Computing Probabilities in time for the Infinite Square Well

In [1]:
```
import vpython as vp
import numpy as np

vp.canvas()

N=20                            # how many fourier coefficients?
NA=40                           # how many arrows?
NA2=int(NA/2)                   # halfway index (NA/2)
a=1.0                           # range of x is 1 unit
x = np.linspace(0, a, NA)          # NA locations from 0 to a
sleeptime=0.2                   # how long to sleep
framerate=50                    # max framerate
E_1=1.0                         # work in units of the ground state energy (i.e., E_2=4*E_1 etc.)
hbar=1.0                        # work in units where hbar=1 (so omega_1 = E_1/hbar = 1.0)

n = np.arange(1,N+1)
omega = (n**2)

coefs = (1/n)*(1.0-np.cos(n*np.pi/2.0)) # compute all the fourier coefficients in one go.

def SetArrowFromCN( cn, a):
    """
    SetArrowWithCN takes a complex number  cn  and an arrow object  a .
    It sets the  y  and  z  components of the arrow s axis to the real
    and imaginary parts of the given complex number.

    Just like Computing Project 1, except y and z for real/imag.
    """
    a.axis.y = cn.real
    a.axis.z = cn.imag
    a.axis.x = 0.0

gr = vp.gcurve(color=vp.color.black)
gx = vp.gcurve(color=vp.color.blue)
gxp = vp.gcurve(color=vp.color.red)
gxm = vp.gcurve(color=vp.color.red)

psi=np.zeros(len(x),np.complex)

for m in n:
    psi += coefs[m-1]*np.sqrt(2.0/a)*np.sin(m*np.pi*x/a)  # put together initial wavefuncton

pTot = (abs(psi)**2).sum()
psi = psi/np.sqrt(pTot) # normalize!

alist = []
for i in range(NA):
    alist.append(vp.arrow(pos=vp.vec(6*(x[i]-(a/2.0)),0,0), shaftwidth=4*a/NA, color=vp.color.red))
    SetArrowFromCN(psi[i],alist[i])

pTot = (abs(psi)**2).sum()
psi = psi/np.sqrt(pTot) # normalize!

pLeft = (abs(psi[:NA2])**2).sum()
pRight = (abs(psi[NA2:])**2).sum()

print ("At t=0.0...")
print ("The probability of being on the left is:", pLeft)
print ("The probability of being on the right is:", pRight)
print ("The total is:", pLeft + pRight)

t = 0.0
dt = 2*np.pi/1000.0

def DisplayWF(t, scaleFactor=5):
    """
    Update the display at a single timestep at time t.
    """
    psi1 = np.zeros(len(x),complex)

    for m in n:
        psi1 += coefs[m-1]*np.sin(m*np.pi*x/a) * np.exp(-1j * omega[m-1] * t) # add time dependence her
e!

    pTot1 = (abs(psi1)**2).sum()
    psi1 = psi1/np.sqrt(pTot1) # normalize!

    for i in range(NA):
        SetArrowFromCN(scaleFactor*psi1[i], alist[i])              # update the arrows


    pLeft1 = (abs(psi1[:NA2])**2).sum()

    gr.plot(pos=(t, pLeft1))

    mean = (x*abs(psi1)**2).sum()

    gx.plot(pos=(t, mean))

    meansquare = ((x**2)*abs(psi1)**2).sum()
    sigma = np.sqrt(meansquare - (mean**2))
    sigmapos = mean + sigma
    sigmaneg = mean - sigma


    gxp.plot(pos=(t, sigmapos))

    gxm.plot(pos=(t, sigmaneg))


    #
    # now compute probabilities, expectation values, etc. and graph in this space
    #
```

```
At t=0.0...
The probability of being on the left is: 0.9965601990196631
The probability of being on the right is: 0.003439800980336981
The total is: 1.0
```

In [2]:
```
t = 0.0
dt = 2*np.pi/1000.0

status = {'go':False}

def Toggle(b):
    status['go'] = not status['go']

vp.button( bind=Toggle, text='Toggle' )

print("Starting Condition Click Toggle to begin")

while status['go'] == False:
    vp.rate(framerate)

print("OK here we go!")

while status['go'] and t<np.pi:
    vp.rate(framerate)
    DisplayWF(t)
    t+=dt

print("Done with first step")
status['go'] = False

while status['go'] == False:
    vp.rate(framerate)

print("OK! Keep going")

while status['go'] and t<2*np.pi:
    vp.rate(framerate)
    DisplayWF(t)
    t+=dt

print("Done!")
```

```
Starting Condition Click Toggle to begin
OK here we go!
Done with first step
OK! Keep going
Done!
```

Expectation.png

Snapshot.png

Questions

1.

After one cycle of the ground state energy, the probability of being on the left half returns to 1 because all other excited energy states return to its original position at t = 0 after one ground state cycle. The first excited goes around 4 times, the second, 9 times, the third 16 times but they still return to their original state as their frequencies are just multiples of the frequency of the ground state. Therefore, the wavefunction returns to its original position.

2.

When the ground state completes half a cycle, the wave function now becomes reflected on the right half. The probability of being on the left half now becomes zero and the probability on being on the right half is one. We can see that after we press the toggle button once for a cycle of $/pi$.

3.

Any wave function that gives a constant value from 0 to $\frac{a}{2}$ would give something similar as this wave function and it also starts the particle at the left half but a function that depends on x from 0 to $\frac{a}{2}$ might bring up something different.