Solving Schrodinger's equation: Coherent states

In this exercise, we'll be exploring coherent states -- a solution to the quantum harmonic oscillator of the evolution of a displaced ground-state wavefunction.

Animations in ipython notebooks appear to conflict with inline. So start this notebook with "ipython notebook --pylab" (without the final 'inline').

We start by importing the fft packages and defining the constants. We'll use McEuen's buckyballs: $m = 60 m_C = 60 * 12 m_p$ in a harmonic well with frequency one THz:

In [ ]:
```
from scipy import fft, ifft
from matplotlib import animation

hbar = ;
omega = ;
protonMass = ;
m = ;
a0 = ; # RMS width of Gaussian
```

We reserve space for a complex ψ on a lattice of Np points x, from -L/2 to L/2, and Nt times separated by dt = Period / 100.

In [12]:
```
L = 30 * a0;
Np = ;
dx = ;
x =
Period = ;
dt = ;
T = 2 * Period;
Nt = int(round(T/dt));
times =
psi = (1.+0.j) * zeros((Nt, Np))
```

We solve Schrodinger's equation by using the Baker-Campbell-Hausdorff formula to approximate the time evolution operator, applying the potential energy for dt/2, then the kinetic energy by dt, and then the potential energy by dt/2:

$$U(dt) = U_{pot}(dt/2) U_{kin}(dt) U_{pot}(dt/2)$$

$$\exp(-i(p^2/2m + V(x))dt/\hbar) \sim \exp(-iV(x)dt/(2\hbar)) \exp(-i(p^2/2m)dt/\hbar) \exp(-iV(x)dt/2\hbar)$$

If $\psi(x)$ is expressed in real space, applying the potential energy is just multiplication:

In [13]:
```
V = ;
```

```
UpotDtOver2 = ;
```

If $\tilde{\psi}(k)$ is expressed in Fourier space, applying the kinetic energy is multiplication, as we saw for the free particle. Again, the FFT returns $\tilde{\psi}(k)$ at points separated by $dk = 2\pi/L$:
$k = [0, dk, 2dk, \cdots, (Np/2)k, -(Np/2-1)k, \cdots, -2dk, -dk]$

In [14]:
```
dk = ;
k = ;
for j in range(0,Np/2+1):

for


k2 =
UkinTildeDt =
```

Now we define the initial wavefunction ψ_0. (The zero tells us this is the zero'th time-step.) Let's start in the ground state:

In [15]:
```
psi[0] = (m * omega/(pi * hbar))**(1./4.) * exp(-m * omega * x**2/(2*hbar))
psiMax = abs(max(psi[0]))
```

We now evolve in small time steps dt, storing $\psi_n(x) = \psi(x, ndt)$.

In [16]:
```
for n in range(1,len(times)):
    psi[n] =  * ifft( *fft(  * psi[n-1]))
```

Plot the real part, the imaginary part, and the absolute value of psi at time $t = \text{Period}/5$, hence $n = Nt/10$. (Why don't we plot the probability density $|\psi(x)|^2$ along with the real and imaginary parts?)

In [17]:
```
plot(x, psi[Nt/10].real, x, psi[Nt/10].imag, x, abs(psi[Nt/10]));
```

We animate the evolution, plotting $\psi_n(x)$ (real part, imaginary part, and absolute value).

Set up the figure, axis, and the two lines. Set up an init function and an animation function. See Matplotlib Animation Example, by Jake Vanderplas (vanderplas@astro.washington.edu, http://jakevdp.github.com).

Did you start the notebook without the "inline" option?

In [26]:
```
fig = figure()
ax = axes( xlim=(-L/2,L/2), ylim = (-psiMax,psiMax) )
realLine, imagLine, absLine = ax.plot(x, psi[0].real, x, psi[0].imag, x, abs(
psi[0]))

def animate(n):
    realLine.set_data(x, psi[n].real)
    imagLine.set_data(x, psi[n].imag)
    absLine.set_data(x, abs(psi[n]))
    return realLine, imagLine, absLine

anim = animation.FuncAnimation(fig, animate, frames=Nt, interval=20)
show()
```

Note that the probability distribution doesn't evolve with time; the ground state is a stationary state. What happens to the real and imaginary parts? Why?

Now let's evolve from an initial condition that is the ground state displaced by 10 a0:

In [19]:
```
x0 = 
psi[0] = 
for n in :
    psi[n] = 
```

Again, plot the evolution after $t = \text{Period}/5$.

In [20]:
```
plot(...);
```

Animate. How does the probability density evolve, compared to a classical mass in the same harmonic oscillator?

In [21]:
```
fig = figure()
ax = axes( xlim=(-L/2,L/2), ylim = (-psiMax,psiMax) )
realLine, imagLine, absLine = ax.plot(x, psi[0].real, x, psi[0].imag, x, abs(
psi[0]))

def animate(n):
    realLine.set_data(x, psi[n].real)
    imagLine.set_data(x, psi[n].imag)
    absLine.set_data(x, abs(psi[n]))
    return realLine, imagLine, absLine

anim = animation.FuncAnimation(fig, animate, frames=Nt, interval=20)
show()
```

Translating the ground state of the harmonic oscillator, either in position or in momentum, yields a coherent state. Coherent states are great tools for harmonic systems (like optics). They travel in phase space like classical particles. One can show that they are eigenstates of the annihilation operator. My graduate quantum course was filled with them (perhaps because the instructor, Roy Glauber, got the Nobel prize for developing them).