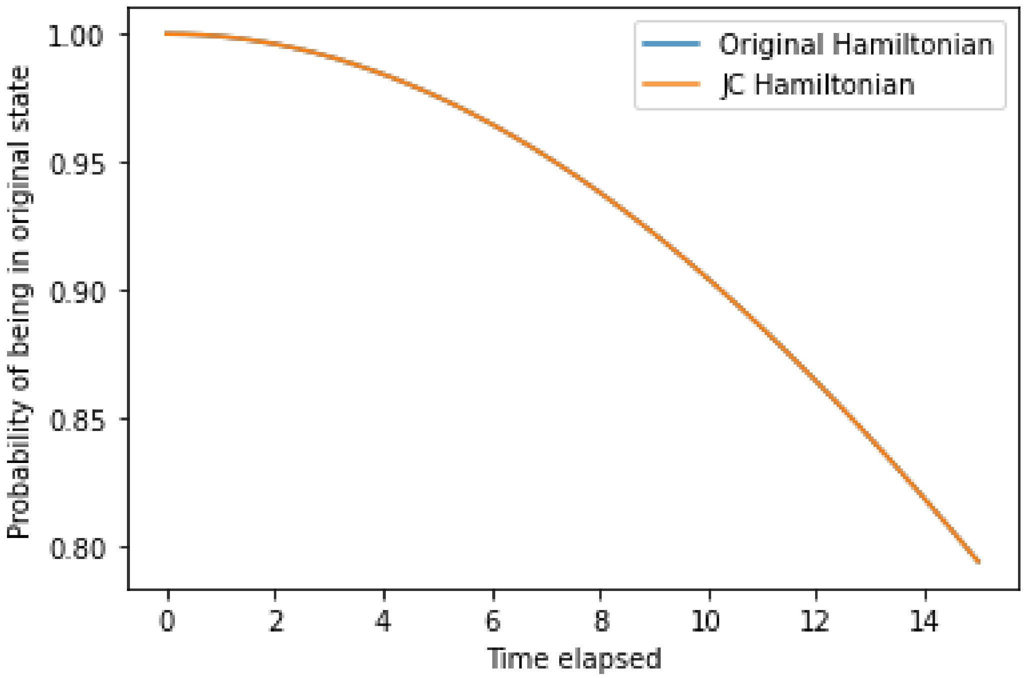
So up to a global phase, this method is equivalent to a CNOT gate when $s\pi/4 = \chi = tg/2$, so $gt = \pi/2$.

notebook

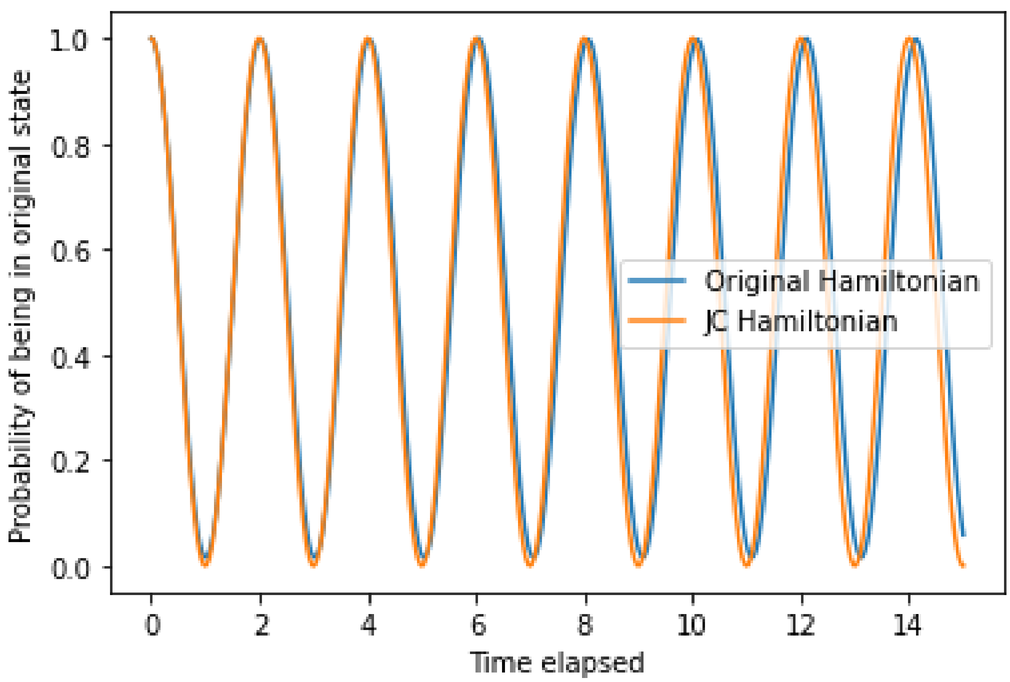
November 26, 2024

[1]:
```python
import qutip as qt
import matplotlib.pyplot as plt
import numpy as np
```

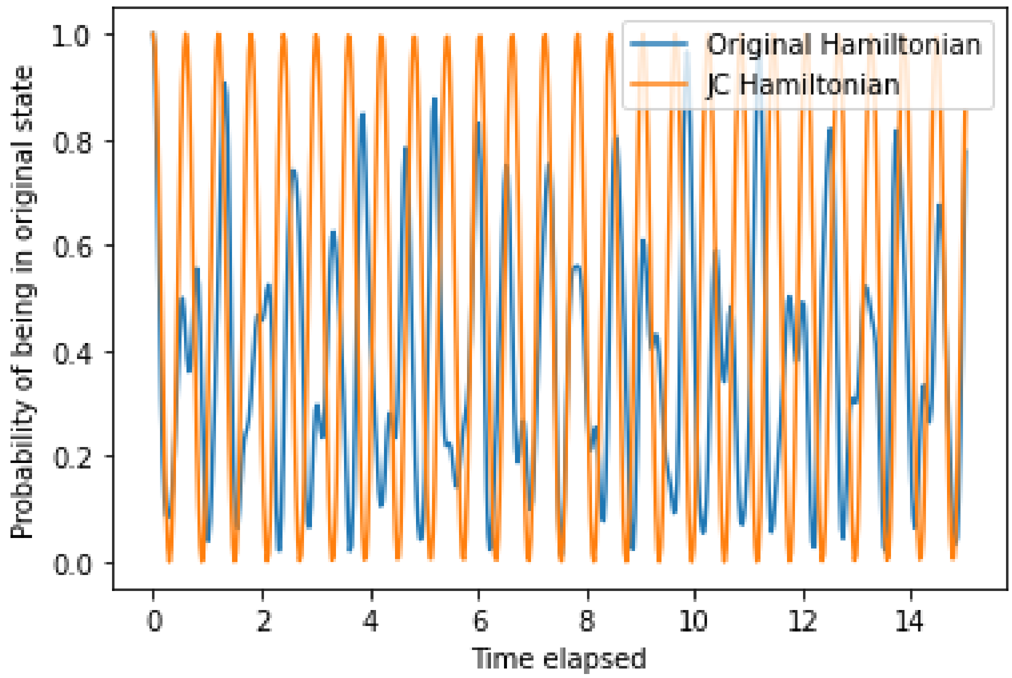
[2]:
```python
# Problem 1
N = 12

def H(g):
    return np.pi * qt.tensor(qt.sigmaz(), qt.qeye(N)) + \
           np.pi * qt.tensor(qt.qeye(2), (2 * qt.num(N) + qt.qeye(N))) + \
           (g / 2) * qt.tensor(qt.sigmax(), qt.destroy(N) + qt.create(N))

def U(t, n, g):
    Omega = g * np.sqrt(n + 1)
    return np.cos(Omega * t / 2) * qt.qeye(2) - 1j * np.sin(Omega * t / 2) * qt.
↪sigmax()

for (n, g) in zip([0, 0, 10, 0], [np.pi / 50, np.pi, np.pi, 2 * np.pi]):
    times = np.linspace(0, 15, 500)

    Psi = qt.tensor(qt.basis(2, 0), qt.basis(N, n))
    states = [(-1j * t * H(g)).expm() * Psi for t in times]
    probs = [qt.expect(Psi.proj(), s) for s in states]
    plt.plot(times, probs, label="Original Hamiltonian")

    Psi = qt.basis(2, 1)
    states = [U(t, n, g) * Psi for t in times]
    probs = [qt.expect(Psi.proj(), s) for s in states]
    plt.plot(times, probs, label="JC Hamiltonian")


    plt.ylabel("Probability of being in original state")
    plt.xlabel("Time elapsed")
    plt.legend()
    plt.show()
```

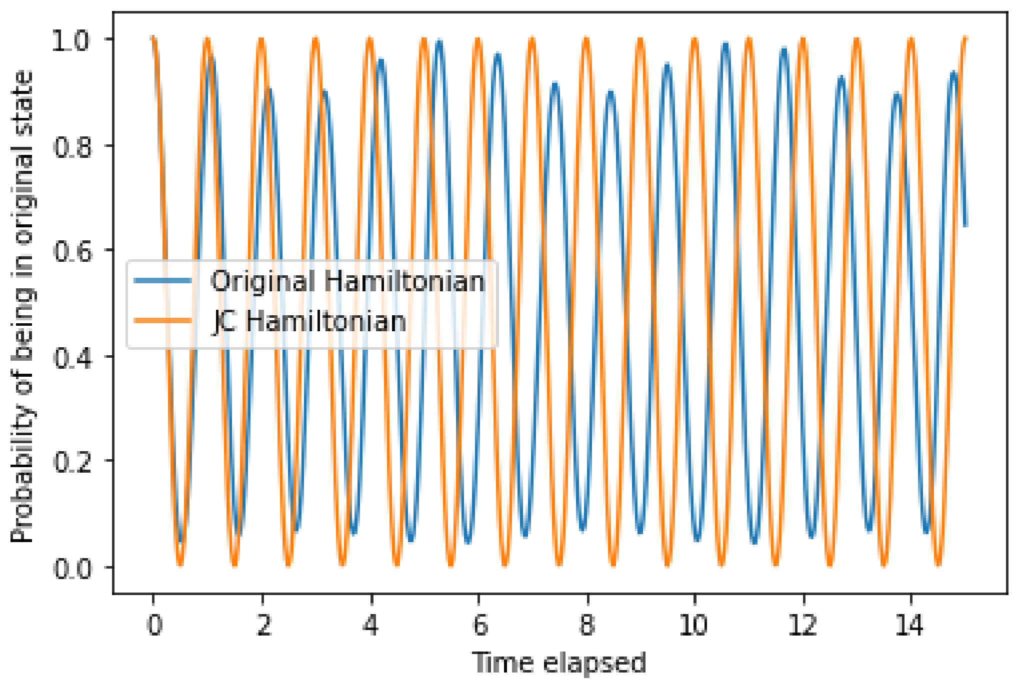


Original Hamiltonian
JC Hamiltonian
Probability of being in original state
Time elapsed
1.00
0.95
0.90
0.85
0.80
0
2
4
6
8
10
12
14



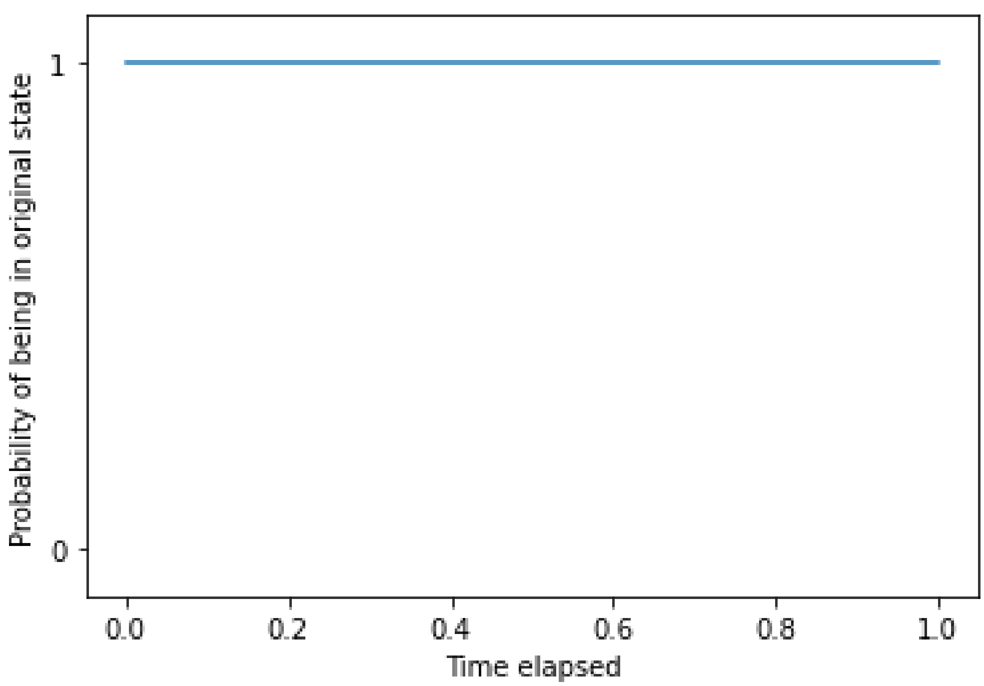
Original Hamiltonian
JC Hamiltonian
Probability of being in original state
Time elapsed
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
12
14

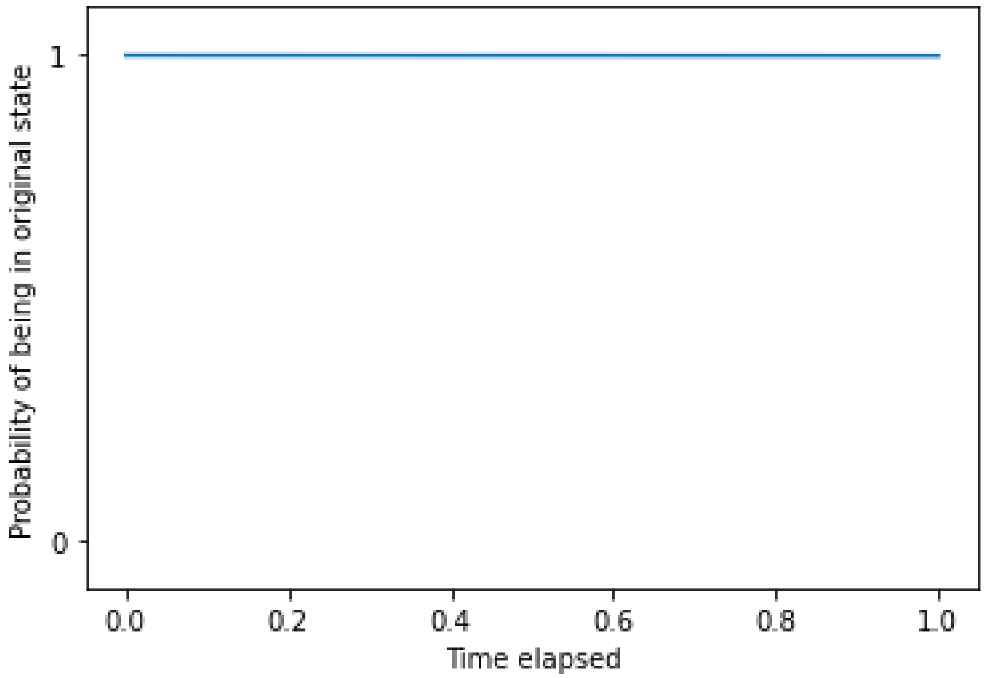


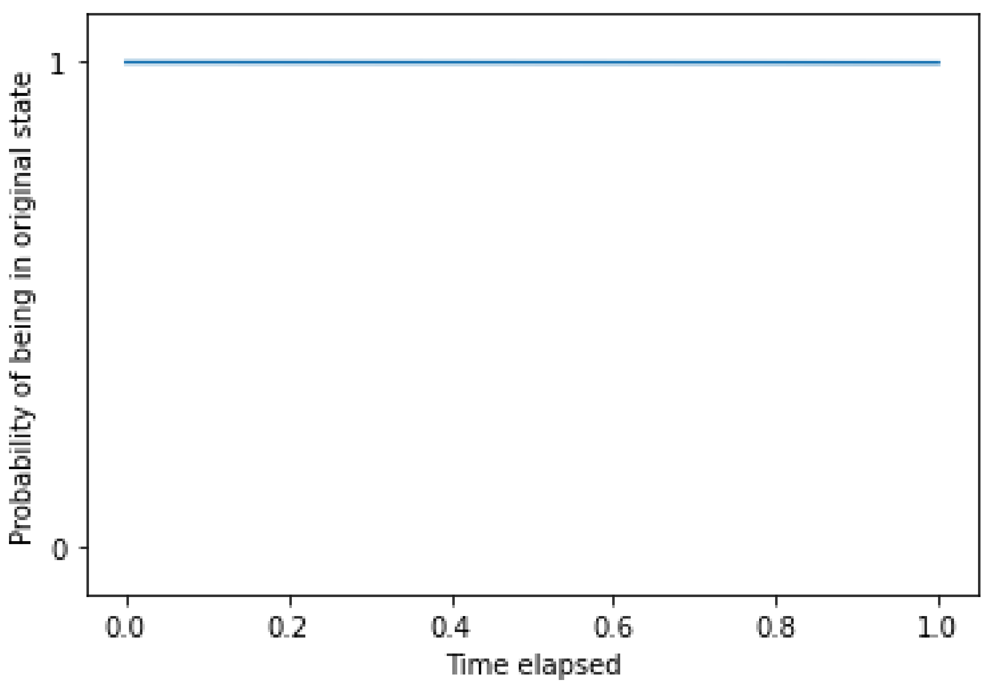


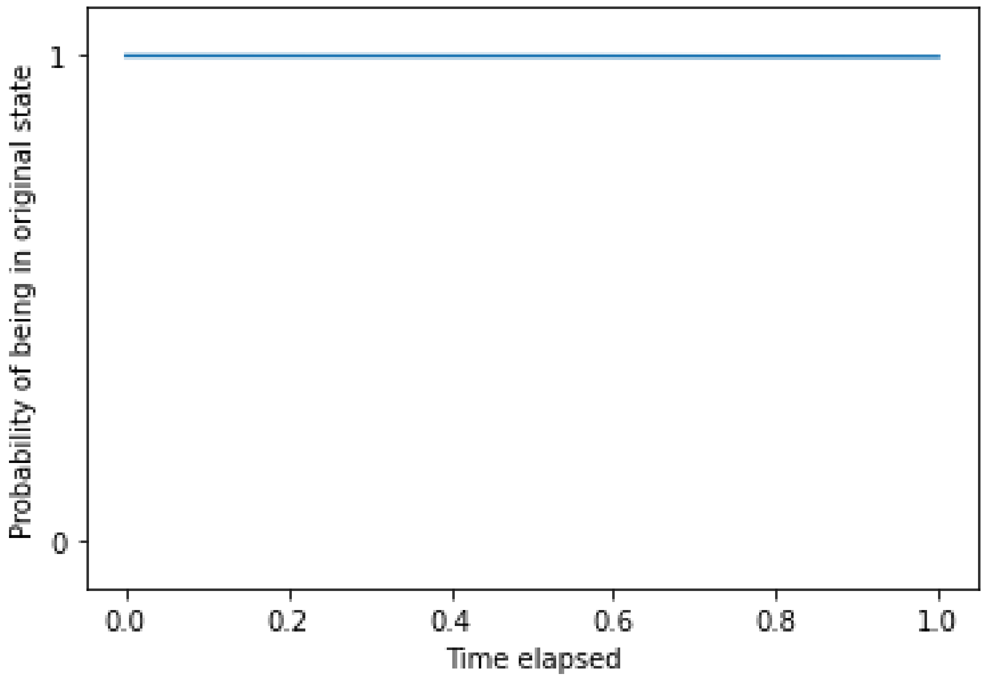
[3]:
```
# Problem 2
N = 10
```

```
original_states = [
    qt.ket("110", [2, 2, N]),
    qt.ket("010", [2, 2, N]),
    qt.ket("100", [2, 2, N]),
    qt.ket("000", [2, 2, N]),
    (qt.ket("010", [2, 2, N]) + qt.ket("100", [2, 2, N])).unit(),
    qt.ket("019", [2, 2, N])
]
H = np.pi * qt.tensor(qt.sigmaz(), qt.qeye(2), qt.qeye(N)) + \
    np.pi * qt.tensor(qt.qeye(2), qt.sigmaz(), qt.qeye(N)) + \
    np.pi * qt.tensor(qt.qeye(2), qt.qeye(2), (2 * qt.num(N) + qt.qeye(N))) + \
    (np.pi / 100) * qt.tensor(qt.sigmap(), qt.qeye(2), qt.destroy(N)) + \
    (np.pi / 100) * qt.tensor(qt.qeye(2), qt.sigmap(), qt.destroy(N)) + \
    (np.pi / 100) * qt.tensor(qt.sigmam(), qt.qeye(2), qt.create(N)) + \
    (np.pi / 100) * qt.tensor(qt.qeye(2), qt.sigmam(), qt.create(N))

for Psi in original_states:
    times = np.linspace(0, 1, 200)
    states = [(-1j * t * H).expm() * Psi for t in times]
    probs = [qt.expect(Psi.proj(), s) for s in states]
    plt.plot(times, probs)
    plt.ylim(-.1, 1.1)
    plt.yticks([0, 1])
    plt.ylabel("Probability of being in original state")
    plt.xlabel("Time elapsed")
    plt.show()
```

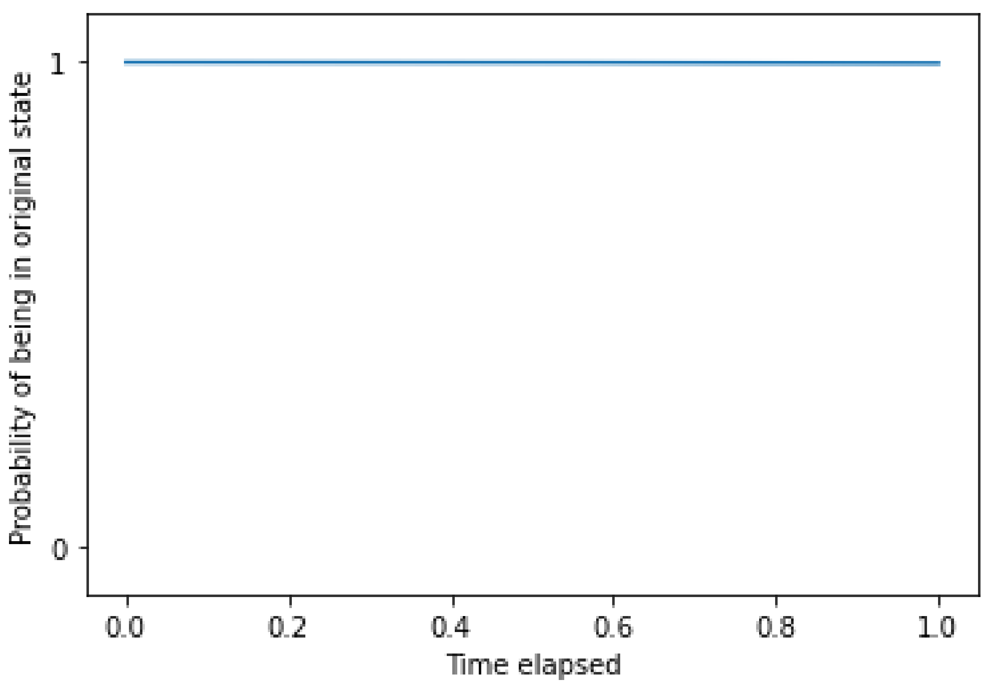



1
0
Probability of being in original state
0.0
0.2
0.4
0.6
0.8
1.0
Time elapsed

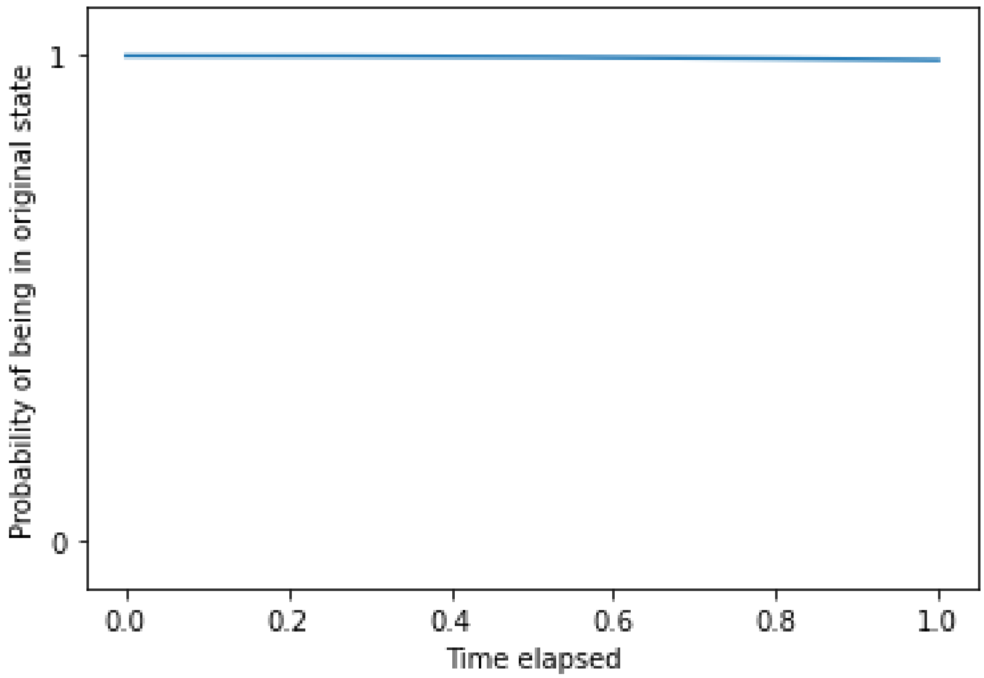

1
0
Probability of being in original state
0.0
0.2
0.4
0.6
0.8
1.0
Time elapsed



1
0
Probability of being in original state
0.0
0.2
0.4
0.6
0.8
1.0
Time elapsed



1
0
Probability of being in original state
0.0
0.2
0.4
0.6
0.8
1.0
Time elapsed



[6]:
```
# Problem 3
def XX(chi):
    return np.cos(chi) * qt.tensor(qt.qeye(2), qt.qeye(2)) - \
      1j * np.sin(chi) * qt.tensor(qt.sigmax(), qt.sigmax())

s = 1
v = 1

def RX(phi):
    return (-1j * phi * qt.sigmax() / 2).expm()
def RY(phi):
    return (-1j * phi * qt.sigmay() / 2).expm()

cnot = qt.tensor(RY(v * np.pi / 2), qt.qeye(2))
cnot = XX(s * np.pi / 4) * cnot
cnot = qt.tensor(RX(-1 * s * np.pi / 2), RX(-1 * v * s * np.pi / 2)) * cnot
cnot = qt.tensor(RY(-1 * v * np.pi / 2), qt.qeye(2)) * cnot

print(cnot)
```

```
Quantum object: dims=[[2, 2], [2, 2]], shape=(4, 4), type='oper', dtype=Dense,
isherm=False
Qobj data =
[[ 7.07106781e-01+7.07106781e-01j  1.23259516e-32-1.96261557e-17j
   7.85046229e-17+9.81307787e-17j -7.85046229e-17-5.55111512e-17j]
```

```
[ 3.92523115e-17-9.81307787e-17j  7.07106781e-01+7.07106781e-01j
 -1.17756934e-16-5.55111512e-17j  7.85046229e-17+9.81307787e-17j]
[ 1.11022302e-16+4.87765193e-17j -1.57009246e-16-1.37383090e-16j
  1.57009246e-16-1.37383090e-16j  7.07106781e-01+7.07106781e-01j]
[-1.57009246e-16-1.37383090e-16j  7.17699910e-17+9.52420783e-18j
  7.07106781e-01+7.07106781e-01j  2.35513869e-16-1.76635402e-16j]]
```

[5]:
```
np.matrix([
    [1+1j, -1+1j, 1+1j, -1+1j],
    [-1+1j, 1+1j, -1+1j, 1+1j],
    [-1+1j, -1-1j, 1-1j, 1+1j],
    [-1-1j, -1+1j, 1+1j, 1-1j]
]) @ np.matrix([
    [1, -1j, -1, -1j],
    [-1j, 1, -1j, -1],
    [1, -1j, 1, 1j],
    [-1j, 1, 1j, 1],
])
```

[5]:
```
matrix([[4.+4.j, 0.+0.j, 0.+0.j, 0.+0.j],
        [0.+0.j, 4.+4.j, 0.+0.j, 0.+0.j],
        [0.+0.j, 0.+0.j, 0.+0.j, 4.+4.j],
        [0.+0.j, 0.+0.j, 4.+4.j, 0.+0.j]])
```

[ ]:

Phys 245 Quantum Computation
Homework 7

1. *[60] RWA too!* In class we found the time evolution operator under the Jaynes-Cummings Hamiltonian to be:

$$\hat{U} = \begin{pmatrix} \cos\left(\frac{\Omega' t}{2}\right) - i\frac{\delta}{\Omega'}\sin\left(\frac{\Omega' t}{2}\right) & -i\frac{\Omega}{\Omega'}\sin\left(\frac{\Omega' t}{2}\right) \\ -i\frac{\Omega}{\Omega'}\sin\left(\frac{\Omega' t}{2}\right) & \cos\left(\frac{\Omega' t}{2}\right) + i\frac{\delta}{\Omega'}\sin\left(\frac{\Omega' t}{2}\right) \end{pmatrix}$$

on the basis of $|\psi\rangle = a|1, n+1\rangle + b|0, n\rangle$, where the first label in the ket is the qubit state and the second the harmonic oscillator state. Here $\Omega = \frac{g}{2}\sqrt{n+1}$, $\delta = \omega - \omega_o$, and $\Omega' = \sqrt{\Omega^2 + \delta^2}$. When deriving this time-evolution operator, we originally started with the Hamiltonian

$$\frac{H}{\hbar} = \frac{\omega_o}{2}\hat{\sigma}_z + \omega\left(\hat{a}^\dagger\,\hat{a} + \frac{1}{2}\right) + \frac{g}{2}\hat{\sigma}_x(\hat{a} + \hat{a}^\dagger)$$

before using the RWA to drop the counter rotating terms. Use a numerical solver (e.g QuTIP) to find the evolution under this original Hamiltonian and compare it to that predicted by our time evolution operator for the cases:

   a. [15] $|\psi(t=0)\rangle = |0{,}0\rangle$, $\omega = \omega_o = 2\pi$ and $g = \omega/100$
   b. [15] $|\psi(t=0)\rangle = |0{,}0\rangle$, $\omega = \omega_o = 2\pi$ and $g = \frac{\omega}{2}$
   c. [15] $|\psi(t=0)\rangle = |0, 10\rangle$, $\omega = \omega_o = 2\pi$ and $g = \frac{\omega}{2}$
   d. [15] $|\psi(t=0)\rangle = |00\rangle$, $\omega = \omega_o = 2\pi$ and $g = \omega$

2. [60] *Jayne says too!* Suppose that two qubits coupled to the same harmonic oscillator can described by the Hamiltonian:

$$\frac{H}{\hbar} = \frac{\omega_o}{2}\hat{\sigma}_z^{(1)} + \frac{\omega_o}{2}\hat{\sigma}_z^{(2)} + \omega\left(\hat{a}^\dagger\,\hat{a} + \frac{1}{2}\right) + \frac{g}{2}\left((\hat{\sigma}_+^{(1)} + \hat{\sigma}_+^{(2)})\hat{a} + (\hat{\sigma}_-^{(1)} + \hat{\sigma}_-^{(2)})\hat{a}^\dagger\right)$$

Implement this Hamiltonian in QuTip and use it to find the time dynamics of the following situations. For simplicity assume $\omega_o = 2\pi$, and $g = \omega_o/100$. Since you cannot use an infinite dimensioned Hilbert space in QuTip you'll need to truncate the basis at some maximum $|n\rangle$. Make sure and choose that maximum n large enough to not affect your answer. For $\omega = 2\omega_o$ find the time evolution of the population in the initial state for the following initial states over one period of oscillation:

   i. [10] $|\psi(t=0)\rangle = |110\rangle$ -- for this problem our ordering in the ket is $|qubit\ 1, qubit\ 2, QHO\rangle$.
   ii. [10] $|\psi(t=0)\rangle = |010\rangle$
   iii. [10] $|\psi(t=0)\rangle = |100\rangle$
   iv. [10] $|\psi(t=0)\rangle = |000\rangle$
   v. [10] $|\psi(t=0)\rangle = \frac{1}{\sqrt{2}}(|010\rangle + |100\rangle)$
   vi. [10] $|\psi(t=0)\rangle = |019\rangle$
   vii. [10] Compare your answer in part (vi) to the analytical expression we derived in class. Comment on similarities and differences.

3. *[50] Quantum computing runs on effective Hamiltonians.* As we saw in class, two qubits can be coupled by a harmonic oscillator to produce an effective Hamiltonian that reproduces their evolution but does not contain the harmonic oscillator. For example, we saw that two qubits coupled by a harmonic oscillator on resonance with the qubit transitions produced an effective Hamiltonian of the form
$$\frac{\hat{H}_{eff}}{\hbar} = \frac{g}{4}\left(\hat{\sigma}_x^{(1)}\hat{\sigma}_x^{(2)} + \hat{\sigma}_y^{(1)}\hat{\sigma}_y^{(2)}\right).$$
It turns out that using very similar schemes these system can be engineered to produce other effective Hamiltonians. In this problem we will explore the time-evolution due to these effective Hamiltonians. Suppose an effective Hamiltonian is created between two qubits of the form
$$\frac{\hat{H}_{eff}}{\hbar} = \frac{g}{2}\hat{\sigma}_x^{(1)}\hat{\sigma}_x^{(2)}.$$
   a. [20] Calculate the time evolution operator of this Hamiltonian.

   b. [30] A controlled-NOT gate can be constructed from an XX gate by sandwiching it between some single qubit rotations. The paper:
   D. Maslov, *New J. Phys.* **19** 023035 (2017)
   shows one way of doing this in Figure 1 – incidentally, this is how IonQ implements a CNOT gate on their hardware. Show that this prescription indeed produces a CNOT gate up to a global phase. A few tips:
      1. Remember that to apply the unitaries from left to right means that you operate first with the leftmost operator.
      2. Choose $\frac{gt}{2}$ to reproduce the needed angle χ for the XX gate – χ is defined in the text above Fig. 1.
      3. Remember that to apply e.g. a Y rotation to just qubit 1 means you are also applying the identity to qubit 2 and therefore the appropriate operator is the tensor produce of the Y rotation and the identity operator.