notebook

November 19, 2024

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

[2]:
```
# Problem 2
N = 10
H = qt.num(N) + qt.qeye(N) / 2
lambd = .01

def state_correction(n):
    total = 0 * qt.basis(N)
    psi_n = qt.basis(N, n)
    for m in range(N):
        if m == n:
            continue
        psi_m = qt.basis(N, m)
        total += psi_m * (psi_m.dag() * lambd * qt.position(N)**4 * psi_n / (n␣
 ↪- m))
    return total

print(f"First order correction to state 0:\n{state_correction(0)}")
print(f"\nFirst order correction to state 1:\n{state_correction(1)}\n")

def first_energy_correction(n):
    return qt.expect(lambd * qt.position(N)**4, qt.basis(N, n))

def second_energy_correction(n):
    total = 0
    psi_n = qt.basis(N, n)
    for m in range(N):
        if m == n:
            continue
        psi_m = qt.basis(N, m)
        total += (psi_m.dag() * lambd * qt.position(N)**4 * psi_n)**2 / (n - m)
    # imaginary component will only exist because of rounding errors
    return np.real(total)
```

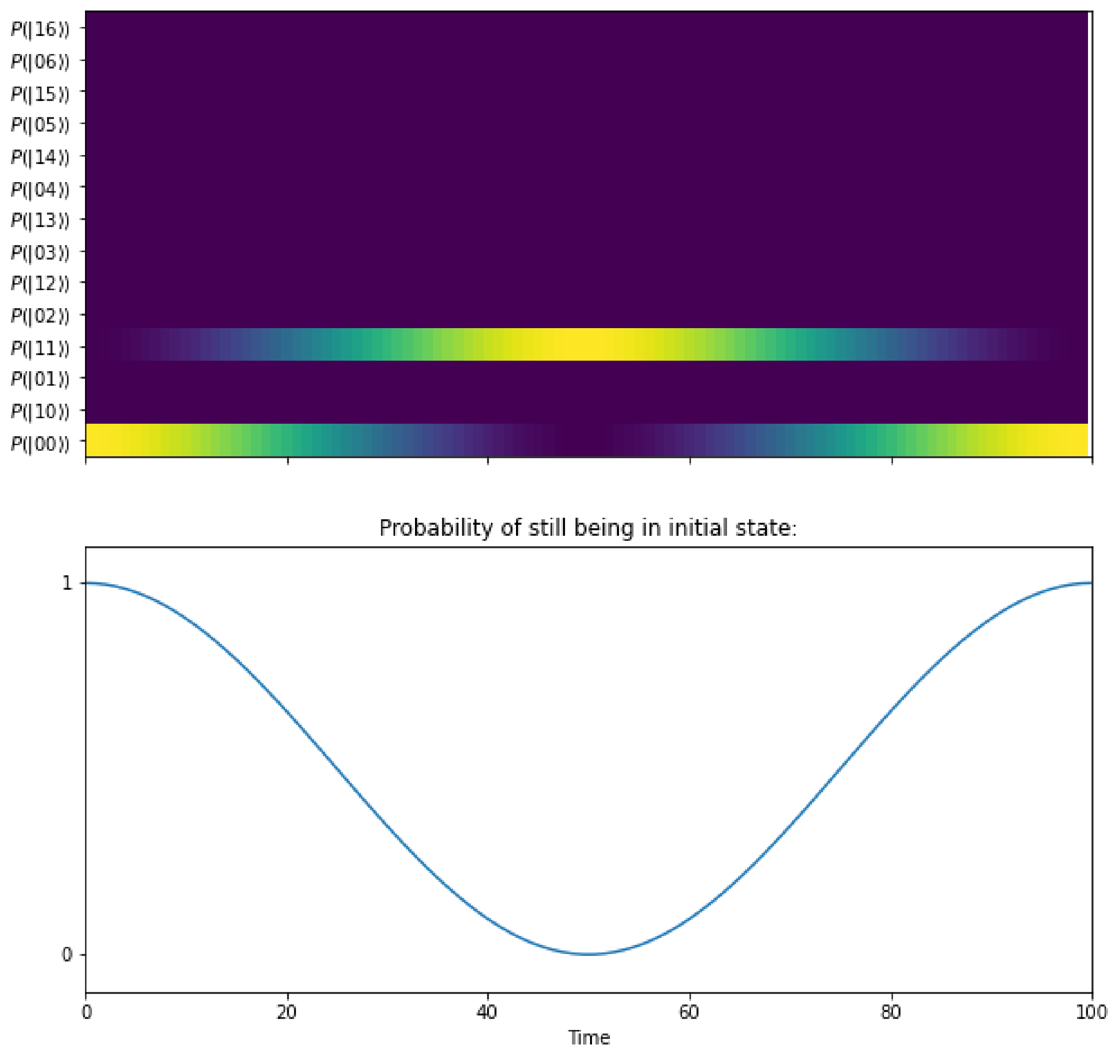
```
for i in range(N):
    print(f"First order correction to energy of state {i}:␣
  ↪{first_energy_correction(i):.5f}")
print('\n')
for i in range(N):
    print(f"Second order correction to energy of state {i}:␣
  ↪{second_energy_correction(i):.5f}")
print("\nPredicted energies, based on unperturbed energy" \
      " plus first and second order corrections:")
for i in range(N):
    predicted_energy = i + 0.5 + \
        first_energy_correction(i) + \
        second_energy_correction(i)
    print(f"Predicted energy of state {i}: {predicted_energy:.5f}")

# Make new Hamiltonian
H += lambd * qt.position(N)**4
actual_energies = H.eigenenergies()
print('\n')
for i in range(N):
    print(f"Actual energy of state {i}: {actual_energies[i]:.5f}" +
          f" = {2*i+1}/2 + {actual_energies[i] - (i+1/2):.5f}")
```

```
First order correction to state 0:
Quantum object: dims=[[10], [1]], shape=(10, 1), type='ket', dtype=Dense
Qobj data =
[[ 0.        ]
 [ 0.        ]
 [-0.0106066 ]
 [ 0.        ]
 [-0.00306186]
 [ 0.        ]
 [ 0.        ]
 [ 0.        ]
 [ 0.        ]
 [ 0.        ]]

First order correction to state 1:
Quantum object: dims=[[10], [1]], shape=(10, 1), type='ket', dtype=Dense
Qobj data =
[[ 0.        ]
 [ 0.        ]
 [ 0.        ]
 [-0.03061862]
 [ 0.        ]
 [-0.00684653]
 [ 0.        ]
 [ 0.        ]
 [ 0.        ]
 [ 0.        ]]

First order correction to energy of state 0: 0.00750
First order correction to energy of state 1: 0.03750
First order correction to energy of state 2: 0.09750
First order correction to energy of state 3: 0.18750
First order correction to energy of state 4: 0.30750
First order correction to energy of state 5: 0.45750
First order correction to energy of state 6: 0.63750
First order correction to energy of state 7: 0.84750
First order correction to energy of state 8: 0.86250
First order correction to energy of state 9: 0.38250


Second order correction to energy of state 0: -0.00026
Second order correction to energy of state 1: -0.00206
Second order correction to energy of state 2: -0.00769
Second order correction to energy of state 3: -0.01969
Second order correction to energy of state 4: -0.04061
Second order correction to energy of state 5: -0.07301
Second order correction to energy of state 6: -0.11156
Second order correction to energy of state 7: -0.03956
Second order correction to energy of state 8: 0.16013
Second order correction to energy of state 9: 0.13433

Predicted energies, based on unperturbed energy plus first and second order
corrections:
Predicted energy of state 0: 0.50724
Predicted energy of state 1: 1.53544
Predicted energy of state 2: 2.58981
Predicted energy of state 3: 3.66781
Predicted energy of state 4: 4.76689
Predicted energy of state 5: 5.88449
Predicted energy of state 6: 7.02594
Predicted energy of state 7: 8.30794
Predicted energy of state 8: 9.52263
Predicted energy of state 9: 10.01683


Actual energy of state 0: 0.50726 = 1/2 + 0.00726
Actual energy of state 1: 1.53565 = 3/2 + 0.03565
Actual energy of state 2: 2.59085 = 5/2 + 0.09085
Actual energy of state 3: 3.67110 = 7/2 + 0.17110
Actual energy of state 4: 4.77501 = 9/2 + 0.27501
Actual energy of state 5: 5.90070 = 11/2 + 0.40070
Actual energy of state 6: 7.03714 = 13/2 + 0.53714
Actual energy of state 7: 8.25292 = 15/2 + 0.75292
Actual energy of state 8: 9.50224 = 17/2 + 1.00224
Actual energy of state 9: 10.05213 = 19/2 + 0.55213
```

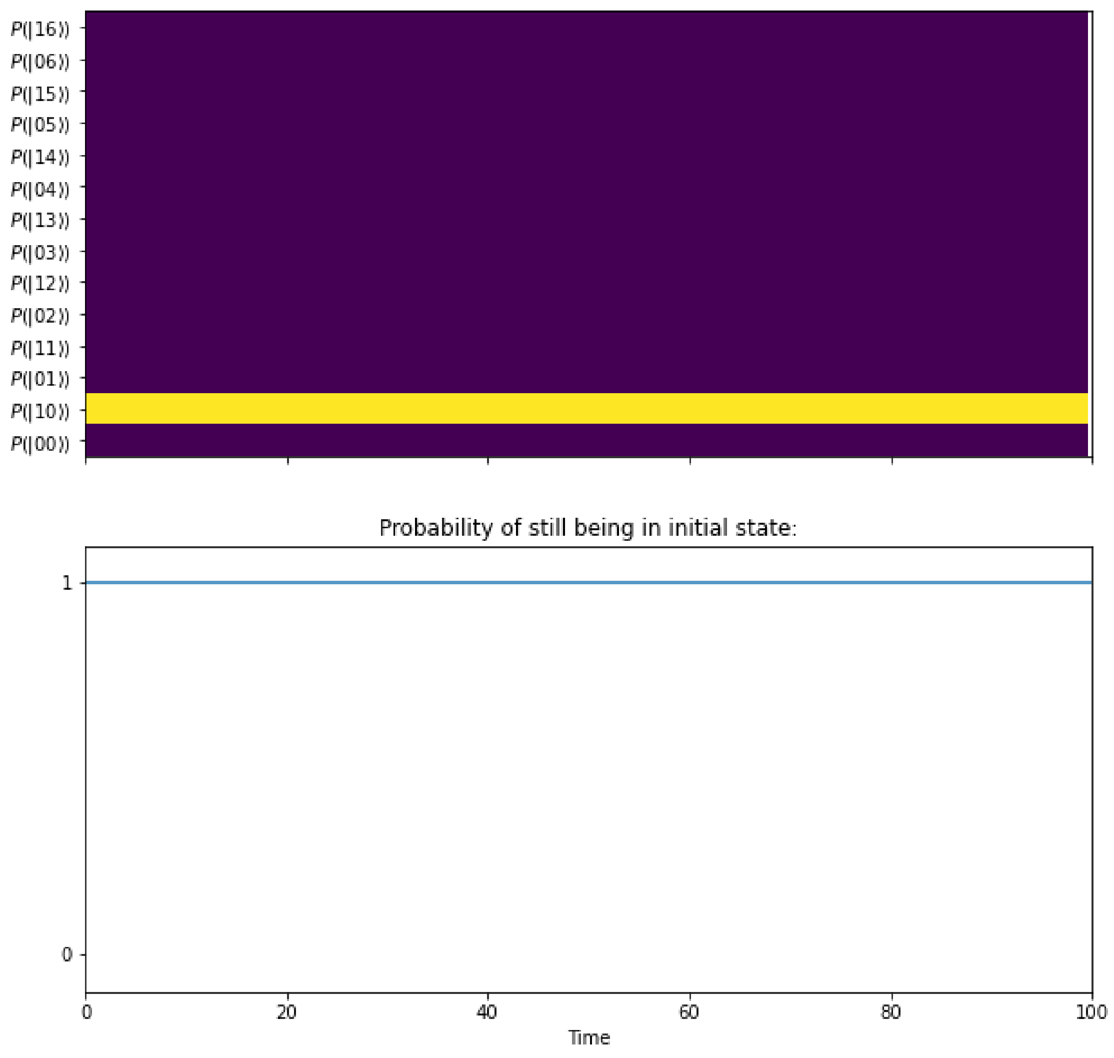
[4]:
```
# Problem 3

# Part (a)
Psi_1 = (qt.ket("0") + qt.ket("1")).unit()
Psi_2 = qt.ket("0")
Psi = qt.tensor(Psi_1, Psi_2)
print("Part (a)")
print(f"Total state vector:\n{Psi}")
expectation_value = qt.expect(qt.tensor(qt.sigmaz(), qt.qeye(2)), Psi)
print(f"Expectation value of sigma_z for qubit 1: {expectation_value}")
expectation_value = qt.expect(qt.tensor(qt.qeye(2), qt.sigmaz()), Psi)
print(f"Expectation value of sigma_z for qubit 2: {expectation_value}")
expectation_value = qt.expect(qt.tensor(qt.sigmaz(), qt.sigmaz()), Psi)
print(f"Expectation value of sigma_z tensor sigma_z: {expectation_value}")

# Part (b)
Psi_1 = (qt.ket("0") + qt.ket("1")).unit()
Psi_2 = Psi_1
Psi = qt.tensor(Psi_1, Psi_2)
print("\n\nPart (b)")
print(f"Total state vector:\n{Psi}")
expectation_value = qt.expect(qt.tensor(qt.sigmaz(), qt.sigmaz()), Psi)
print(f"Expectation value of sigma_z tensor sigma_z: {expectation_value}")
expectation_value = qt.expect(qt.tensor(qt.sigmax(), qt.sigmax()), Psi)
print(f"Expectation value of sigma_x tensor sigma_x: {expectation_value}")

# Part (c)
Psi = (qt.ket("00") + qt.ket("11")).unit()
print("\n\nPart (c)")
print(f"Total state vector:\n{Psi}")
expectation_value = qt.expect(qt.tensor(qt.sigmaz(), qt.sigmaz()), Psi)
print(f"Expectation value of sigma_z tensor sigma_z: {expectation_value}")
expectation_value = qt.expect(qt.tensor(qt.sigmax(), qt.sigmax()), Psi)
print(f"Expectation value of sigma_x tensor sigma_x: {expectation_value}")
```

```
Part (a)
Total state vector:
Quantum object: dims=[[2, 2], [1, 1]], shape=(4, 1), type='ket', dtype=Dense
Qobj data =
[[0.70710678]
 [0.        ]
 [0.70710678]
 [0.        ]]
Expectation value of sigma_z for qubit 1: 0.0
Expectation value of sigma_z for qubit 2: 0.9999999999999998
Expectation value of sigma_z tensor sigma_z: 0.0


Part (b)
Total state vector:
Quantum object: dims=[[2, 2], [1, 1]], shape=(4, 1), type='ket', dtype=Dense
Qobj data =
[[0.5]
 [0.5]
 [0.5]
 [0.5]]
Expectation value of sigma_z tensor sigma_z: 0.0
Expectation value of sigma_x tensor sigma_x: 0.9999999999999996


Part (c)
Total state vector:
Quantum object: dims=[[2, 2], [1, 1]], shape=(4, 1), type='ket', dtype=Dense
Qobj data =
[[0.70710678]
 [0.        ]
 [0.        ]
 [0.70710678]]
Expectation value of sigma_z tensor sigma_z: 0.9999999999999998
Expectation value of sigma_x tensor sigma_x: 0.9999999999999998
```

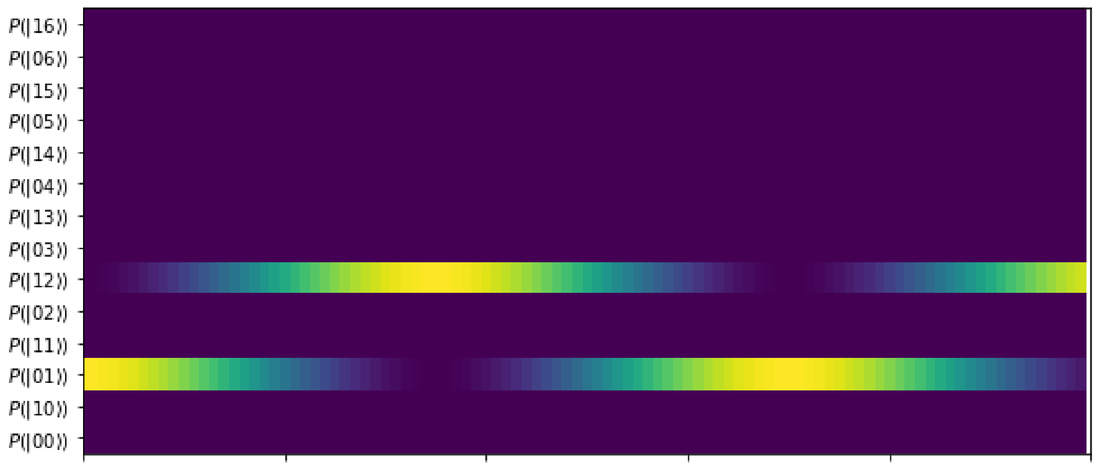
[5]:
```
# Problem 4, part (a)
N = 7
omega_0 = 2 * np.pi
g = omega_0 / 100
a = qt.destroy(N)
a_dag = qt.create(N)
def H(omega):
    return (omega_0 / 2) * qt.tensor(qt.sigmaz(), qt.qeye(N)) + \
        omega * (qt.tensor(qt.qeye(2), qt.num(N)) + qt.tensor(qt.qeye(2), qt.
 ↪qeye(N))) + \
        (g / 2) * (qt.tensor(qt.sigmap(), qt.destroy(N)) + qt.tensor(qt.
 ↪sigmam(), qt.create(N)))

H_a = H(omega_0)

def plot_time_evo(initial_state, hamiltonian):
    times = np.linspace(0, 100, 100)
    evolved_states = []
    for t in times:
        U = (-1j * hamiltonian * t).expm()
        evolved_states.append(U * initial_state)

    _, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(10, 10))
    # Make heatmap
    basis_states = [qt.ket([i % 2, i // 2], [2, N]) for i in reversed(range(2 *␣
 ↪N))]
    heatmap = np.matrix([[qt.expect(b.proj(), s) for s in evolved_states]
                         for b in basis_states])
    ax1.imshow(heatmap, aspect="auto")
    ax1.set_yticks(range(2 * N))
    ax1.set_yticklabels([f"$P(|{i%2}{i//2}\\rangle)$" for i in reversed(range(2␣
 ↪* N))])
    # Make line graph
    y_values = [qt.expect(initial_state.proj(), s) for s in evolved_states]
    plt.xlim(0, 100)
    plt.ylim(-.1, 1.1)
    ax2.set_yticks([0, 1])
    ax2.set_title("Probability of still being in initial state:")
    ax2.plot(times, y_values)
    plt.xlabel("Time")
    plt.show()


for Psi in [
    qt.ket("00", [2, N]),
    qt.ket("10", [2, N]),
    qt.ket("01", [2, N]),
    qt.ket("05", [2, N]),
    qt.ket("16", [2, N])
]:
    plot_time_evo(Psi, H_a)
```



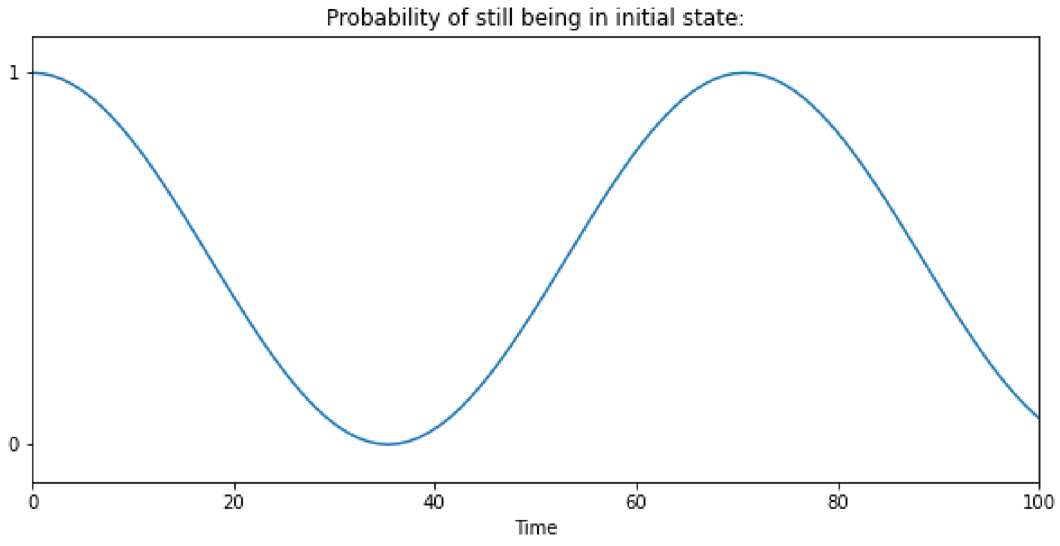
$P(|16\rangle)$
$P(|06\rangle)$
$P(|15\rangle)$
$P(|05\rangle)$
$P(|14\rangle)$
$P(|04\rangle)$
$P(|13\rangle)$
$P(|03\rangle)$
$P(|12\rangle)$
$P(|02\rangle)$
$P(|11\rangle)$
$P(|01\rangle)$
$P(|10\rangle)$
$P(|00\rangle)$
Probability of still being in initial state:
1
0
0
20
40
60
80
100
Time



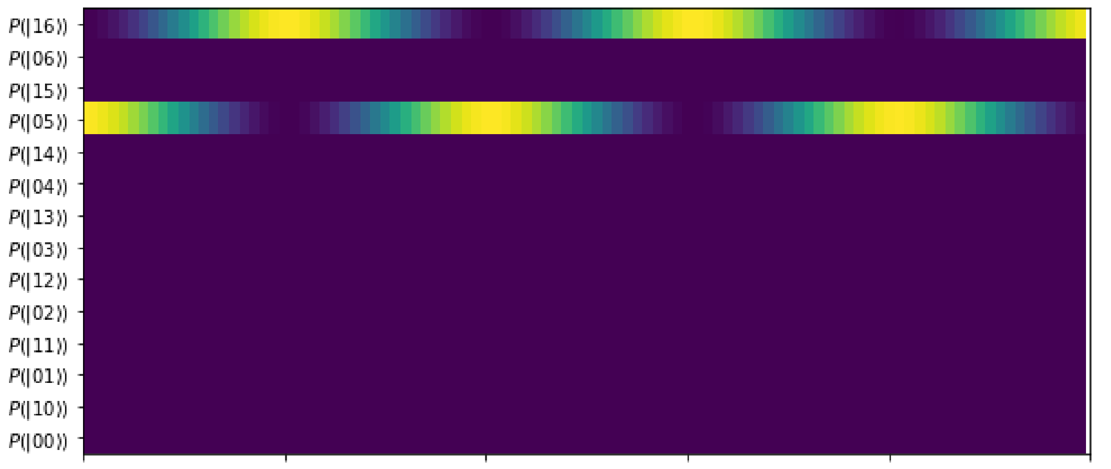
$P(|16\rangle)$
$P(|06\rangle)$
$P(|15\rangle)$
$P(|05\rangle)$
$P(|14\rangle)$
$P(|04\rangle)$
$P(|13\rangle)$
$P(|03\rangle)$
$P(|12\rangle)$
$P(|02\rangle)$
$P(|11\rangle)$
$P(|01\rangle)$
$P(|10\rangle)$
$P(|00\rangle)$
Probability of still being in initial state:
1
0
0
20
40
60
80
100
Time



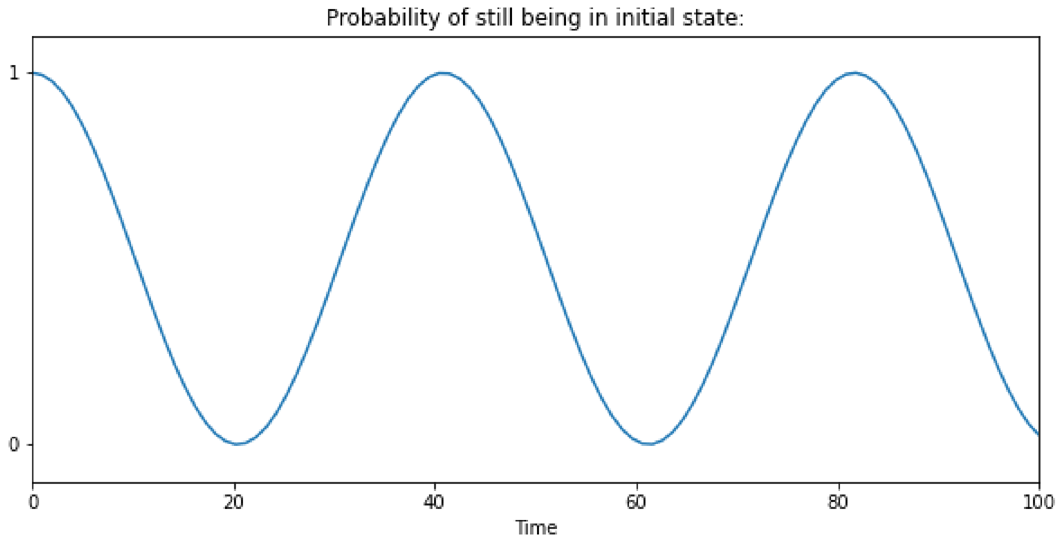
$P(|16\rangle)$
$P(|06\rangle)$
$P(|15\rangle)$
$P(|05\rangle)$
$P(|14\rangle)$
$P(|04\rangle)$
$P(|13\rangle)$
$P(|03\rangle)$
$P(|12\rangle)$
$P(|02\rangle)$
$P(|11\rangle)$
$P(|01\rangle)$
$P(|10\rangle)$
$P(|00\rangle)$



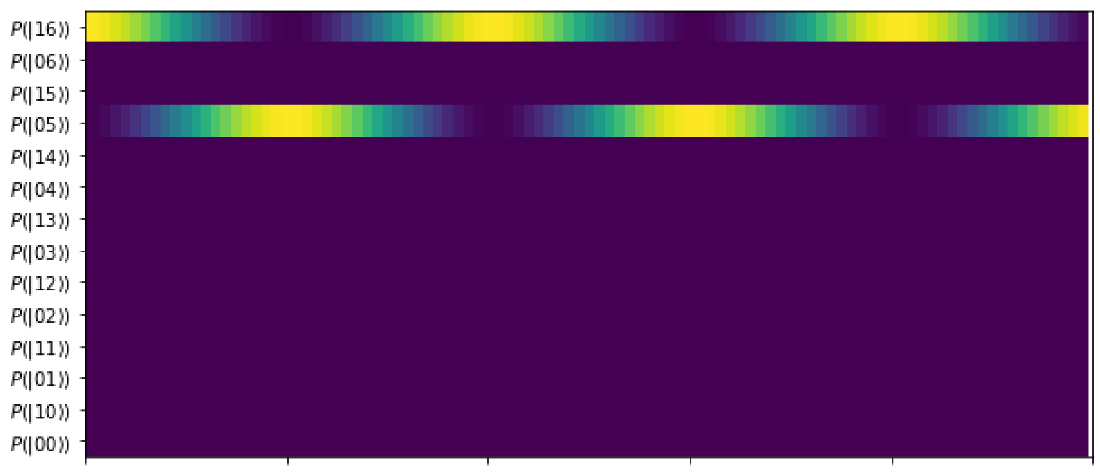
Probability of still being in initial state:
1
0
0
20
40
60
80
100
Time

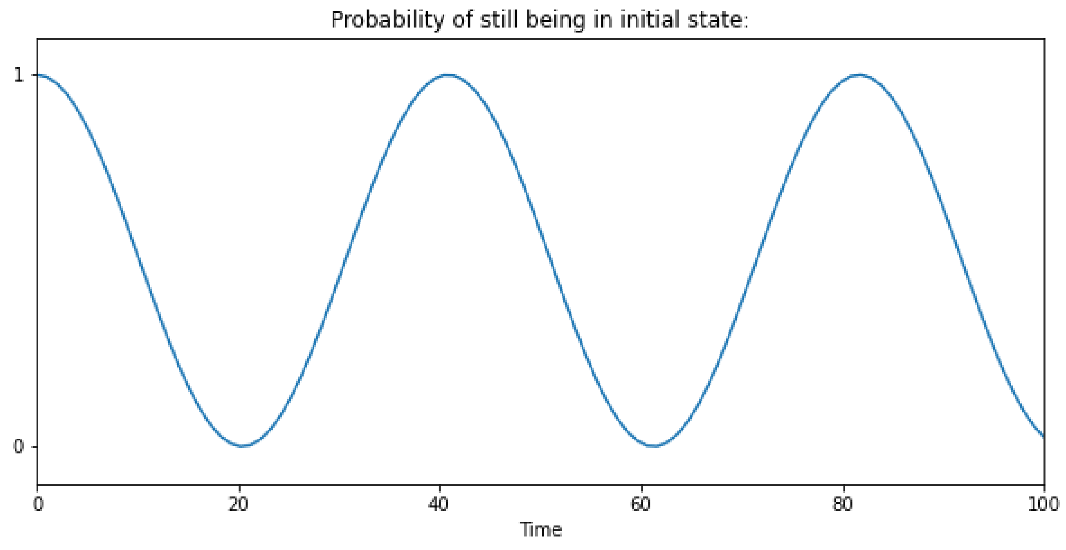


$P(|16\rangle)$
$P(|06\rangle)$
$P(|15\rangle)$
$P(|05\rangle)$
$P(|14\rangle)$
$P(|04\rangle)$
$P(|13\rangle)$
$P(|03\rangle)$
$P(|12\rangle)$
$P(|02\rangle)$
$P(|11\rangle)$
$P(|01\rangle)$
$P(|10\rangle)$
$P(|00\rangle)$

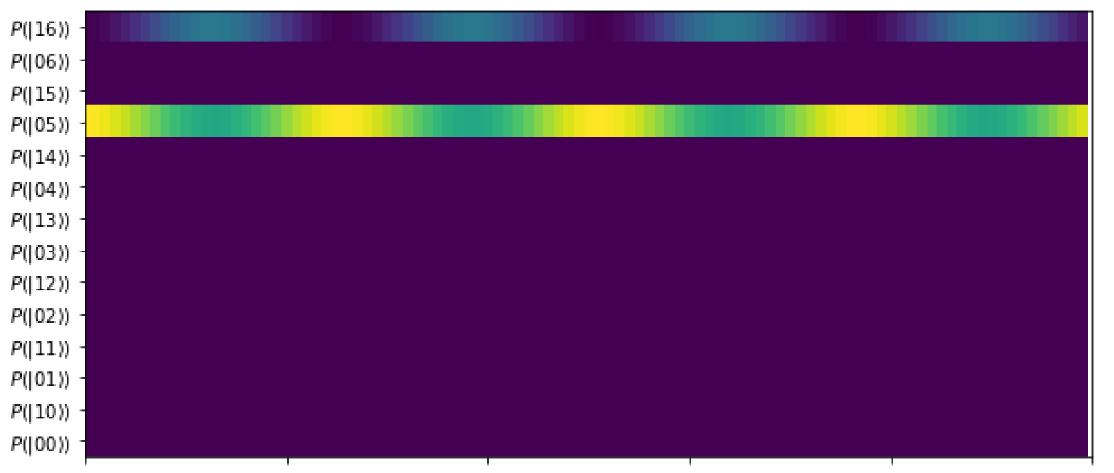


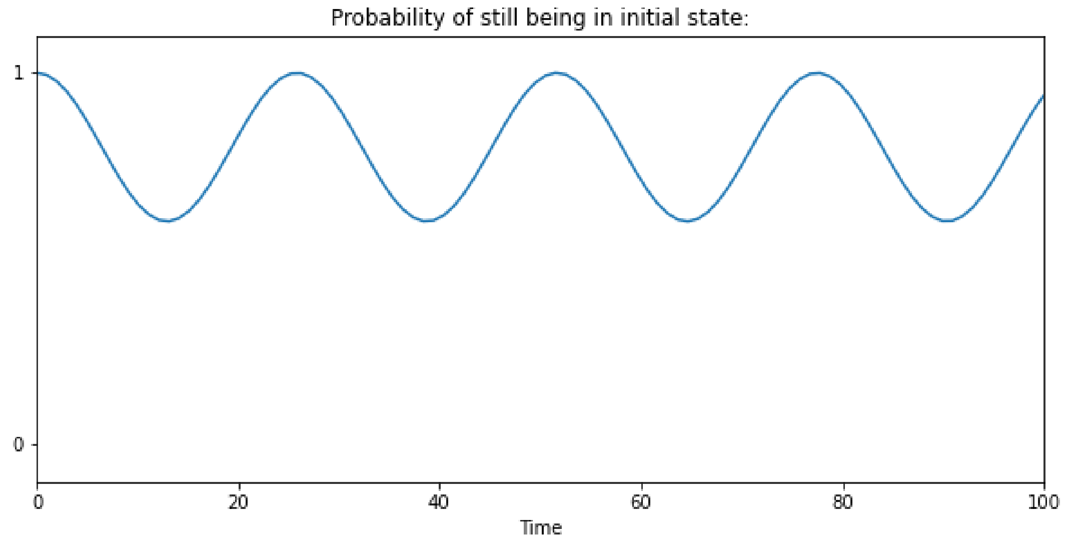
Probability of still being in initial state:
1
0
0
20
40
60
80
100
Time

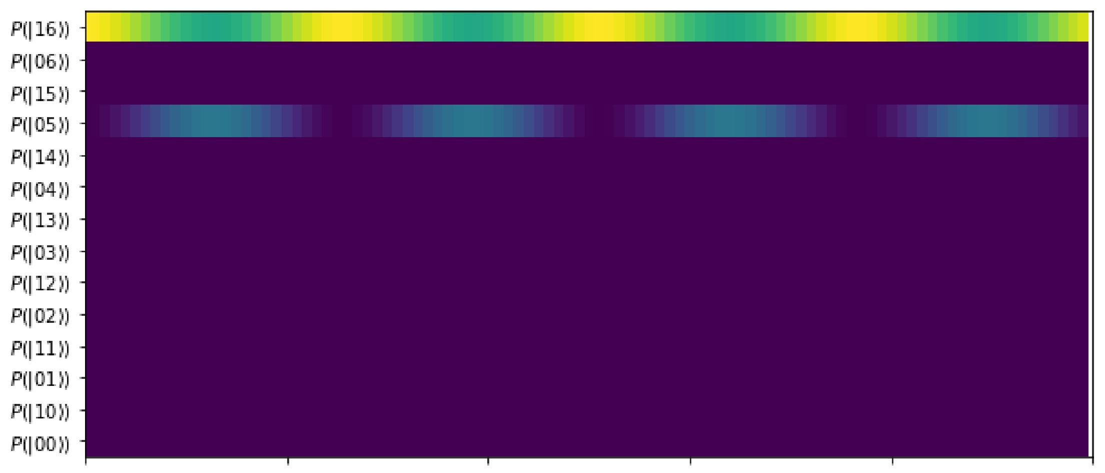


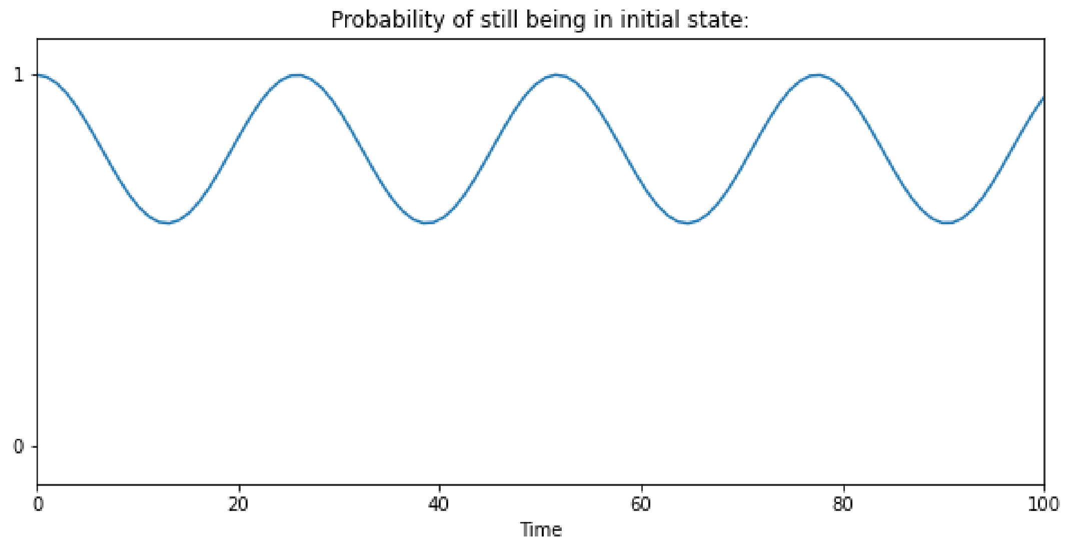


```
[6]: # Problem 4, part (b)
     H_b = H(omega_0 * 1.03)

     for Psi in [
         qt.ket("05", [2, N]),
         qt.ket("16", [2, N])
     ]:
         plot_time_evo(Psi, H_b)
```


$P(|16\rangle)$
$P(|06\rangle)$
$P(|15\rangle)$
$P(|05\rangle)$
$P(|14\rangle)$
$P(|04\rangle)$
$P(|13\rangle)$
$P(|03\rangle)$
$P(|12\rangle)$
$P(|02\rangle)$
$P(|11\rangle)$
$P(|01\rangle)$
$P(|10\rangle)$
$P(|00\rangle)$


Probability of still being in initial state:
1
0
0
20
40
60
80
100
Time


$P(|16\rangle)$
$P(|06\rangle)$
$P(|15\rangle)$
$P(|05\rangle)$
$P(|14\rangle)$
$P(|04\rangle)$
$P(|13\rangle)$
$P(|03\rangle)$
$P(|12\rangle)$
$P(|02\rangle)$
$P(|11\rangle)$
$P(|01\rangle)$
$P(|10\rangle)$
$P(|00\rangle)$


Probability of still being in initial state:
1
0
0
20
40
60
80
100
Time