example-qubit-dynamics

June 25, 2014

1 QuTiP example: Single-Qubit Dynamics

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
```

In [3]:
```
def qubit_integrate(epsilon, delta, g1, g2, solver):

    H = epsilon / 2.0 * sigmaz() + delta / 2.0 * sigmax()

    # collapse operators
    c_ops = []

    if g1 > 0.0:
        c_ops.append(sqrt(g1) * sigmam())

    if g2 > 0.0:
        c_ops.append(sqrt(g2) * sigmaz())

    e_ops = [sigmax(), sigmay(), sigmaz()]

    if solver == "me":
        output = mesolve(H, psi0, tlist, c_ops, e_ops)
    elif solver == "es":
        output = essolve(H, psi0, tlist, c_ops, e_ops)
    elif solver == "mc":
        ntraj = 250
        output = mcsolve(H, psi0, tlist, ntraj, c_ops, [sigmax(), sigmay(), sigmaz()])
    else:
        raise ValueError("unknown solver")

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

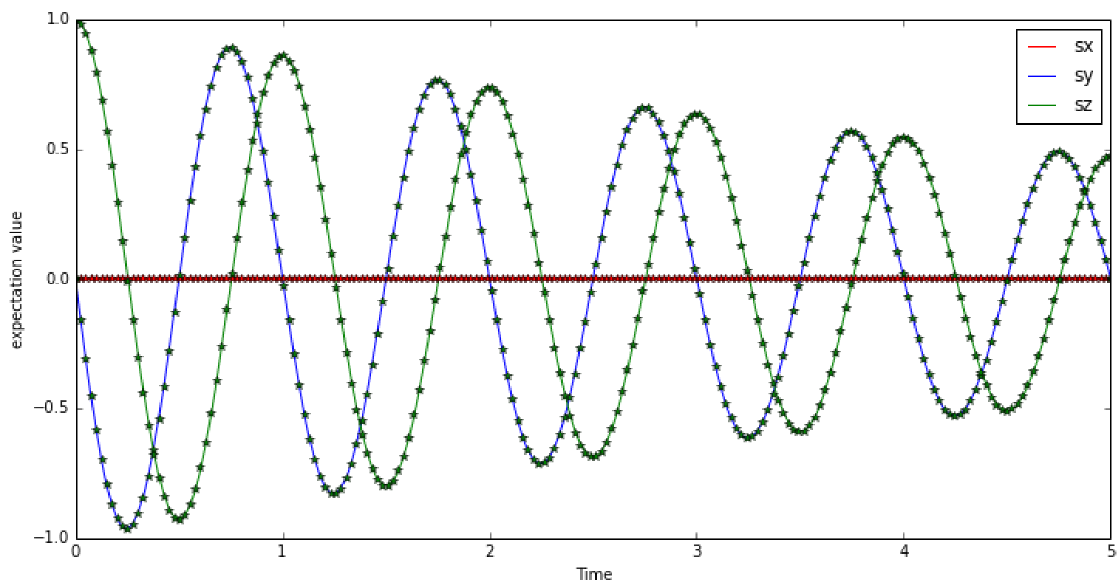
In [4]:
```
epsilon = 0.0 * 2 * pi   # cavity frequency
delta   = 1.0 * 2 * pi   # atom frequency
g2 = 0.15
g1 = 0.0
```

```
# intial state
psi0 = basis(2,0)

tlist = linspace(0,5,200)

# analytics
sx_analytic = zeros(shape(tlist))
sy_analytic = -sin(2*pi*tlist) * exp(-tlist * g2)
sz_analytic = cos(2*pi*tlist) * exp(-tlist * g2)
```

In [5]:
```
start_time = time.time()
sx1, sy1, sz1 = qubit_integrate(epsilon, delta, g1, g2, "me")
print('ME time elapsed = ' + str(time.time() - start_time))
```

```
ME time elapsed = 0.035593986511230347
```

In [6]:
```
figure(figsize=(12,6))
plot(tlist, real(sx1), 'r')
plot(tlist, real(sy1), 'b')
plot(tlist, real(sz1), 'g')
plot(tlist, sx_analytic, 'r*')
plot(tlist, sy_analytic, 'g*')
plot(tlist, sz_analytic, 'g*')
legend(("sx", "sy", "sz"))
xlabel('Time')
ylabel('expectation value');
```

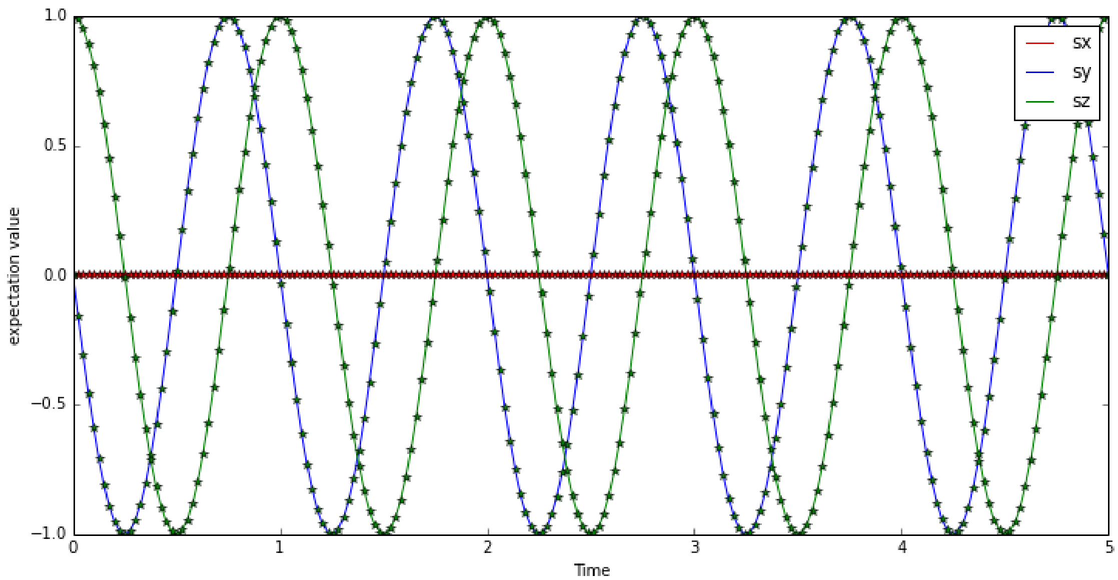


In [7]:
```
start_time = time.time()
sx2, sy2, sz2 = qubit_integrate(epsilon, delta, 0, 0, "me")
print('WF time elapsed = ' + str(time.time() - start_time))

# analytics
sx_analytic = zeros(shape(tlist))
```

```
        sy_analytic = -sin(2*pi*tlist)
        sz_analytic = cos(2*pi*tlist)
```

```
WF time elapsed = 0.08973956108093262
```

```
In [8]: figure(figsize=(12,6))
        plot(tlist, real(sx2), 'r')
        plot(tlist, real(sy2), 'b')
        plot(tlist, real(sz2), 'g')
        plot(tlist, sx_analytic, 'r*')
        plot(tlist, sy_analytic, 'g*')
        plot(tlist, sz_analytic, 'g*')
        legend(("sx", "sy", "sz"))
        xlabel('Time')
        ylabel('expectation value');
```



1.1 Bloch sphere

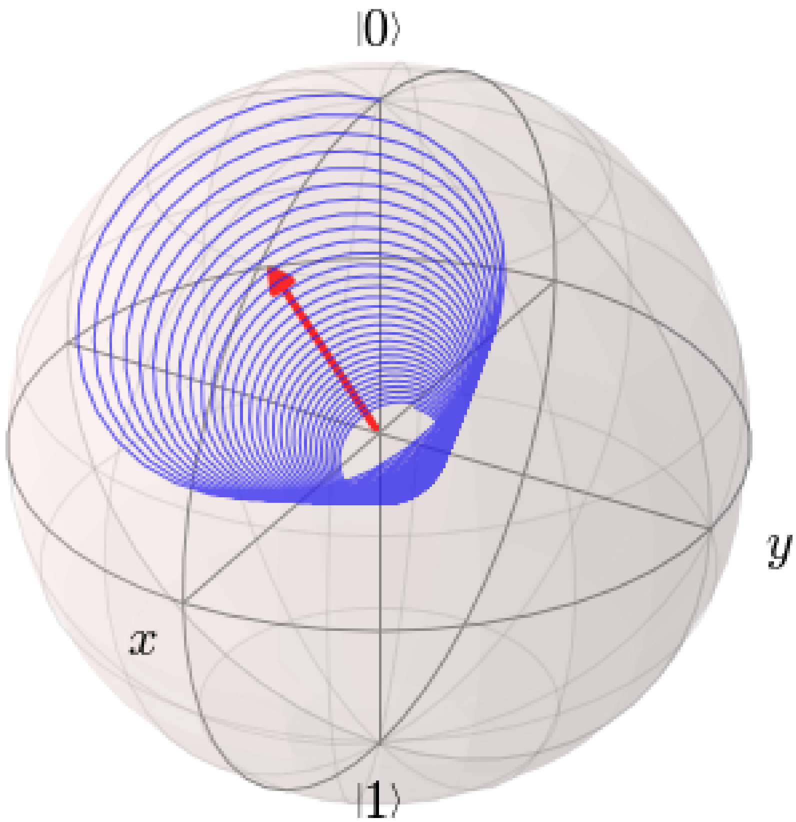
```
In [9]: w     = 1.0 * 2 * pi   # qubit angular frequency
        theta = 0.2 * pi       # qubit angle from sigma_z axis (toward sigma_x axis)
        gamma1 = 0.05      # qubit relaxation rate
        gamma2 = 0.02      # qubit dephasing rate
        # initial state
        a = 1.0
        psi0 = (a* basis(2,0) + (1-a)*basis(2,1))/(sqrt(a**2 + (1-a)**2))
        tlist = linspace(0,15,1000)
```

```
In [10]: def qubit_integrate(w, theta, gamma1, gamma2, psi0, tlist):
             # Hamiltonian
             sx = sigmax()
             sy = sigmay()
             sz = sigmaz()
             sm = sigmam()
```

```
        H = w * (cos(theta) * sz + sin(theta) * sx)
        # collapse operators
        c_op_list = []
        n_th = 0.5 # zero temperature
        rate = gamma1 * (n_th + 1)
        if rate > 0.0:
            c_op_list.append(sqrt(rate) * sm)
        rate = gamma1 * n_th
        if rate > 0.0:
            c_op_list.append(sqrt(rate) * sm.dag())
        rate = gamma2
        if rate > 0.0:
            c_op_list.append(sqrt(rate) * sz)
        # evolve and calculate expectation values
        output = mesolve(H, psi0, tlist, c_op_list, [sx, sy, sz])
        return output.expect[0], output.expect[1], output.expect[2]
```

In [11]:
```
sx, sy, sz = qubit_integrate(w, theta, gamma1, gamma2, psi0, tlist)
```

In [12]:
```
sphere=Bloch()
sphere.add_points([sx,sy,sz], meth='l')
sphere.vector_color = ['r']
sphere.add_vectors([sin(theta),0,cos(theta)])
sphere.show()
```



1.2 Versions

```
In [13]: from qutip.ipynbtools import version_table

         version_table()
```

Out[13]: <IPython.core.display.HTML at 0x7fdbbe4ba400>