example-atom-cavity-correlation-function

June 25, 2014

1 QuTiP example: Correlation functions and spectrum of a atom-cavity system

J.R. Johansson and P.D. Nation

For more information about QuTiP see http://qutip.org

```
In [1]: %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
In [2]: from qutip import *
        import time
```

1.1 Model and parameters

We use the Jaynes-Cumming model of a single two-level atom interacting with a single-mode cavity via a dipole interaction and under the rotating wave approximation.

```
In [3]: kappa = 2
        gamma = 0.2
        g = 5
        wc = 0
        w0 = 0
        wl = 0
        N = 5
        E = 0.5

        tlist = linspace(0,10.0,500)
```

1.2 Correlation functions and spectrum using `eseries`

```
In [4]: def probcorr(E,kappa,gamma,g,wc,w0,wl,N):
            #
            # returns the two-time correlation and covariance of the intracavity
            # field as exponential series for the problem of a coherently driven
            # cavity with a two-level atom
            #
            # E = amplitude of driving field, kappa = mirror coupling,
            # gamma = spontaneous emission rate, g = atom-field coupling,
            # wc = cavity frequency, w0 = atomic frequency, wl = driving field frequency,
            # N = size of Hilbert space for intracavity field (zero to N-1 photons)
            #
```

```
    # Hamiltonian
    a  = tensor(destroy(N),qeye(2))
    sm = tensor(qeye(N),destroy(2))
    H = (w0-wl)*sm.dag()*sm + (wc-wl)*a.dag()*a + 1j*g*(a.dag()*sm - sm.dag()*a) \
        + E*(a.dag()+a)

    # collapse operators
    C1 = sqrt(2*kappa) * a
    C2 = sqrt(gamma)   * sm.dag()
    c_op_list = [C1, C2]

    # Calculate the Liouvillian
    L = liouvillian(H, c_op_list)

    # Find steady state density matrix and field
    rhoss = steadystate(L);

    ass = expect(a,rhoss);

    # Initial condition for regression theorem
    arho = a*rhoss;

    # Solve differential equation with this initial condition
    solES = ode2es(L,arho);

    # Find trace(a' * solution)
    corrES = expect(a.dag(),solES);

    # Calculate the covariance by subtracting product of means
    covES = corrES - real(conjugate(ass)*ass)

    return corrES, covES
```

```
In [5]: start_time=time.time()
        corrES,covES = probcorr(E,kappa,gamma,g,wc,w0,wl,N);
        print('time elapsed (probcorr) = ' +str(time.time()-start_time))
```

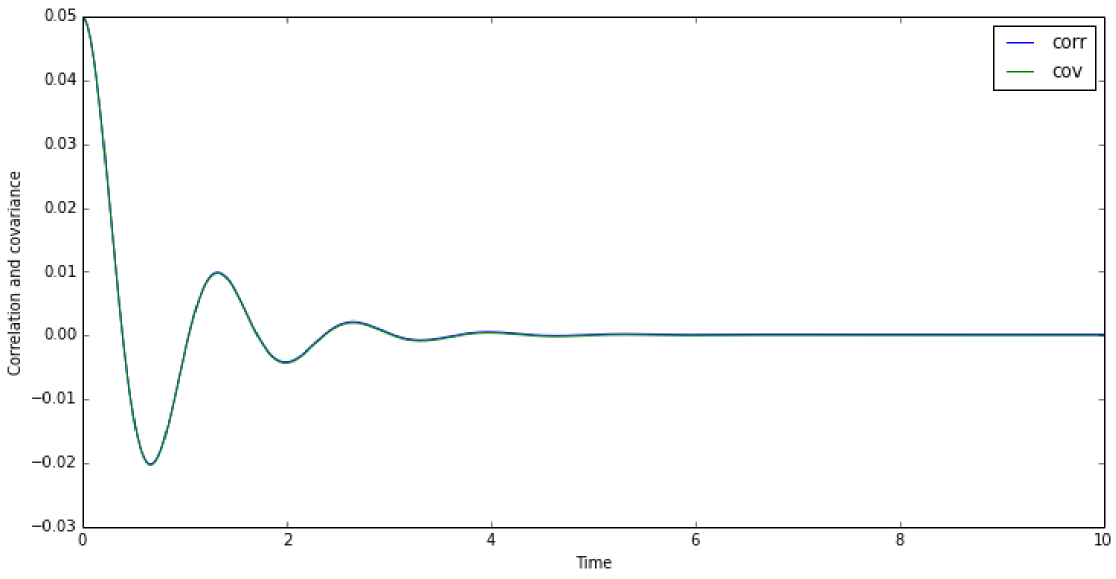
```
time elapsed (probcorr) = 0.23326349258422852
```

```
In [6]: start_time=time.time()
        corr  = esval(corrES,tlist);
        cov   = esval(covES,tlist);
        print('time elapsed (esval) = ' +str(time.time()-start_time))
```

```
time elapsed (esval) = 0.05180668830871582
```

1.2.1 Correlation function

```
In [7]: figure(figsize=(12, 6))
        plot(tlist,real(corr))
        plot(tlist,real(cov))
        xlabel('Time')
        ylabel('Correlation and covariance')
        legend(("corr", "cov"));
```

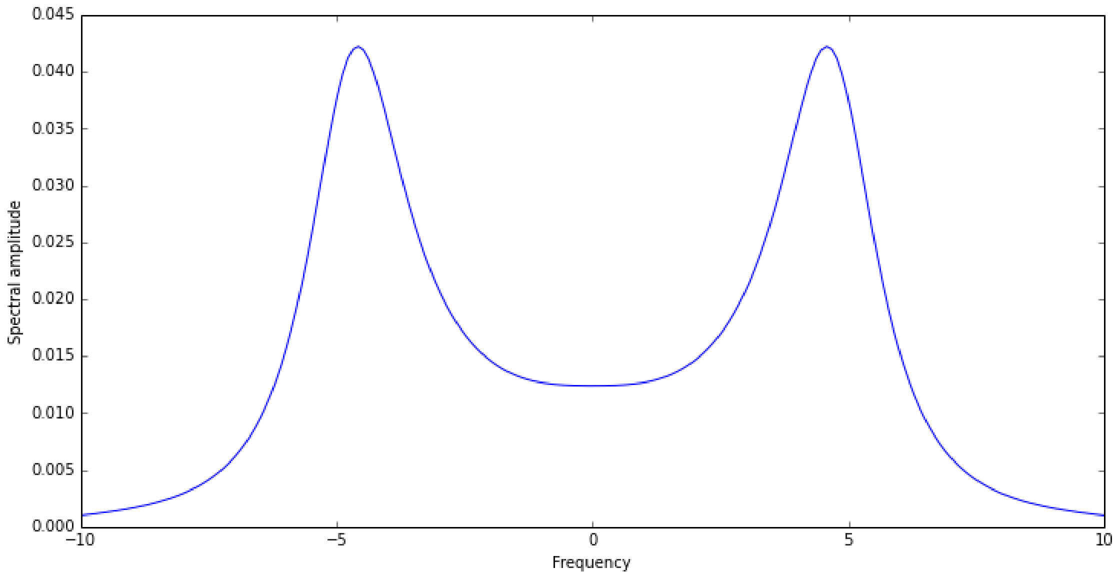


1.2.2 Spectrum

```
In [8]: start_time=time.time()
        wlist = linspace(-10,10,200);
        Ps = esspec(covES,wlist);
        print('time elapsed (esspec) = ' +str(time.time()-start_time))
```

```
time elapsed (esspec) = 0.008437156677246094
```

```
In [9]: figure(figsize=(12, 6))
        plot(wlist,real(Ps))
        xlabel('Frequency')
        ylabel('Spectral amplitude')
        show()
```



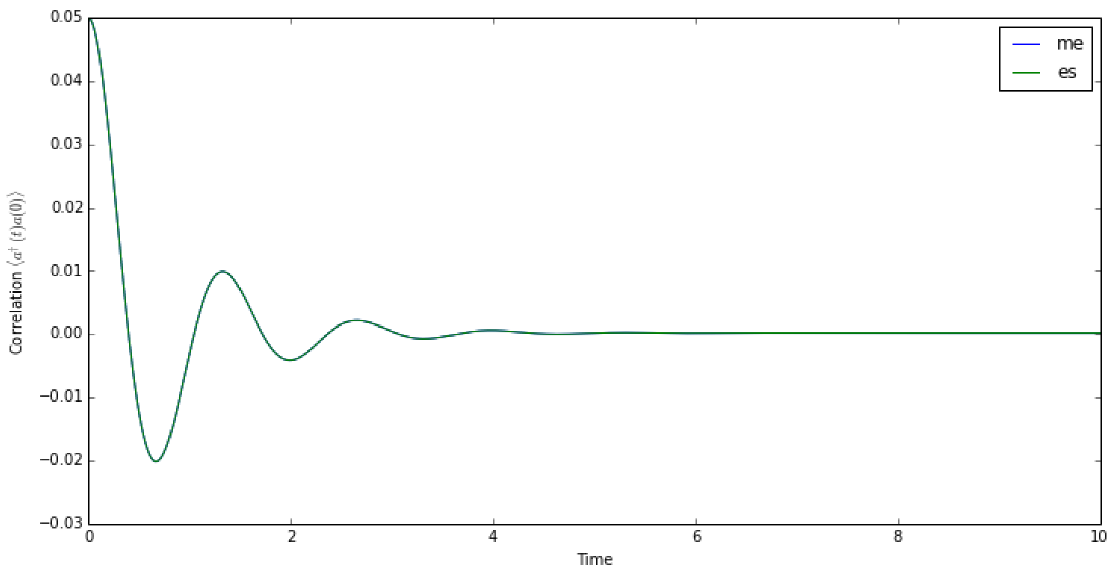
1.3 Comparison: Correlation function using correlation_ss with mesolve and eseries

In [10]:
```
def calc_correlation(E, kappa, gamma, g, wc, w0, wl, N, tlist):
    #
    # returns the two-time correlation of the intracavity field as exponential
    # series for the problem of a coherently driven cavity with a two-level atom
    #
    # E = amplitude of driving field, kappa = mirror coupling,
    # gamma = spontaneous emission rate, g = atom-field coupling,
    # wc = cavity frequency, w0 = atomic frequency, wl = driving field frequency,
    # N = size of Hilbert space for intracavity field (zero to N-1 photons)
    #

    # Define cavity field and atomic operators
    a  = tensor(destroy(N),qeye(2))
    sm = tensor(qeye(N),sigmam())

    # Hamiltonian
    H = (w0-wl)*sm.dag()*sm + (wc-wl)*a.dag()*a + 1j*g*(a.dag()*sm - sm.dag()*a) \
        + E*(a.dag()+a)

    # collapse operators
    C1=sqrt(2*kappa)*a
    C2=sqrt(gamma)*sm.dag()

    A = a

    corr_ode = correlation_ss(H, tlist, [C1, C2], A.dag(), A, solver="me")
    corr_es  = correlation_ss(H, tlist, [C1, C2], A.dag(), A, solver="es")

    print("real corr at 0 [me]:", corr_ode[0])
    print("real corr at 0 [es] :", corr_es[0])

    return corr_ode, corr_es
```

In [11]:
```
start_time=time.time()
corr1, corr2 = calc_correlation(E, kappa, gamma, g, wc, w0, wl, N, tlist)
print('time elapsed (probcorr) = ' +str(time.time()-start_time))
```

```
real corr at 0 [me]: (0.0499554840591+0j)
real corr at 0 [es] : (0.0499554840591-8.27026344783e-18j)
time elapsed (probcorr) = 0.3170664310455322
```

In [12]:
```
figure(figsize=(12, 6))
plot(tlist,real(corr1), tlist, real(corr2))
xlabel('Time')
ylabel(r'Correlation $\langle a^\dag(t)a(0)\rangle$')
legend(("me", "es"))
show()
```



1.4 Correlation functions and spectrum at two different coupling strengths

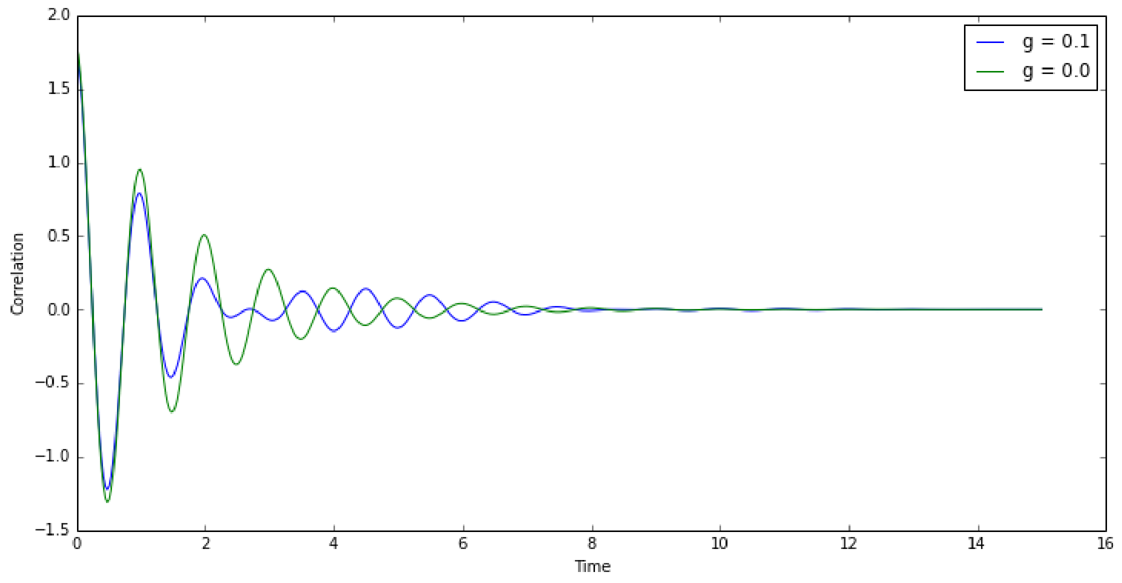
```
In [13]: def calc_spectrum(N, wc, wa, g, kappa, gamma, tlist, wlist):

             # Hamiltonian
             a  = tensor(destroy(N), qeye(2))
             sm = tensor(qeye(N), destroy(2))
             H = wc * a.dag() * a + wa * sm.dag() * sm + g * (a.dag() * sm + a * sm.dag())

             # collapse operators
             c_op_list = []

             n_th_a = 0.5
             rate = kappa * (1 + n_th_a)
             if rate > 0.0:
                 c_op_list.append(sqrt(rate) * a)

             rate = kappa * n_th_a
             if rate > 0.0:
                 c_op_list.append(sqrt(rate) * a.dag())

             rate = gamma
             if rate > 0.0:
                 c_op_list.append(sqrt(rate) * sm)

             A = a.dag() + a
             B = A

             # calculate the power spectrum
             corr = correlation_ss(H, tlist, c_op_list, A, B, solver='es')

             # calculate the power spectrum
```

```
        spec = spectrum_ss(H, wlist, c_op_list, A, B)

        return corr, spec
```

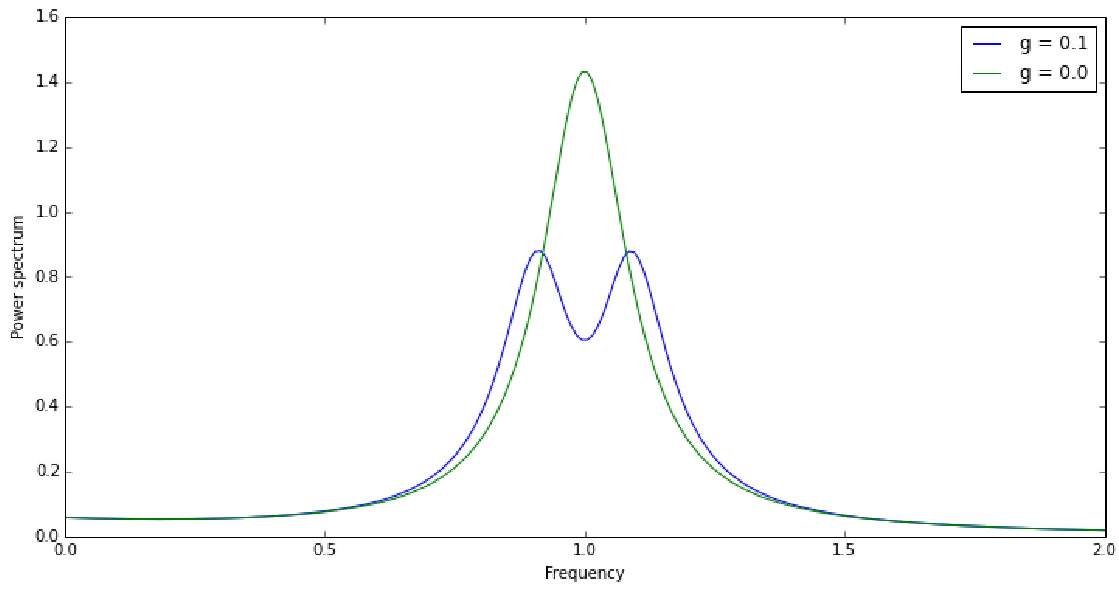
```
In [14]: N = 4                # number of cavity fock states
         wc = 1.00 * 2 * pi  # cavity frequency
         wa = 1.00 * 2 * pi  # atom frequency
         g  = 0.10 * 2 * pi  # coupling strength
         kappa = 1.0         # cavity dissipation rate
         gamma = 0.2         # atom dissipation rate

         wlist = linspace(0, 2*pi*2, 200)
         tlist = linspace(0, 15, 500)
```

```
In [15]: corr1, spec1 = calc_spectrum(N, wc, wa, g, kappa, gamma, tlist, wlist)
         corr2, spec2 = calc_spectrum(N, wc, wa, 0, kappa, gamma, tlist, wlist)
```

```
In [16]: figure(figsize=(12,6))
         plot(tlist,real(corr1), tlist, real(corr2))
         xlabel('Time')
         ylabel('Correlation')
         legend(("g = 0.1", "g = 0.0"));
```



```
In [17]: figure(figsize=(12,6))
         plot(wlist/(2*pi),abs(spec1), wlist/(2*pi), abs(spec2))
         xlabel('Frequency')
         ylabel('Power spectrum')
         legend(("g = 0.1", "g = 0.0"));
```



1.5 Versions

```
In [18]: from qutip.ipynbtools import version_table

         version_table()

Out[18]: <IPython.core.display.HTML at 0x7fdc068d5668>
```