example-atom-cavity-dynamics

June 25, 2014

1 QuTiP example: Dynamics of an atom-cavity system using three different solvers

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

```
In [3]: kappa = 2;
        gamma = 0.2;
        g  = 1;
        wc = 0;
        w0 = 0;
        wl = 0;
        N  = 4;
        E  = 0.5;
        tlist = linspace(0,10,200);
```

1.1.1 mesolve

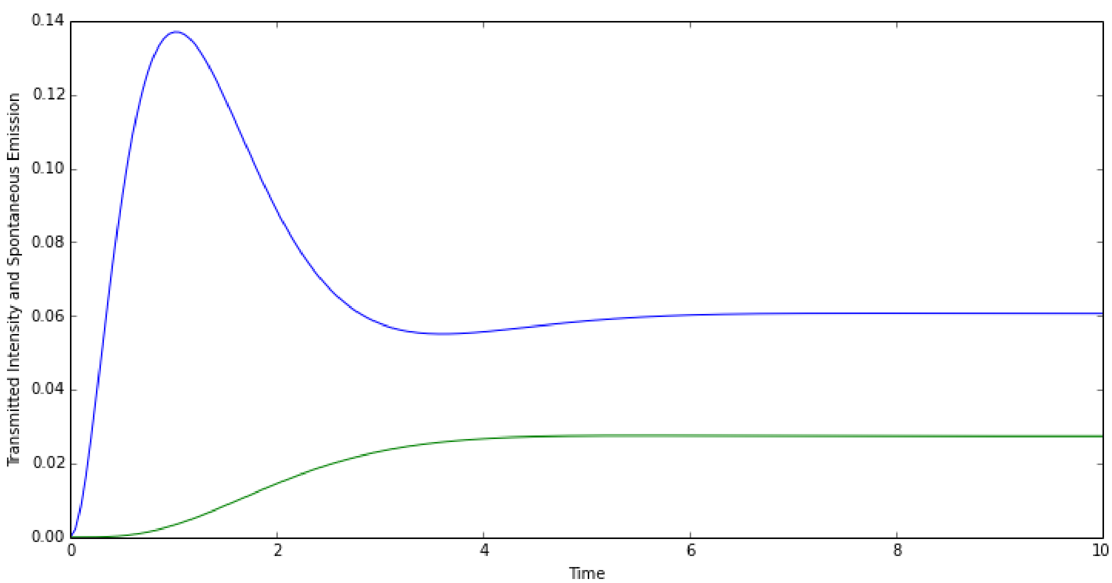
```
In [4]: def solve(E,kappa,gamma,g,wc,w0,wl,N,tlist):

            ida    = qeye(N)
            idatom = qeye(2)

            # Define cavity field and atomic operators
            a  = tensor(destroy(N),idatom)
            sm = tensor(ida,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
```

```
        C1dC1=C1.dag()*C1
        C2dC2=C2.dag()*C2

        #intial state
        psi0 = tensor(basis(N,0),basis(2,1))
        rho0 = psi0.dag() * psi0;

        # evolve and calculate expectation values
        output = mesolve(H, psi0, tlist, [C1, C2], [C1dC1, C2dC2, a])

        return output.expect[0], output.expect[1], output.expect[2]
```

```
In [5]: start_time=time.time()
        count1, count2, infield = solve(E,kappa,gamma,g,wc,w0,wl,N,tlist)
        print('time elapsed = ' +str(time.time()-start_time))
```

```
time elapsed = 0.054757118225097656
```

```
In [6]: figure(figsize=(12,6))
        plot(tlist,real(count1))
        plot(tlist,real(count2))
        xlabel('Time')
        ylabel('Transmitted Intensity and Spontaneous Emission');
```



1.1.2 eseries

```
In [ ]: def solve(E,kappa,gamma,g,wc,w0,wl,N,tlist):

            # 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
    C1dC1 = C1.dag() * C1
    C2dC2 = C2.dag() * C2

    #intial state
    psi0 = tensor(basis(N,0),basis(2,1))
    rho0 = ket2dm(psi0)

    # Calculate the Liouvillian
    L = liouvillian(H, [C1, C2])

    # Calculate solution as an exponential series
    start_time = time.time()
    rhoES = ode2es(L,rho0);
    print('time elapsed (ode2es) = ' + str(time.time()-start_time))

    # Calculate expectation values
    start_time = time.time()
    count1  = esval(expect(C1dC1,rhoES),tlist);
    count2  = esval(expect(C2dC2,rhoES),tlist);
    infield = esval(expect(a,rhoES),tlist);
    print('time elapsed (esval) = ' +str(time.time()-start_time))

    # alternative
    start_time = time.time()
    expt_list = essolve(H, psi0, tlist, [C1, C2], [C1dC1, C2dC2, a]).expect
    print('time elapsed (essolve) = ' +str(time.time()-start_time))

    return count1, count2, infield, expt_list[0], expt_list[1], expt_list[2]
```

In []:
```
start_time = time.time()
count1, count2, infield, count1_2, count2_2, \
infield_2 = solve(E,kappa,gamma,g,wc,w0,wl,N,tlist);
print('time elapsed = ' + str(time.time()-start_time))
```

In []:
```
figure(figsize=(12,6))
plot(tlist, real(count1), tlist, real(count1_2), '.')
plot(tlist, real(count2), tlist, real(count2_2), '.')
xlabel('Time')
ylabel('Transmitted Intensity and Spontaneous Emission');
```

1.1.3 mcsolve

In []:
```
ntraj = 500 #number of Monte-Carlo trajectories
# Hamiltonian
ida = qeye(N)
idatom = qeye(2)
a = tensor(destroy(N),idatom)
sm = tensor(ida,sigmam())
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
C1dC1 = C1.dag() * C1
C2dC2 = C2.dag() * C2
#intial state
psi0=tensor(basis(N,0),basis(2,1))
```

In [ ]:
```
start_time = time.time()
avg = mcsolve(H,psi0,tlist,[C1,C2],[C1dC1,C2dC2],ntraj)
elapsed_time = time.time() - start_time
print("elapsed time =", elapsed_time)
```

In [ ]:
```
figure(figsize=(12, 6))
plot(tlist,avg.expect[0],tlist,avg.expect[1],'--')
xlabel('Time')
ylabel('Photocount rates')
legend(('Cavity ouput', 'Spontaneous emission'));
```

1.2 Versions

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

version_table()
```