QuTiP lecture: Correlation functions

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]:
```
%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]:
```
from qutip import *
```

First-order coherence function

Consider an oscillator that is interacting with a thermal environment. If the oscillator initially is in a coherent state, it will gradually decay to a thermal (incoherent) state. The amount of coherence can be quantified using the first-order optical coherence function

$$g^{(1)}(\tau) = \langle a^\dagger(\tau)a(0)\rangle/\sqrt{\langle a^\dagger(\tau)a(\tau)\rangle\langle a^\dagger(0)a(0)\rangle}$$

For a coherent state $|g^{(1)}(\tau)| = 1$, and for a completely incoherent (thermal) state $g^{(1)}(\tau) = 0$.

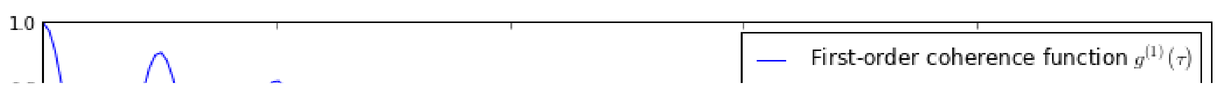
The following code calculates and plots $g^{(1)}(\tau)$ as a function of τ.

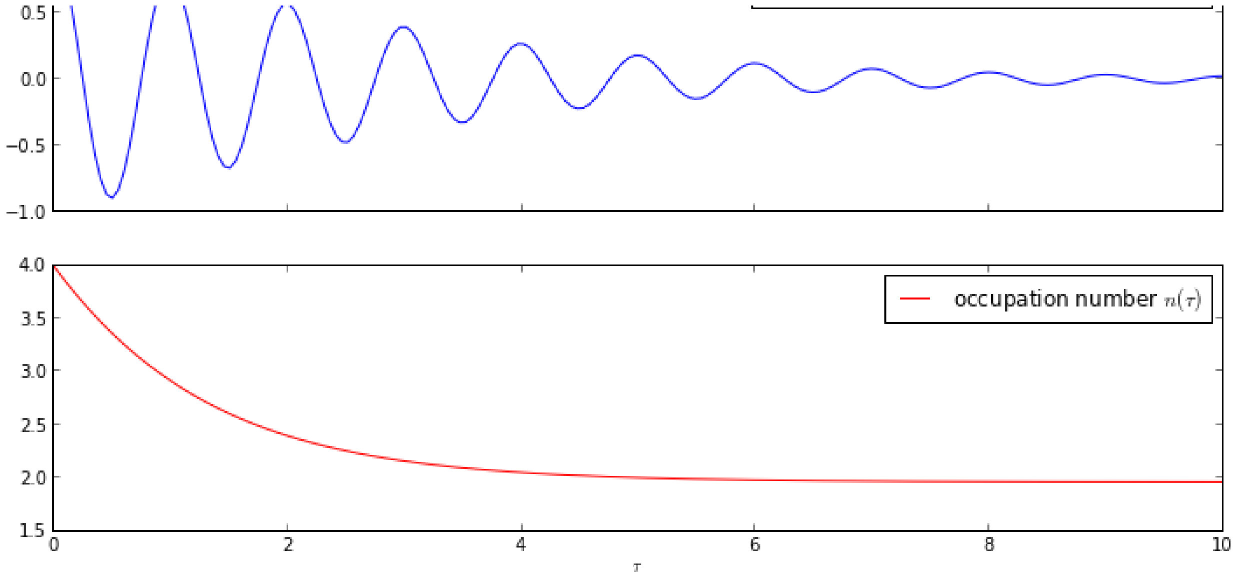
Example: Decay of a coherent state to an incoherent (thermal) state

In [3]:
```
N = 15
taulist = linspace(0,10.0,200)
a = destroy(N)
H = 2*pi*a.dag()*a

# collapse operator
G1 = 0.75
n_th = 2.00  # bath temperature in terms of excitation number
c_ops = [sqrt(G1*(1+n_th)) * a, sqrt(G1*n_th) * a.dag()]

# start with a coherent state
rho0 = coherent_dm(N, 2.0)

# first calculate the occupation number as a function of time
n = mesolve(H, rho0, taulist, c_ops, [a.dag() * a]).expect[0]

# calculate the correlation function G1 and normalize with n to obtain g1
G1 = correlation(H, rho0, None, taulist, c_ops, a.dag(), a)
g1 = G1 / sqrt(n[0] * n)
```

In [4]:
```
fig, axes = subplots(2, 1, sharex=True, figsize=(12,6))

axes[0].plot(taulist, real(g1), 'b', label=r'First-order coherence function $g^{(1)}
(\tau)$')
axes[1].plot(taulist, real(n),  'r', label=r'occupation number $n(\tau)$')
axes[0].legend()
axes[1].legend()
axes[1].set_xlabel(r'$\tau$')
```

Out [4]:
```
<matplotlib.text.Text at 0x2b5e250>
```





Leggett-Garg inequality

Definition: Given an observable $Q(t)$ that is bound below and above by $|Q(t)| \leq 1$, the assumptions of

- macroscopic realism
- noninvasive measurements

implies that

$$L(t_1, t_2) = \langle Q(t_1)Q(0)\rangle + \langle Q(t_1 + t_2)Q(t_1)\rangle - \langle Q(t_1 + t_2)Q(0)\rangle \leq 1$$

If Q is at a steady state at the initial time of measurement, we can set $\tau = t_1 = t_2$ and the Leggett-Garg inequality then reads

$$L(\tau) = 2\langle Q(\tau)Q(0)\rangle - \langle Q(2\tau)Q(0)\rangle \leq 1$$

References

- A. J. Leggett and A. Garg, Phys. Rev. Lett. 54, 857 (1985)
- A. J. Leggett, J. Phys. Condens. Matter 14, R415 (2002)

In [5]:
```
def leggett_garg(c_mat):
    """
    For a given correlation matrix c_mat = <Q(t1+t2)Q(t1)>, calculate the Leggett-Garg correlation.
    """

    N, M = shape(c_mat)

    lg_mat = zeros([N/2,M/2], dtype=complex)
    lg_vec = zeros(N/2, dtype=complex)

    # c_mat(i, j) = <Q(dt i+dt j)Q(dt i)>
    # LG = <Q(t_1)Q(0)> + <Q(t_1+t_2)Q(t_1)> - <Q(t_1+t_2)Q(0)>

    for i in range(N/2):
        lg_vec[i] = 2*c_mat[0, i] - c_mat[0, 2*i];

        for j in range(M/2):
            lg_mat[i, j] = c_mat[0, i] + c_mat[i, j] - c_mat[0, i+j];


    return lg_mat, lg_vec
```

Example: Leggett-Garg inequality for two coupled resonators (optomechanical system)

References:

- N. Lambert, J.R. Johansson, F. Nori, Phys. Rev. B 82, 245421 (2011).

In [6]:
```
wc = 1.0  * 2 * pi  # cavity frequency
wa = 1.0  * 2 * pi  # resonator frequency
g  = 0.3  * 2 * pi  # coupling strength
kappa = 0.075       # cavity dissipation rate
gamma = 0.005       # resonator dissipation rate
Na = Nc = 3         # number of cavity fock states
n_th = 0.0          # avg number of thermal bath excitation

tlist = linspace(0, 7.5, 251)
tlist_sub = tlist[0:(len(tlist)/2)]
```

In [7]:
```
# start with an excited resonator
rho0 = tensor(fock_dm(Na,0), fock_dm(Nc,1))

a = tensor(qeye(Nc), destroy(Na))
c = tensor(destroy(Nc), qeye(Na))

na = a.dag() * a
nc = c.dag() * c

H = wa * na + wc * nc - g * (a + a.dag()) * (c + c.dag())
```

In [8]:
```
# measurement operator on resonator
Q = na                                              # photon number resolving detector
#Q = tensor(qeye(Nc), 2 * fock_dm(Na, 1) - qeye(Na)) # fock-state |1> detector
#Q = tensor(qeye(Nc), qeye(Na) - 2 * fock_dm(Na, 0)) # click or no-click detector
```

In [9]:
```
c_op_list = []

rate = kappa * (1 + n_th)
if rate > 0.0:
    c_op_list.append(sqrt(rate) * c)

rate = kappa * n_th
if rate > 0.0:
    c_op_list.append(sqrt(rate) * c.dag())

rate = gamma * (1 + n_th)
if rate > 0.0:
    c_op_list.append(sqrt(rate) * a)

rate = gamma * n_th
if rate > 0.0:
    c_op_list.append(sqrt(rate) * a.dag())
```

Calculate the correlation function $\langle Q(t_1 + t_2)Q(t_1)\rangle$

Using the regression theorem, and QuTiP function `correlation`.

In [10]:
```
corr_mat = correlation(H, rho0, tlist, tlist, c_op_list, Q, Q)
```

Calculate the Leggett-Garg correlation

In [11]:
```
LG_tt, LG_t = leggett_garg(corr_mat)
```

Plot results

In [13]:

```
fig, axes = subplots(1, 2, figsize=(12,4))

axes[0].pcolor(tlist,tlist,abs(corr_mat),edgecolors='none')
axes[0].set_xlabel(r'$t_1 + t_2$')
axes[0].set_ylabel(r'$t_1$')
axes[0].autoscale(tight=True)

axes[1].pcolor(tlist_sub,tlist_sub,abs(LG_tt),edgecolors='none')
axes[1].set_xlabel(r'$t_1$')
axes[1].set_ylabel(r'$t_2$')
axes[1].autoscale(tight=True)

fig, axes = subplots(1, 1, figsize=(12,4))
axes.plot(tlist_sub, diag(real(LG_tt)), label=r'$\tau = t_1 = t_2$')
axes.plot(tlist_sub, ones(shape(tlist_sub)), 'k', label=r'quantum boundary')
axes.fill_between(tlist_sub, diag(LG_tt), 1, where=(diag(LG_tt)>1), color="green",
alpha=0.5)
axes.set_xlim([0, max(tlist_sub)])
axes.legend(loc=0)
axes.set_xlabel(r'$\tau$', fontsize=18)
axes.set_ylabel(r'LG($\tau$)', fontsize=18);
```