PSE607A Project

Signatures of two-photon pulses from a quantum two-level system

Abhinav Raj Sharma
May 15, 2021

Abstract

Under intense Gaussian laser fields, photon emissions by a two-level atom through spontaneous emissions can be better understood by quantum trajectory approach. Re-excitations of the atom under laser field opens up probability of multiple photon emissions.Such re-excitations give rise to bunching of photons for even-π laser fields as they preferentially emit two-correlated pair of photons, if any emissions at all. Signature of such two photon emissions can be found in second-order coherence experimental measurements. [1]

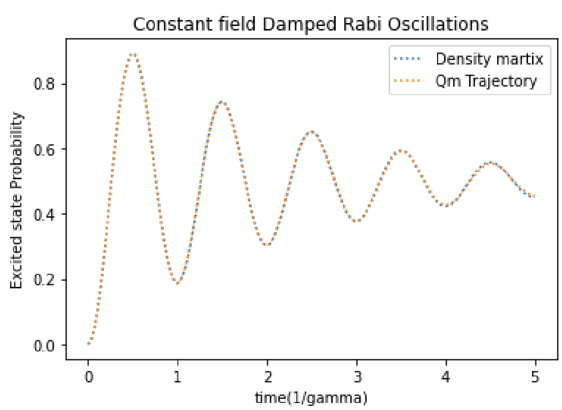

Figure 1. Damped Rabi Oscillations Average excited state probability at 5000 trajectories start to conveniently converge to that from evolution of density matrix directly.

In quantum trajectory approach to evolution of the system in presence of decay processes, we consider different possible trajectories, where each trajectory represents a path that can be taken by a single quantum system. So in different trajectories atom can decay and emit photon at different times(based on that some trajectories may have re-excitation), and therefore probability of emission by the two-level atom at different time varies based on distribution of emission times in the total trajectories. For large number of trajectories, the expected values observables converges to the results from density matrix evolution calculations(which it should, because mathematics is all the same).(Fig 1.)

$$|\psi\rangle = \sqrt{1-P_e}\,|g\rangle + \sqrt{P_e}\,|e\rangle$$

If the two-level system is prepared in above state at t=0 (using Rabi Oscillations by laser fields) then there is P_e probability that the system will decay by spontaneous emission and probability of decay in time [t,t+dt] decays exponentially with time constant $1/\gamma$ (where γ is decay rate constant). More generally, in a general evolving state of two-level atom the probability of decay in time [t,t+dt] is equal to $(P_e\gamma \cdot dt)$. Higher the excited level probability at a time, higher the probability decay and photon emission at that time.

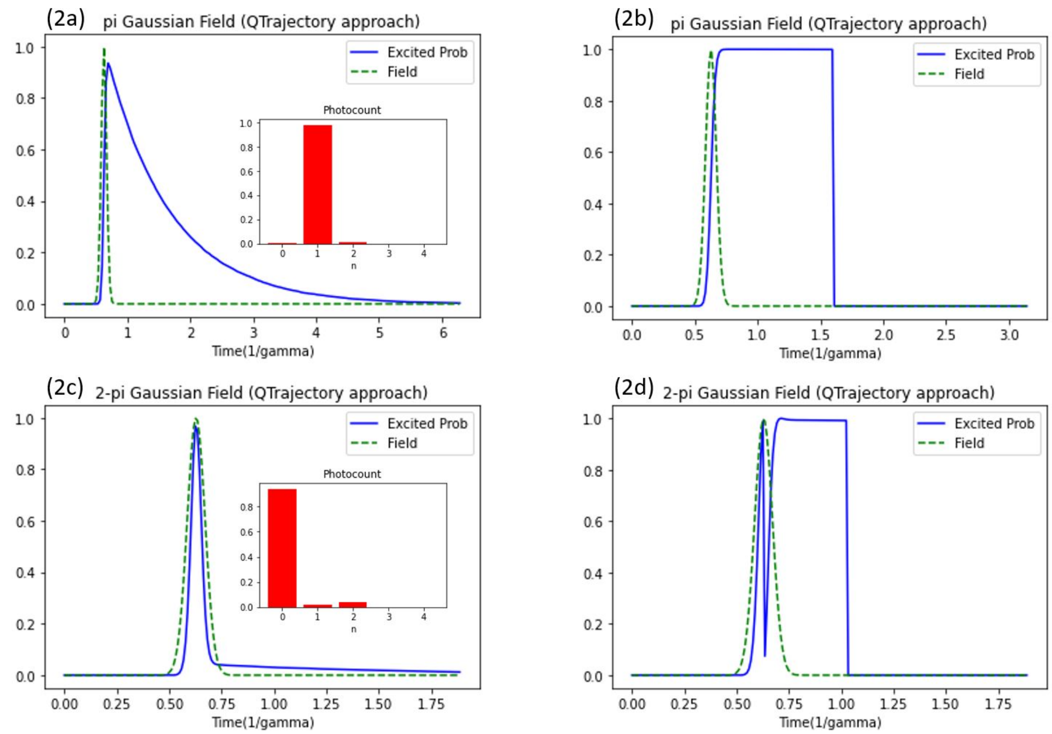

Figure 2.
FWHM $= \tau_\gamma/10 = 1/(10\gamma)$ for both pulses
(2a) and **(2c)** represent the evolution of average excited state probabilities for 10,000 trajectories and Photocounts (P_n : Probability of emission of n photons) under π and 2π gaussian pulses.
(2b) represents typical trajectory under π-pulse which results in single photon emission
(2d) represents typical trajectory under 2π-pulse which results in two photon emission

In different trajectories atom can decay and emit photon at different times. If photon is emitted and atom goes to ground state at a time during the laser field, then field re-excites the atom and possibility of another photon emission arises. There can be trajectories with single emissions or multiple or none.

Since here we have short gaussian pulses with **FWHM** tenth of relaxation time, the probability of emission during pulse is low, therefore probability of multiple emissions during the pulse kind of exponentially plummets.

For π pulse, (see Fig (2a),(2b)) $E_n = 1.0098$ emission during pulse is less probable, in most of the trajectories (see (2a)) atom is excited to excited state during the pulse add after that it exponentially decays with almost certainity contributing to P_1(those that collapse to ground contribute to P_0). Out of few trajectories that decay during the pulse and re-excite, some decay again contributing to P_2. Therefore such system can be used as single photon sources.

For 2π pulse, (see Fig (2c),(2d)) $E_n = 0.0991$ emission during pulse is less probable, in most of the trajectories (see (2a)) atom just oscillates once during the pulse and then remains at ground state contributing to P_0. Those few trajectories that decay near field peak(max. P_e gives max. decays) are re-excited to almost excited state by near-half pulse area, gives another almost certain decay contributing to P_2 (some few trajectories don't decay and contribute to P_1, therefore $P_2 > P_1$). Therefore such systems preferentially emit two photons, if any emission at all.

For both the pulses, due to small FWHM more than two emission counts, $P_n, (n > 2)$ are suppressed, as multiple emission during pulse is highly unlikely, small time length for many decays to occur.

Defining Photon number purity $Pi_n, (n > 0)$ as $\mathrm{Pi}_n = \frac{P_n}{\sum_{n>0} P_n}$

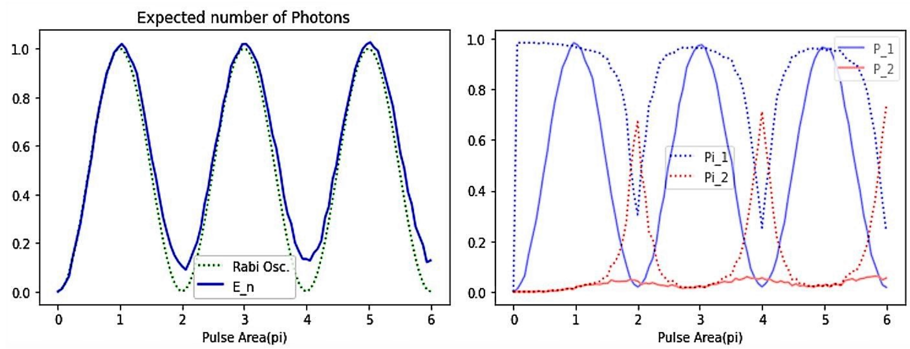

Figure 3. **(3a)** E_n closely follow P_e which it should because of most probably no emissions during pulse and after that emission probability $= P_e$, but deviates significantly for even-π pulse due to preferential two photon emissions. **(3b)** For odd-π pulses single emissions are highly likely. For even-π pulses no emission are highly likely and $Pi_2 > Pi_1$.

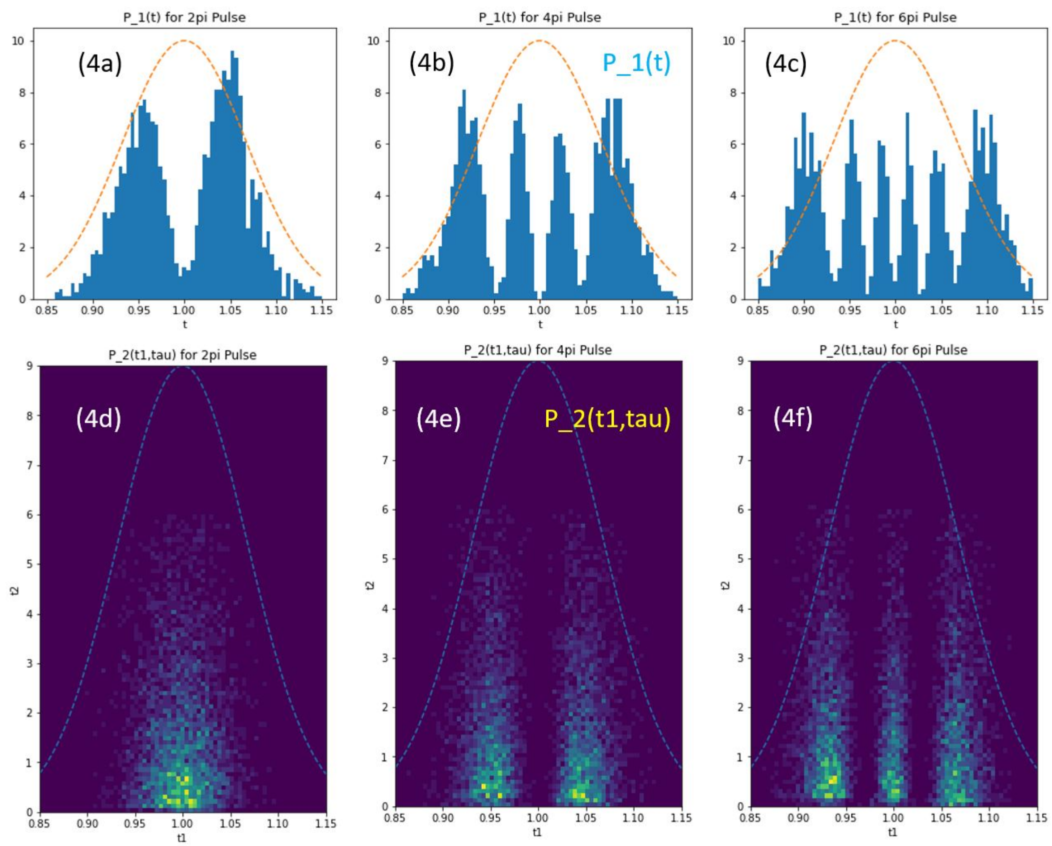

Figure 4. **(4a)**, **(4b)** and **(4c)** represent variation of single photon emission time in trajectories with single photon emission *t1* among 100,000 trajectories simulated.
(4d), **(4e)** and **(4f)** represent variation of first photon emission time *t1* (along x) and time difference τ b/w two emissions, in trajectories with two photon emission among 100,000 trajectories simulated.

Considering two photon emissions only, 2π**,** 4π **and** 6π **pulse** have 1, 2 and 3 P_e peaks respectively, and first photon are most likely to be emitted at times close to those peaks, that explains clusters of first emission time(t1). (see Fig (4d), (4e) and (4f))

After first emission for 2π pulse, atom re-excites again and then is allowed to decay exponentially resulting in fading spread in τ. (see Fig (4d))

After first emission for 4π and 6π pulse, atom oscillates eventually reaching excited state when pulse ends, second photon emission either occur at one of the intermediate peaks in the oscillation(if present) resulting in small clusters in τ or it occurs after the laser pulse during exponential decay resulting in spread in τ. (see Fig (4e) and (4f))

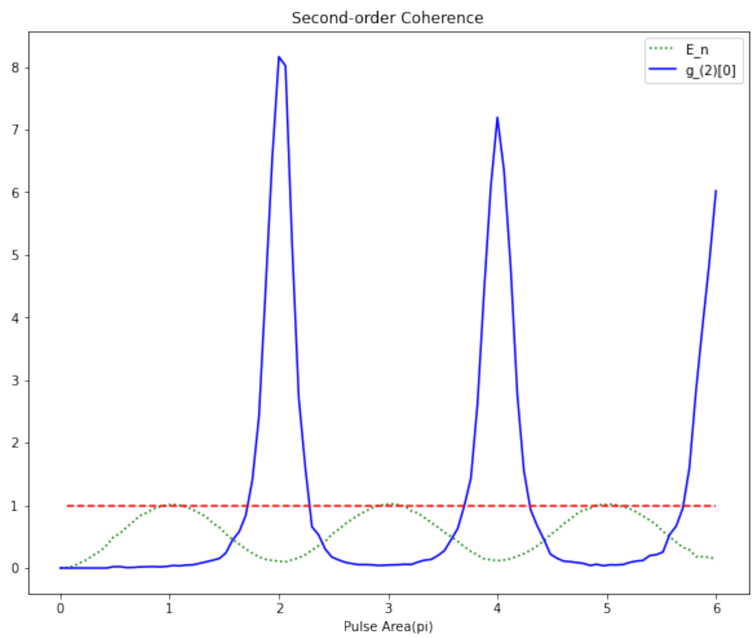

Figure 5. Second-order coherence $g^{(2)}[0]$ and Expected number of photons Vs Pulse Area (averaged over 2000 trajectories for each pulse)

Second-order coherence values($g^{(2)}[0]$) describes probability to detect a correlated photon pair relative to probability of detecting uncorrelated photon pair in a coherent pulse of same expected number of photons. $g^{(2)}[0]$ values can be experimentally observed, and their greater than 1 experimental values can be seen as signature of preferential two photon emission from two-level atom under even-π pulses. For even-π pulses, high $g^{(2)}[0]$ values suggest bunching of photons, which is what we observed in probability density function of two photon emission times(time of first emissions kind of gives out much information about time of second emission).

References

1. Fischer, K., Hanschke, L., Wierzbowski, J. et al. Signatures of two-photon pulses from a quantum two-level system. Nature Phys 13, 649–654 (2017).
2. Fox, Mark. Quantum Optics: an Introduction. Oxford University Press, (2013).
3. http://qutip.org/docs/latest/guide/dynamics/dynamics-monte.html
4. Johansson, J. R., Nation, P. D. and Nori, F. QuTiP 2: a Python framework for the dynamics of open quantum systems. Comput. Phys. Commun. 184, 1234-1240 (2013).

Codes

[**all codes below ran in google colab]

[0]'Imports (common in all of the below codes)'

```
!pip install qutip
import numpy as np
import matplotlib.pyplot as plt
from qutip import *
```

[1]'Constant field Damped Rabi Oscillations'

```
omega = 2*np.pi      #Rabi frequency (arbitrary value chosen)
gamma = omega/10     #Relaxation constant
psi0 = basis(2,0)    #Initially in ground

times = np.linspace(0.0, 5.0, 200)
psi_g = basis(2,0)
psi_e = basis(2,1)
a  = psi_g*psi_e.dag()
H = (omega/2)*(psi_g*psi_e.dag() + psi_e*psi_g.dag())

# using Density matrix formalism
data_dm = mesolve(H, psi0, times, [np.sqrt(gamma) * a], [psi_g*psi_g.dag()
, psi_e*psi_e.dag()])
                                #Collapse operator

# using Quantum Trajectory approach
data_qt = mcsolve(H, psi0, times, [np.sqrt(gamma) * a], [psi_g*psi_g.dag()
, psi_e*psi_e.dag()], ntraj=5000)
                                #Collapse operator

plt.figure()
plt.plot(times, data_dm.expect[1], linestyle = 'dotted')
plt.plot(times, data_qt.expect[1], linestyle = 'dotted')
plt.title('Constant field Damped Rabi Oscillations')
```

```
plt.xlabel('time(1/gamma)')
plt.ylabel('Excited state Probability')
plt.legend(('Density martix','Qm Trajectory'))
plt.show()
```

[2] **'Excited State Probability and Photocount for n-pie Gaussian pulse'**

```
n=1;                          #n-pi pulse
num_traj = 10000              #Number of trajectories
gamma = 2*np.pi/10            #Relaxation constant (arbitrary value chosen)
psi0 = basis(2,0)             #Initially in ground

times = np.linspace(0.0, 10.0, 200)
psi_g = basis(2,0)
psi_e = basis(2,1)
a  = psi_g*psi_e.dag()

H0 = Qobj([[0,0],[0,0]])                                #time-independent
comp. of H

H1 = (1/2)*(psi_g*psi_e.dag() + psi_e*psi_g.dag())  #time-dependent comp.
of H
def H1_coeff(t, args):
    return (20*n*gamma*np.sqrt(np.pi*np.log(2))) *
np.exp(-(20*gamma*np.sqrt(np.log(2))*(t-1)) ** 2)
            # Rabi freq.(t) [ for n-pi Gaussian pulse with FWHM =
1/(10*gamma) ]

H = [H0,[H1, H1_coeff]]

data = mcsolve(H, psi0, times, [np.sqrt(gamma) * a], [psi_g*psi_g.dag() ,
psi_e*psi_e.dag()], ntraj=num_traj)
                                #Collapse operator

plt.figure()
plt.plot(times*gamma, data.expect[1],'b', linestyle = 'solid')
```

```
plt.plot(times*gamma,np.exp(-(20*gamma*np.sqrt(np.log(2))*(times-1)) **
2),'g', linestyle = 'dashed')
plt.title('pi Gaussian Field (QTrajectory approach)')
plt.xlabel('Time(1/gamma)')
plt.legend(("Excited Prob", "Field"))
plt.show()

total_col = 0
x_P_n = [0,1,2,3,4]
P_n = [0,0,0,0,0]                              #Photocount


for x in data.col_times:
  total_col += len(x)
  for y in range(6):
    if len(x) == y:
      P_n[y] += 1/num_traj

E_n = total_col/num_traj        #Expected no. of emissions
print("E_n", E_n)

print(P_n)

fig = plt.figure(figsize=(3,2))
ax = fig.add_axes([0,0,1,1])
ax.bar(x_P_n,P_n,color='red')
plt.title('Photocount')
plt.xlabel('n')
plt.show()
```

[3] **'Expected # photons, Photocount and g(2)[0] variation with Pulse area'**

```
possible_n = np.linspace(0,6,100)   #possible n-pi pulse

E_n = [0,0]
E_n.clear()
P_0 = [0,0]
```

```
P_0.clear()
P_1 = [0,0]
P_1.clear()
P_2 = [0,0]
P_2.clear()

num_traj = 2000              #Number of trajectories
gamma = 2*np.pi/10           #Relaxation constant (arbitrary value chosen)
psi0 = basis(2,0)            #Initially in ground

times = np.linspace(0.0, 10.0, 200)
psi_g = basis(2,0)
psi_e = basis(2,1)
a  = psi_g*psi_e.dag()

H0 = Qobj([[0,0],[0,0]])                             #time-independent
comp. of H
H1 = (1/2)*(psi_g*psi_e.dag() + psi_e*psi_g.dag())  #time-dependent comp.
of H

for n in possible_n:

  def H1_coeff(t, args):
      return (20*n*gamma*np.sqrt(np.pi*np.log(2))) *
np.exp(-(20*gamma*np.sqrt(np.log(2))*(t-1)) ** 2)
              # Rabi freq.(t) [ for n-pi Gaussian pulse with FWHM =
1/(10*gamma) ]

  H = [H0,[H1, H1_coeff]]

  data = mcsolve(H, psi0, times, [np.sqrt(gamma) * a], [psi_g*psi_g.dag()
, psi_e*psi_e.dag()], ntraj=num_traj)
                                   #Collapse operator

  total_col = 0
  total_col2 = 0
  P_n = [0,0,0]                          #Photocounts

  for x in data.col_times:
    total_col += len(x)
```

```
    total_col2 += len(x)*len(x)

    for y in range(0,3):
      if len(x) == y:
        P_n[y] += 1/num_traj

  E_n.append(total_col/num_traj)         #Expected no. of emissions
  E_n2.append(total_col2/num_traj)       #Expected no. of emissions sq.

  P_0.append(P_n[0])                     #Photocounts
  P_1.append(P_n[1])
  P_2.append(P_n[2])

plt.figure()
plt.plot(possible_n, (1-np.cos(np.pi*possible_n))/2,'g', linestyle =
'dotted')
plt.plot(possible_n, E_n,'b', linestyle = 'solid')
plt.title('Expected number of Photons')
plt.xlabel('Pulse Area(pi)')
plt.legend(("Rabi Osc.", "E_n"))
plt.show()

plt.figure()
plt.plot(possible_n, P_1,'b', linestyle = 'solid')
plt.plot(possible_n, P_2,'r', linestyle = 'solid')
plt.xlabel('Pulse Area(pi)')
plt.legend(("P_1", "P_2"))
plt.show()

Pi_1 = [0,0]
Pi_1.clear()
Pi_2 = [0,0]
Pi_2.clear()

for i in range(0,len(P_1)):
  Pi_1.append(P_1[i]/(1-P_0[i]))
  Pi_2.append(P_2[i]/(1-P_0[i]))

plt.figure()
plt.plot(possible_n, Pi_1,'b', linestyle = 'dotted')
```

```
plt.plot(possible_n, Pi_2,'r', linestyle = 'dotted')
plt.xlabel('Pulse Area(pi)')
plt.legend(("Pi_1", "Pi_2"))
plt.show()

g2 = [0,0]
g2.clear()

for i in range(0,len(E_n)):
  if E_n[i]==0:
    g2.append(0)
  else:
    g2.append((E_n2[i]-E_n[i])/(E_n[i]*E_n[i]))

plt.figure(figsize=(10,8))
plt.plot(possible_n, E_n, 'g', linestyle = 'dotted')
plt.plot(possible_n, g2,'b', linestyle = 'solid')
plt.plot(possible_n, possible_n/possible_n, 'r', linestyle = 'dashed')
plt.title('Second-order Coherence')
plt.xlabel('Pulse Area(pi)')
plt.legend(("E_n", "g_(2)[0]"))
plt.show()
```

[4] **'Probability density functions for single and double emission times'**

```
####################   2pi pulse   ##################################
n=2;                        #n-pi pulse
num_traj = 100000            #Number of trajectories
gamma = 2*np.pi/10          #Relaxation constant (arbitrary value chosen)
psi0 = basis(2,0)           #Initially in ground

times = np.linspace(0.0, 7.0, 200)
psi_g = basis(2,0)
psi_e = basis(2,1)
a  = psi_g*psi_e.dag()

H0 = Qobj([[0,0],[0,0]])                             #time-independent
comp. of H
```

```
H1 = (1/2)*(psi_g*psi_e.dag() + psi_e*psi_g.dag())  #time-dependent comp.
of H
def H1_coeff(t, args):
    return (20*n*gamma*np.sqrt(np.pi*np.log(2))) *
np.exp(-(20*gamma*np.sqrt(np.log(2))*(t-1)) ** 2)
            # Rabi freq.(t) [ for n-pi Gaussian pulse with FWHM =
1/(10*gamma) ]

H = [H0,[H1, H1_coeff]]

data_2pi = mcsolve(H, psi0, times, [np.sqrt(gamma) * a],
[psi_e*psi_e.dag()], ntraj=num_traj)
                                  #Collapse operator

one_t1_2pi = [0,0]
one_t1_2pi.clear()
two_t1_2pi = [0,0]
two_t1_2pi.clear()
two_t2_2pi = [0,0]
two_t2_2pi.clear()

for x in data_2pi.col_times:
  if len(x) == 1:
    one_t1_2pi.append(x[0])
  if len(x) == 2:
    two_t1_2pi.append(x[0])
    two_t2_2pi.append(x[1]-x[0])

####################   4pi pulse   ##################################
n=4;                             #n-pi pulse
num_traj = 100000                 #Number of trajectories
gamma = 2*np.pi/10               #Relaxation constant (arbitrary value chosen)
psi0 = basis(2,0)                #Initially in ground

times = np.linspace(0.0, 7.0, 200)
psi_g = basis(2,0)
psi_e = basis(2,1)
a  = psi_g*psi_e.dag()
```

```
H0 = Qobj([[0,0],[0,0]])                             #time-independent
comp. of H

H1 = (1/2)*(psi_g*psi_e.dag() + psi_e*psi_g.dag())  #time-dependent comp.
of H
def H1_coeff(t, args):
    return (20*n*gamma*np.sqrt(np.pi*np.log(2))) *
np.exp(-(20*gamma*np.sqrt(np.log(2))*(t-1)) ** 2)
            # Rabi freq.(t) [ for n-pi Gaussian pulse with FWHM =
1/(10*gamma) ]

H = [H0,[H1, H1_coeff]]

data_4pi = mcsolve(H, psi0, times, [np.sqrt(gamma) * a],
[psi_e*psi_e.dag()], ntraj=num_traj)
                                #Collapse operator

one_t1_4pi = [0,0]
one_t1_4pi.clear()
two_t1_4pi = [0,0]
two_t1_4pi.clear()
two_t2_4pi = [0,0]
two_t2_4pi.clear()

for x in data_4pi.col_times:
  if len(x) == 1:
    one_t1_4pi.append(x[0])
  if len(x) == 2:
    two_t1_4pi.append(x[0])
    two_t2_4pi.append(x[1]-x[0])

####################   6pi pulse   ##################################
n=6;                             #n-pi pulse
num_traj =  100000                #Number of trajectories
gamma = 2*np.pi/10               #Relaxation constant (arbitrary value chosen)
psi0 = basis(2,0)                #Initially in ground

times = np.linspace(0.0, 7.0, 200)
psi_g = basis(2,0)
psi_e = basis(2,1)
```

```
a  = psi_g*psi_e.dag()

H0 = Qobj([[0,0],[0,0]])                                #time-independent
comp. of H

H1 = (1/2)*(psi_g*psi_e.dag() + psi_e*psi_g.dag())  #time-dependent comp.
of H
def H1_coeff(t, args):
    return (20*n*gamma*np.sqrt(np.pi*np.log(2))) *
np.exp(-(20*gamma*np.sqrt(np.log(2))*(t-1)) ** 2)
            # Rabi freq.(t) [ for n-pi Gaussian pulse with FWHM =
1/(10*gamma) ]

H = [H0,[H1, H1_coeff]]

data_6pi = mcsolve(H, psi0, times, [np.sqrt(gamma) * a],
[psi_e*psi_e.dag()], ntraj=num_traj)
                                #Collapse operator

one_t1_6pi = [0,0]
one_t1_6pi.clear()
two_t1_6pi = [0,0]
two_t1_6pi.clear()
two_t2_6pi = [0,0]
two_t2_6pi.clear()

for x in data_6pi.col_times:
  if len(x) == 1:
    one_t1_6pi.append(x[0])
  if len(x) == 2:
    two_t1_6pi.append(x[0])
    two_t2_6pi.append(x[1]-x[0])

                          # no. of traj. with single emissions
print(len(one_t1_2pi))
print(len(two_t1_2pi))
print(len(one_t1_4pi))
                          # no. of traj. with double emissions
print(len(two_t1_4pi))
print(len(one_t1_6pi))
```

```
print(len(two_t1_6pi))

t = np.linspace(0.85,1.15,70)
t2 = np.linspace(0.0,9.0,100)

fig = plt.figure(figsize=(5,5))
plt.hist(one_t1_2pi,bins = t, density=True)
plt.plot(t,10*np.exp(-(20*gamma*np.sqrt(np.log(2))*(t-1)) ** 2), linestyle
= 'dashed')
plt.title("P_1(t) for 2pi Pulse")
plt.xlabel("t")
plt.show()
fig = plt.figure(figsize=(5,8))
plt.hist2d(two_t1_2pi, two_t2_2pi, bins = [t,t2], density=True)
plt.plot(t,9*np.exp(-(20*gamma*np.sqrt(np.log(2))*(t-1)) ** 2), linestyle
= 'dashed')
plt.title("P_2(t1,tau) for 2pi Pulse")
plt.xlabel("t1")
plt.ylabel("t2")
plt.show()

fig = plt.figure(figsize=(5,5))
plt.hist(one_t1_4pi,bins = t, density=True)
plt.plot(t,10*np.exp(-(20*gamma*np.sqrt(np.log(2))*(t-1)) ** 2), linestyle
= 'dashed')
plt.title("P_1(t) for 4pi Pulse")
plt.xlabel("t")
plt.show()
fig = plt.figure(figsize=(5,8))
plt.hist2d(two_t1_4pi, two_t2_4pi, bins = [t,t2], density=True)
plt.plot(t,9*np.exp(-(20*gamma*np.sqrt(np.log(2))*(t-1)) ** 2), linestyle
= 'dashed')
plt.title("P_2(t1,tau) for 4pi Pulse")
plt.xlabel("t1")
plt.ylabel("t2")
plt.show()

fig = plt.figure(figsize=(5,5))
plt.hist(one_t1_6pi,bins = t, density=True)
```

```
plt.plot(t,10*np.exp(-(20*gamma*np.sqrt(np.log(2))*(t-1)) ** 2), linestyle
= 'dashed')
plt.title("P_1(t) for 6pi Pulse")
plt.xlabel("t")
plt.show()
fig = plt.figure(figsize=(5,8))
plt.hist2d(two_t1_6pi, two_t2_6pi, bins = [t,t2], density=True)
plt.plot(t,9*np.exp(-(20*gamma*np.sqrt(np.log(2))*(t-1)) ** 2), linestyle
= 'dashed')
plt.title("P_2(t1,tau) for 6pi Pulse")
plt.xlabel("t1")
plt.ylabel("t2")
plt.show()
```