QuTiP lecture: Vacuum Rabi oscillations in the Jaynes-Cummings model

Author: J.R. Johansson, robert@riken.jp

http://dml.riken.jp/~rob/

Latest version of this ipython notebook lecture is available at: http://github.com/jrjohansson/qutip-lectures

In [1]:
```
# setup the matplotlib graphics library and configure it to show
# figures inline in the notebook
%pylab inline
```

```
Welcome to pylab, a matplotlib-based Python environment [backend: module://IPython.zmq.pylab.backend_inline].
For more information, type 'help(pylab)'.
```

In [2]:
```
# make qutip available in the rest of the notebook
from qutip import *
```

Introduction

The Jaynes-Cumming model is the simplest possible model of quantum mechanical light-matter interaction, describing a single two-level atom interacting with a single electromagnetic cavity mode. The Hamiltonian for this system is (in dipole interaction form)

$$H = \hbar\omega_c a^\dagger a + \frac{1}{2}\hbar\omega_a\sigma_z + \hbar g(a^\dagger + a)(\sigma_- + \sigma_+)$$

or with the rotating-wave approximation

$$H_{\rm RWA} = \hbar\omega_c a^\dagger a + \frac{1}{2}\hbar\omega_a\sigma_z + \hbar g(a^\dagger\sigma_- + a\sigma_+)$$

where ω_c and ω_a are the frequencies of the cavity and atom, respectively, and g is the interaction strength.

Problem parameters

Here we use units where $\hbar = 1$:

In [3]:
```
wc = 1.0  * 2 * pi  # cavity frequency
wa = 1.0  * 2 * pi  # atom frequency
g  = 0.05 * 2 * pi  # coupling strength
kappa = 0.005       # cavity dissipation rate
gamma = 0.05        # atom dissipation rate
N = 15              # number of cavity fock states
n_th_a = 0.0        # avg number of thermal bath excitation
use_rwa = True

tlist = linspace(0,25,101)
```

Setup the operators, the Hamiltonian and initial state

In [4]:
```
# intial state
psi0 = tensor(basis(N,0), basis(2,1))    # start with an excited atom

# operators
a  = tensor(destroy(N), qeye(2))
sm = tensor(qeye(N), destroy(2))

# Hamiltonian
if use_rwa:
    H = wc * a.dag() * a + wa * sm.dag() * sm + g * (a.dag() * sm + a * sm.dag())
else:
    H = wc * a.dag() * a + wa * sm.dag() * sm + g * (a.dag() + a) * (sm + sm.dag())
```

Create a list of collapse operators that describe the dissipation

In [5]:
```
c_ops = []

# cavity relaxation
rate = kappa * (1 + n_th_a)
if rate > 0.0:
```

```
    c_ops.append(sqrt(rate) * a)

# cavity excitation, if temperature > 0
rate = kappa * n_th_a
if rate > 0.0:
    c_ops.append(sqrt(rate) * a.dag())

# qubit relaxation
rate = gamma
if rate > 0.0:
    c_ops.append(sqrt(rate) * sm)
```

Evolve the system

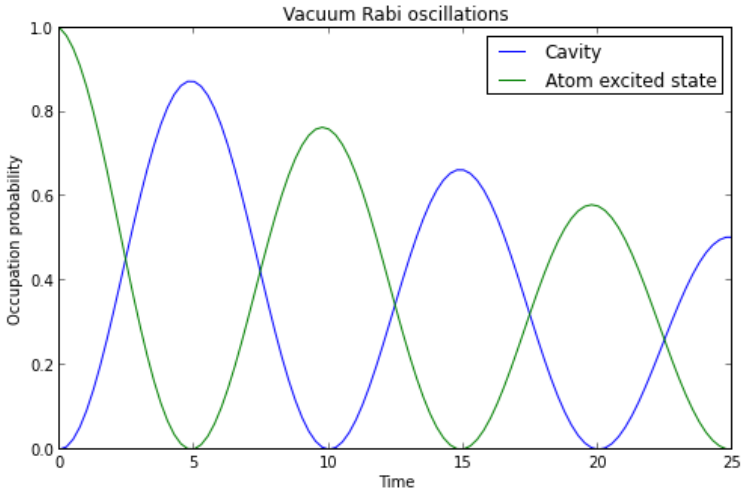
Here we evolve the system with the Lindblad master equation solver, and we request that the expectation values of the operators $a^\dagger a$ and $\sigma_+\sigma_-$ are returned by the solver by passing the list `[a.dag()*a, sm.dag()*sm]` as the fifth argument to the solver.

In [6]:
```
output = mesolve(H, psi0, tlist, c_ops, [a.dag() * a, sm.dag() * sm])
```

Visualize the results

Here we plot the excitation probabilities of the cavity and the atom (these expectation values were calculated by the `mesolve` above). We can clearly see how energy is being coherently transferred back and forth between the cavity and the atom.

In [7]:
```
n_c = output.expect[0]
n_a = output.expect[1]

figure(figsize=(8,5))
plot(tlist, n_c, label="Cavity")
plot(tlist, n_a, label="Atom excited state")
legend()
xlabel('Time')
ylabel('Occupation probability')
title('Vacuum Rabi oscillations')
show()
```



Cavity wigner function

In addition to the cavity's and atom's excitation probabilities, we may also be interested in for example the wigner function as a function of time. The Wigner function can give some valuable insight in the nature of the state of the resonators.

To calculate the Wigner function in QuTiP, we first recalculte the evolution without specifying any expectation value operators, which will result in that the solver return a list of density matrices for the system for the given time coordinates.

In [8]:
```
output = mesolve(H, psi0, tlist, c_ops, [])
```

Now, `output.states` contains a list of density matrices for the system for the time points specified in the list `tlist`:

In [9]:
```
output
```

Out [9]:
```
Odedata object with mesolve data.
---------------------------------
```

```
states = True
num_collapse = 0
```

In [10]:
```
type(output.states)
```

Out [10]:
```
list
```

In [11]:
```
len(output.states)
```

Out [11]:
```
101
```

In [12]:
```
output.states[-1] # indexing the list with -1 results in the last element in the list
```

Out [12]: Quantum object: dims = [[15, 2], [15, 2]], shape = [30, 30], type = oper, isHerm = True

$$\begin{pmatrix} 0.49605855334 & 0.0 & 0.0 & 0.0 & 0.0 & \cdots & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.000481112489832 & -0.0154966903108j & 0.0 & 0.0 & \cdots & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0154966903108j & 0.50346033417 & 0.0 & 0.0 & \cdots & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \cdots & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \cdots & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \cdots & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \cdots & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \cdots & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \cdots & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \cdots & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$$

Now let's look at the Wigner functions at the point in time when atom is in its ground state: $t = \{5, 15, 25\}$ (see the plot above).

For each of these points in time we need to:

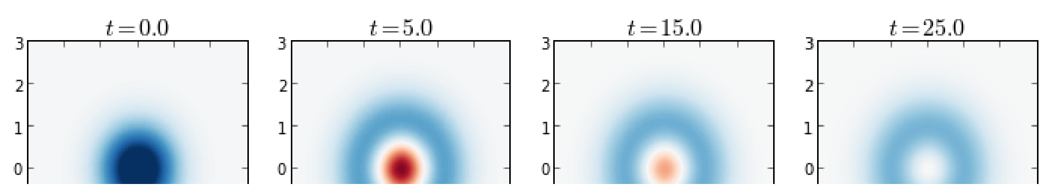
I. Find the system density matrix for the points in time that we are interested in.
II. Trace out the atom and obtain the reduced density matrix for the cavity.
III. Calculate and visualize the Wigner function fo the reduced cavity density matrix.

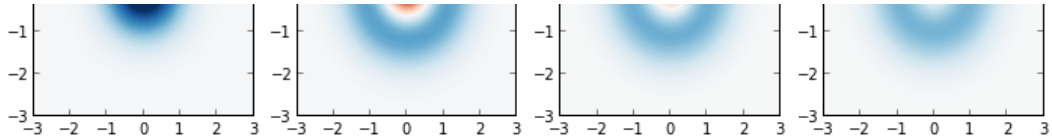
In [13]:
```
# find the indices of the density matrices for the times we are interested in
t_idx = where([tlist == t for t in [0.0, 5.0, 15.0, 25.0]])[1]
tlist[t_idx]
```

Out [13]:
```
array([  0.,   5.,  15.,  25.])
```

In [14]:
```
# get a list density matrices
rho_list = array(output.states)[t_idx]
```

In [15]:
```
# loop over the list of density matrices

xvec = linspace(-3,3,200)

fig, axes = subplots(1,len(rho_list), sharex=True, figsize=(3*len(rho_list),3))

for idx, rho in enumerate(rho_list):

    # trace out the atom from the density matrix, to obtain
    # the reduced density matrix for the cavity
    rho_cavity = ptrace(rho, 0)

    # calculate its wigner function
    W = wigner(rho_cavity, xvec, xvec)

    # plot its wigner function
    axes[idx].contourf(xvec, xvec, W, 100, norm=mpl.colors.Normalize(-.25,.25), cmap=get_cmap('RdBu'))

    axes[idx].set_title(r"$t = %.1f$" % tlist[t_idx][idx], fontsize=16)
```





At $t = 0$, the cavity is in it's ground state. At $t = 5, 15, 25$ it reaches it's maxium occupation in this Rabi-vacuum oscillation process. We can note that for $t = 5$ and $t = 15$ the Wigner function has negative values, indicating a truely quantum mechanical state. At $t = 25$, however, the wigner function no longer has negative values and can therefore be considered a classical state.

Alternative view of the same thing

In [16]:
```
t_idx = where([tlist == t for t in [0.0, 5.0, 10, 15, 20, 25]])[1]
rho_list = array(output.states)[t_idx]

fig_grid = (2, len(rho_list)*2)
fig = figure(figsize=(2.5*len(rho_list),5))

for idx, rho in enumerate(rho_list):
    rho_cavity = ptrace(rho, 0)
    W = wigner(rho_cavity, xvec, xvec)
    ax = subplot2grid(fig_grid, (0, 2*idx), colspan=2)
    ax.contourf(xvec, xvec, W, 100, norm=mpl.colors.Normalize(-.25,.25), cmap=get_cmap('RdBu'))
    ax.set_title(r"$t = %.1f$" % tlist[t_idx][idx], fontsize=16)

# plot the cavity occupation probability in the ground stateLecture-1-Jaynes-Cumming-model
Delete

ax = subplot2grid(fig_grid, (1, 1), colspan=(fig_grid[1]-2))
ax.plot(tlist, n_c, label="Cavity")
ax.plot(tlist, n_a, label="Atom excited state")
ax.legend()
ax.set_xlabel('Time')
ax.set_ylabel('Occupation probability')
```

Out [16]: <matplotlib.text.Text at 0x536d910>