HO3d

January 1, 2018

0.0.1 3-Dimensional Harmonic Oscillator using Finite Differences

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

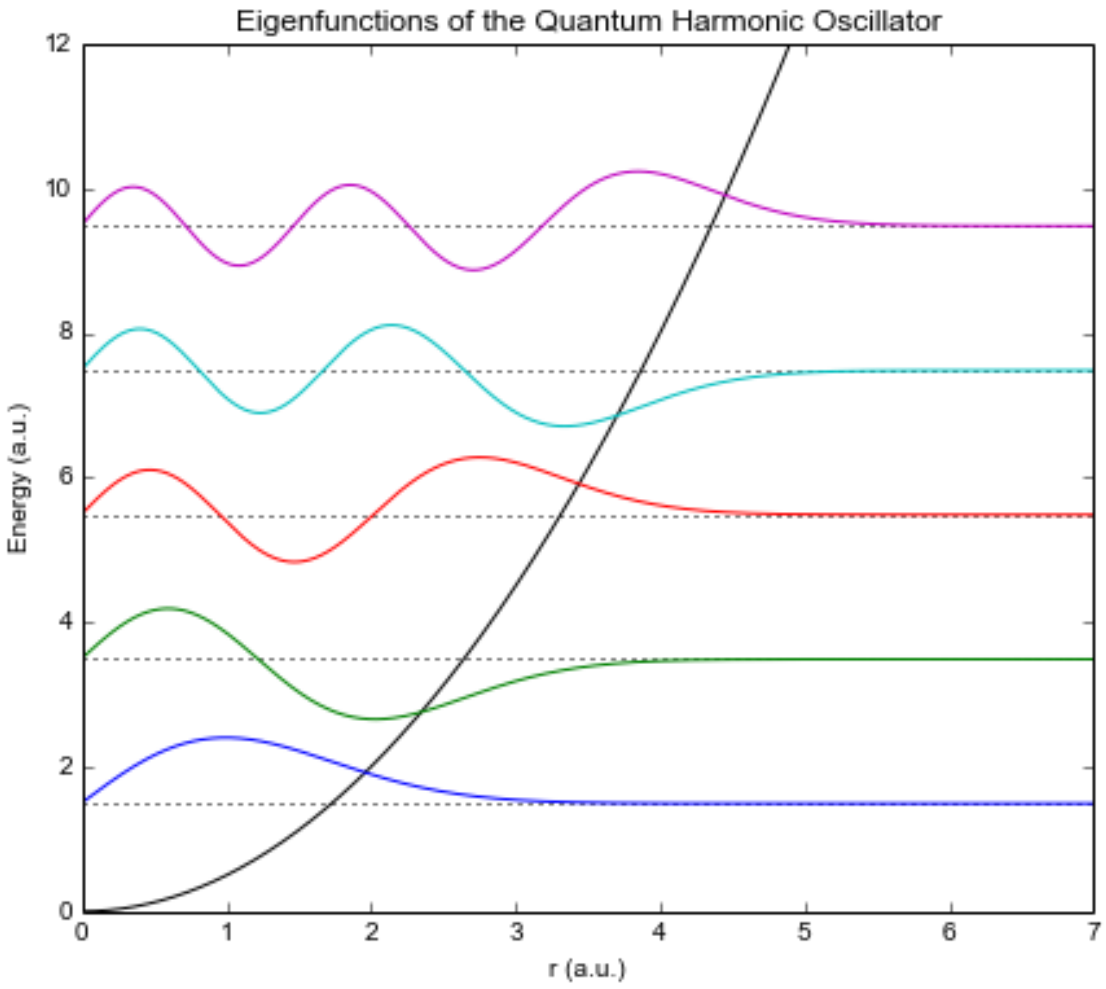
In [2]:
```
from numpy import identity

def Laplacian(x):
    h = x[1]-x[0] # assume uniformly spaced points
    n = len(x)
    M = -2*identity(n,'d')
    for i in range(1,n):
        M[i,i-1] = M[i-1,i] = 1
    return M/h**2
```

In [33]:
```
from numpy import sqrt

# Normalización de las funciones

def Normalizate(U,x):

    h = x[1]-x[0] # assume uniformly spaced points
    n = len(x)


    for j in range(0,n):
        suma = 0.0
        for i in range(1,n):
             suma = suma + U[i,j]**2

        suma = suma*h
        rnorm = 1/sqrt(suma)
#        print j,' integral (sin normalizar) =',rnorm

#       Normalization
        rsign = 1
        if U[1,j] < 0:
            rsign = -1

        rnorm = rnorm * rsign
        for i in range(0,n):
            U[i,j] = U[i,j]*rnorm

#       Check Normalization
#        suma = 0.0
```

```
        #         for i in range(1,n):
        #             suma = suma + U[i,j]**2
        #         print j,' suma=',suma*h

            return
```

In [34]:
```
        from numpy import diag,  linspace, array
        from numpy.linalg import eigh
        from matplotlib.pyplot import axhline, xlabel, ylabel, plot, axis, \
                                       figure, title, show

        nfunctions = 5

        # array definitions
        nsize = 1000
        xmin=0
        xmax=7
        x = linspace(xmin,xmax,nsize)
        T = array([nsize,nsize])
        V = array([nsize,nsize])
        H = array([nsize,nsize])
        E = array([nsize])

        # Oscillator Data
        m = 1.0
        omega = 1.0

        # Kinetic (T) and Potential (V)
        T = (-0.5/m)*Laplacian(x)
        V = 0.5*(omega**2)*(x**2)

        # Hamiltonian
        H =  T + diag(V)

        # Eigenvalues (E) and Eigenvectors (U)
        E,U = eigh(H)

        #define plot size in inches (width, height) & resolution(DPI)
        fig = figure(figsize=(7, 6), dpi=100)

        # Plot the Harmonic potential
        plot(x,V,color='k')

        # Normalization
        Normalizate(U,x)

        # Plot wavefunctions

        for i in range(nfunctions):
            # For each of the first few solutions, plot the energy level:
            axhline(y=E[i],color='k',ls=":")
            # as well as the eigenfunction, displaced by the energy level
            # so they don't all pile up on each other:
            plot(x,U[:,i]+E[i])
```

```
axis([xmin,xmax,0,12])
title("Eigenfunctions of the Quantum Harmonic Oscillator")
xlabel("r (a.u.)")
ylabel("Energy (a.u.)")
show()
```



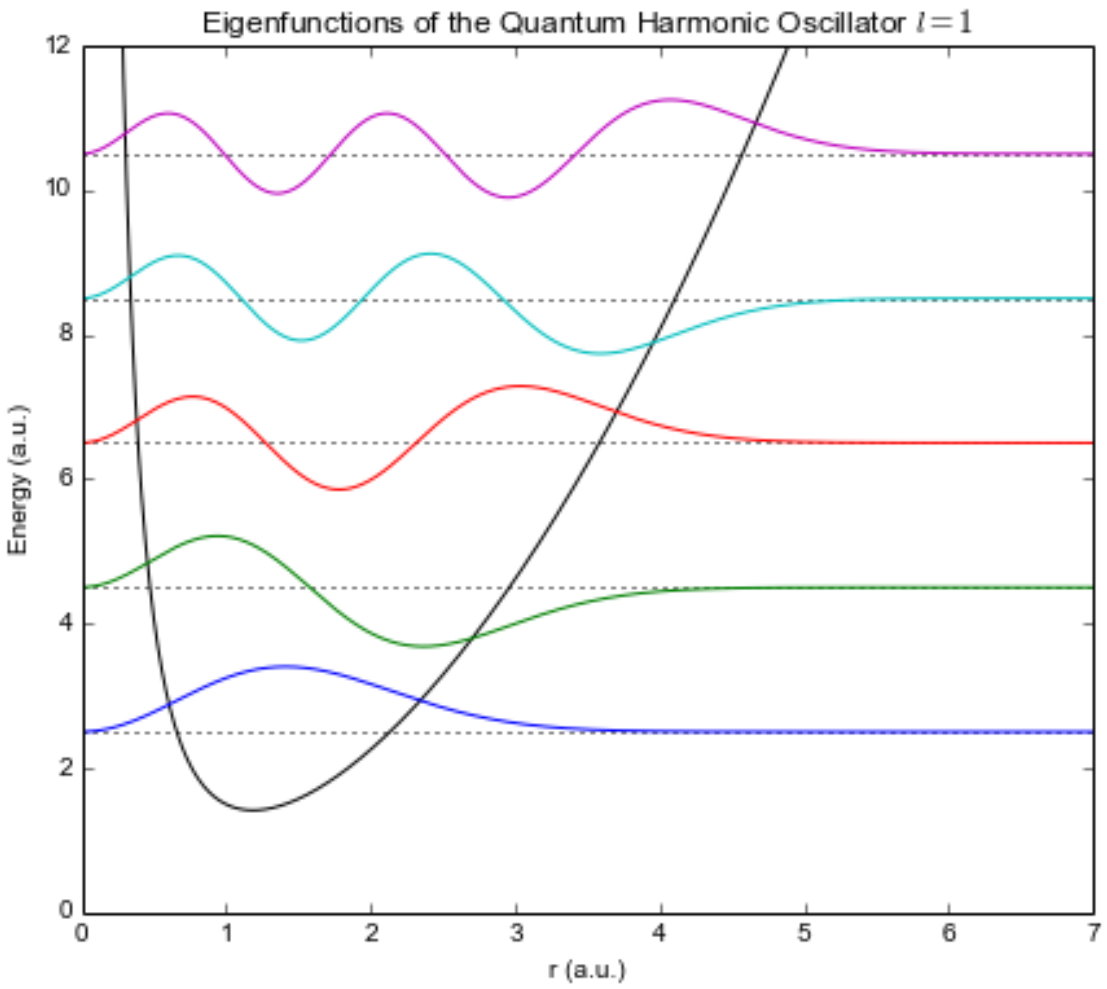

```
In [35]: print E[0],E[1],E[2],E[3],E[4]
         A = E-3/2.
         print A[0],A[1],A[2],A[3],A[4]
```

```
1.49210498492 3.48812166338 5.4851017134 7.4825513343 9.48028526285
-0.007895015082 1.98812166338 3.9851017134 5.9825513343 7.98028526285
```

0.0.2 l=1

```
In [38]: nsize = 1000
         xmin=0
         xmax=7
         dx = (1.*xmax-xmin)/nsize
         xmin = dx
```

```
x1 = linspace(xmin,xmax,nsize)

# Kinetic (T) and Potential (V)
T = (-0.5/m)*Laplacian(x1)
l=1
V1 = 0.5*(omega**2)*(x1**2) + l*(l+1)/(2*x1**2)

# Hamiltonian
H =  T + diag(V1)

# Eigenvalues (E) and Eigenvectors (U)
E1,U1 = eigh(H)

#define plot size in inches (width, height) & resolution(DPI)
fig = figure(figsize=(7, 6), dpi=100)

# Plot the Harmonic potential
plot(x1,V1,color='k')

# Normalization
Normalizate(U1,x)

# Plot wavefunctions

for i in range(nfunctions):
    # For each of the first few solutions, plot the energy level:
    axhline(y=E1[i],color='k',ls=":")
    # as well as the eigenfunction, displaced by the energy level
    # so they don't all pile up on each other:
    plot(x1,U1[:,i]+E1[i])
axis([xmin,xmax,0,12])
title("Eigenfunctions of the Quantum Harmonic Oscillator $l=1$")
xlabel("r (a.u.)")
ylabel("Energy (a.u.)")
show()
```



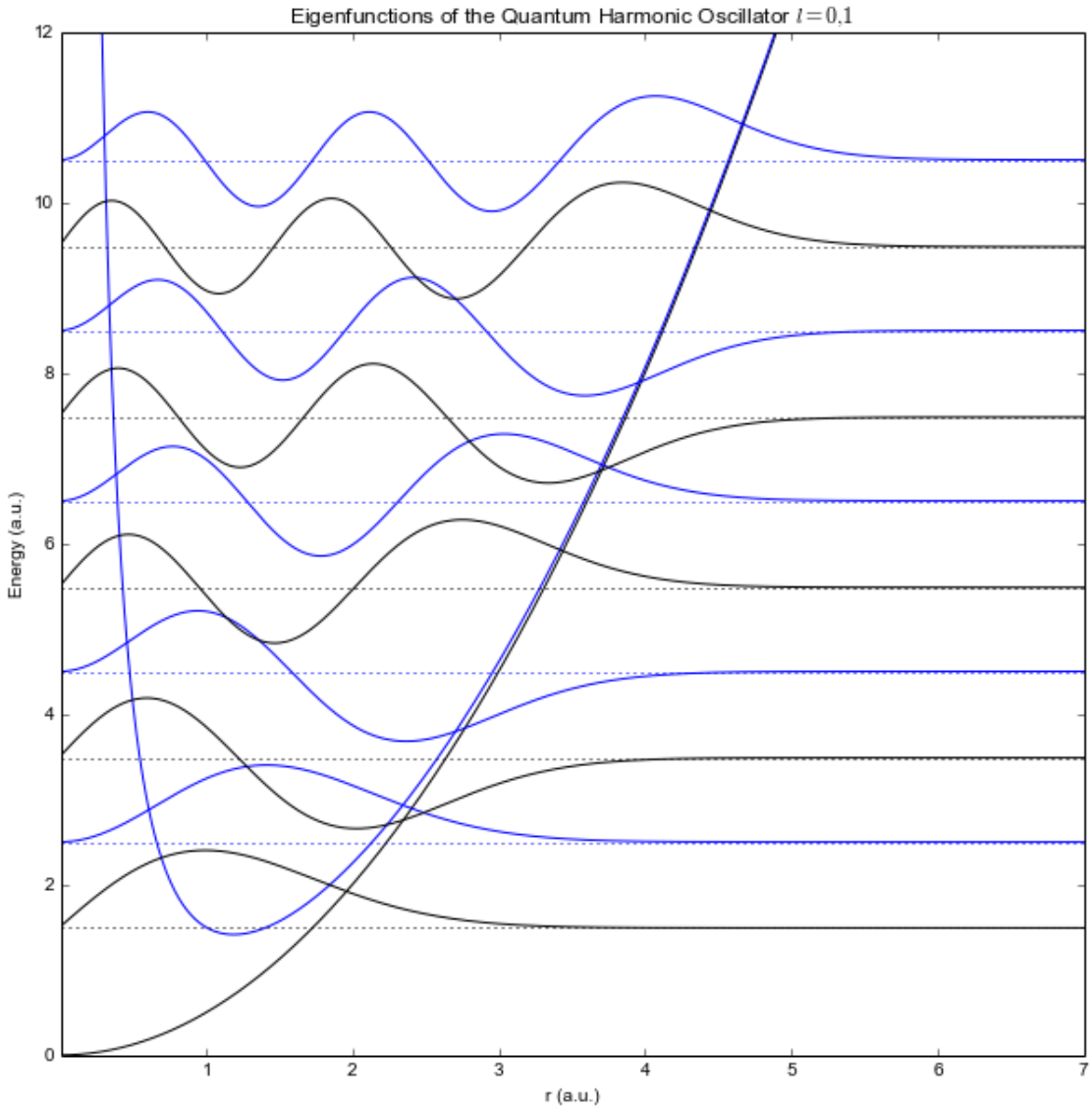

```
In [70]: print E1[0],E1[1],E1[2],E1[3],E1[4]
         A = E1-3/2.
         print A[0],A[1],A[2],A[3],A[4]
```

```
2.49999030207 4.49996049351 6.49990618433 8.49982737439 10.4997240844
0.999990302069 2.99996049351 4.99990618433 6.99982737439 8.99972408441
```

0.0.3 All Together

```
In [59]: #define plot size in inches (width, height) & resolution(DPI)
         fig = figure(figsize=(10, 10), dpi=100)

         # Plot the Harmonic potential
         plot(x,V,color='k')
         plot(x1,V1,color='b')

         for i in range(nfunctions):
             axhline(y=E[i],color='k',ls=":")
             plot(x,U[:,i]+E[i], color='k')
```

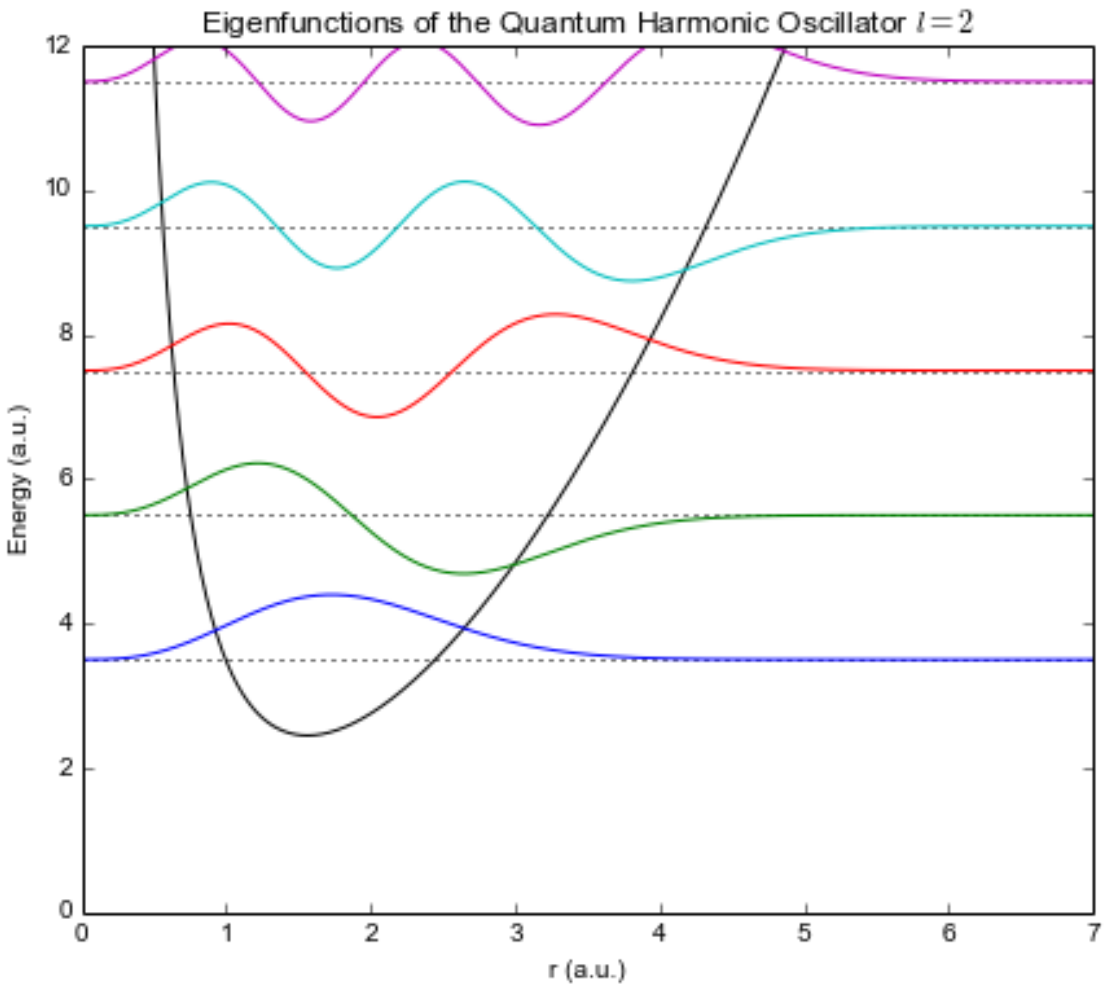
```
    axhline(y=E1[i],color='b',ls=":")
    plot(x1,U1[:,i]+E1[i], color='b')
axis([xmin,xmax,0,12])
title("Eigenfunctions of the Quantum Harmonic Oscillator $l=0,1$")
xlabel("r (a.u.)")
ylabel("Energy (a.u.)")
show()
```



0.0.4 l=2

In [44]:
```
nsize = 1000
xmin=0
xmax=7
dx = (1.*xmax-xmin)/nsize
xmin = dx
```

```
x1 = linspace(xmin,xmax,nsize)

# Kinetic (T) and Potential (V)
T = (-0.5/m)*Laplacian(x1)
l=2
V2 = 0.5*(omega**2)*(x1**2) + l*(l+1)/(2*x1**2)

# Hamiltonian
H =  T + diag(V2)

# Eigenvalues (E) and Eigenvectors (U)
E2,U2 = eigh(H)

#define plot size in inches (width, height) & resolution(DPI)
fig = figure(figsize=(7, 6), dpi=100)

# Plot the Harmonic potential
plot(x1,V2,color='k')

# Normalization
Normalize(U2,x)

# Plot wavefunctions

for i in range(nfunctions):
    # For each of the first few solutions, plot the energy level:
    axhline(y=E2[i],color='k',ls=":")
    # as well as the eigenfunction, displaced by the energy level
    # so they don't all pile up on each other:
    plot(x1,U2[:,i]+E2[i])
axis([xmin,xmax,0,12])
title("Eigenfunctions of the Quantum Harmonic Oscillator $l=2$")
xlabel("r (a.u.)")
ylabel("Energy (a.u.)")
show()
```



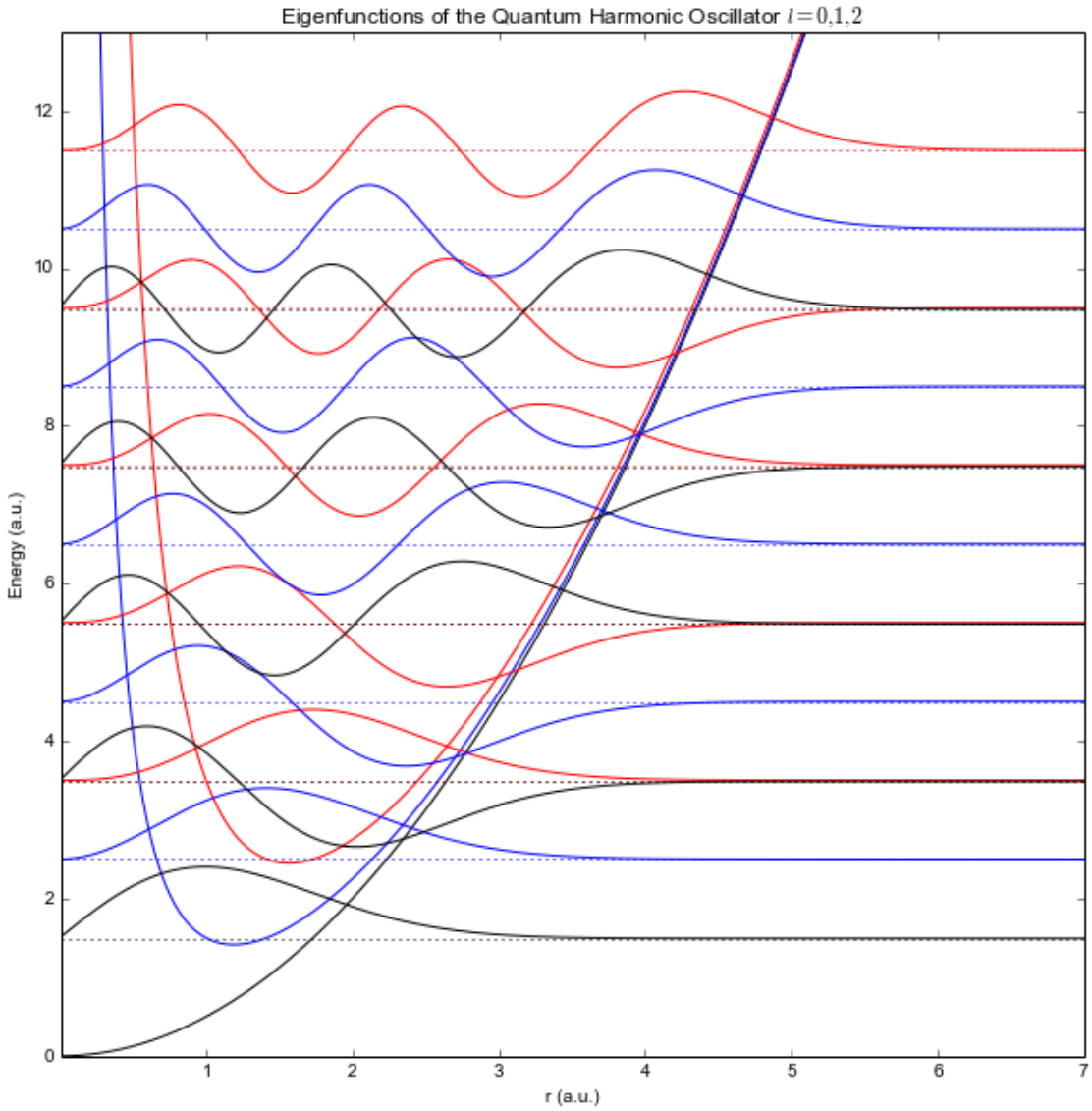

```
In [45]: print E2[0],E2[1],E2[2],E2[3],E2[4]
         A = E2-3/2.
         print A[0],A[1],A[2],A[3],A[4]
```

```
3.49999234373 5.49996521846 7.49991359255 9.49983746789 11.4997369697
1.99999234373 3.99996521846 5.99991359255 7.99983746789 9.99973696968
```

0.0.5 All Together

```
In [60]: #define plot size in inches (width, height) & resolution(DPI)
         fig = figure(figsize=(10, 10), dpi=100)

         # Plot the Harmonic potential
         plot(x,V,color='k')
         plot(x1,V1,color='b')
         plot(x1,V2,color='r')

         for i in range(nfunctions):
             axhline(y=E[i],color='k',ls=":")
```

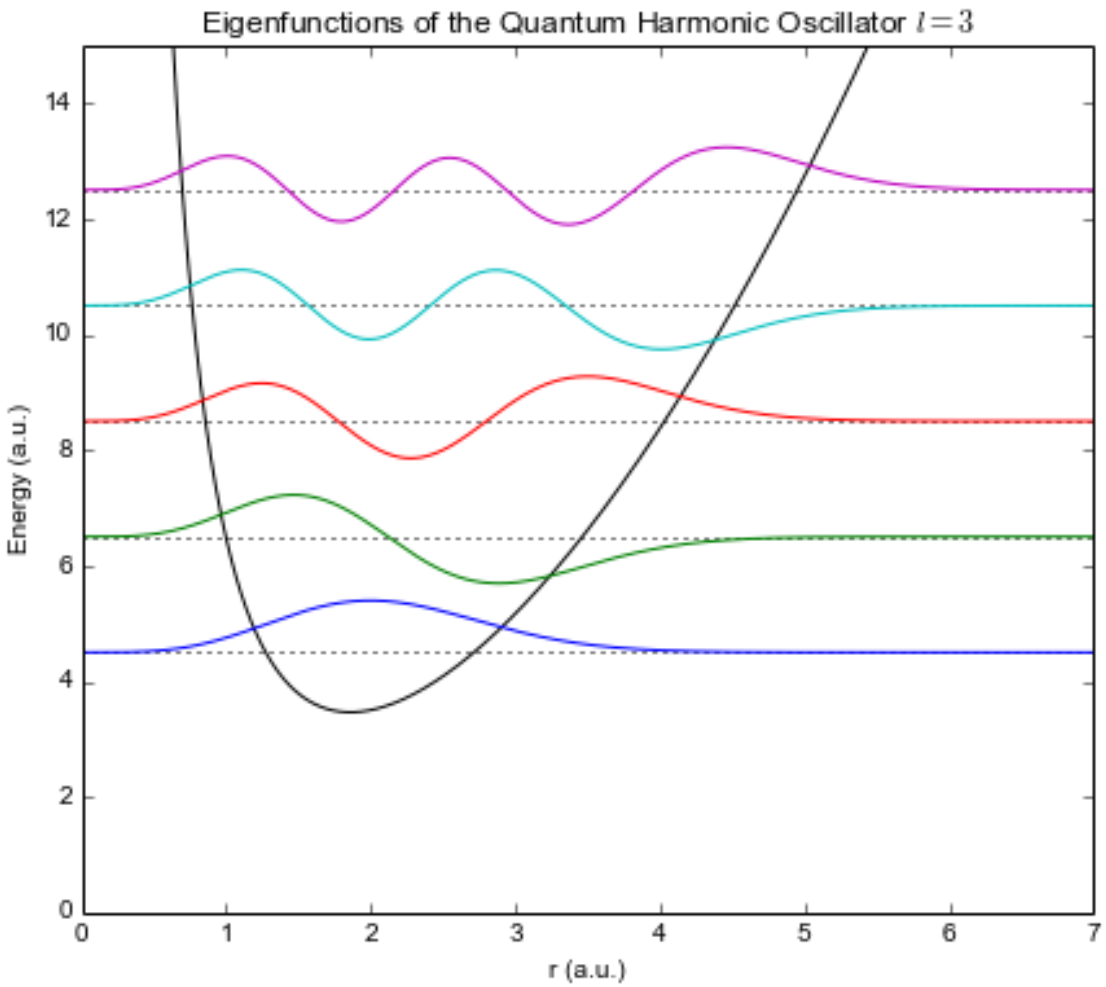
```
        plot(x,U[:,i]+E[i], color='k')
        axhline(y=E1[i],color='b',ls=":")
        plot(x1,U1[:,i]+E1[i], color='b')
        axhline(y=E2[i],color='r',ls=":")
        plot(x1,U2[:,i]+E2[i], color='r')
    axis([xmin,xmax,0,13])
    title("Eigenfunctions of the Quantum Harmonic Oscillator $l=0,1,2$")
    xlabel("r (a.u.)")
    ylabel("Energy (a.u.)")
    show()
```



0.0.6 l=3

In [51]:
```
nsize = 1000
xmin=0
```

```
xmax=7
dx = (1.*xmax-xmin)/nsize
xmin = dx
x1 = linspace(xmin,xmax,nsize)

# Kinetic (T) and Potential (V)
T = (-0.5/m)*Laplacian(x1)
l=3
V3 = 0.5*(omega**2)*(x1**2) + l*(l+1)/(2*x1**2)

# Hamiltonian
H =  T + diag(V3)

# Eigenvalues (E) and Eigenvectors (U)
E3,U3 = eigh(H)

#define plot size in inches (width, height) & resolution(DPI)
fig = figure(figsize=(7, 6), dpi=100)

# Plot the Harmonic potential
plot(x1,V3,color='k')

# Normalization
Normalizate(U3,x)

# Plot wavefunctions

for i in range(nfunctions):
    # For each of the first few solutions, plot the energy level:
    axhline(y=E3[i],color='k',ls=":")
    # as well as the eigenfunction, displaced by the energy level
    # so they don't all pile up on each other:
    plot(x1,U3[:,i]+E3[i])
axis([xmin,xmax,0,15])
title("Eigenfunctions of the Quantum Harmonic Oscillator $l=3$")
xlabel("r (a.u.)")
ylabel("Energy (a.u.)")
show()
```



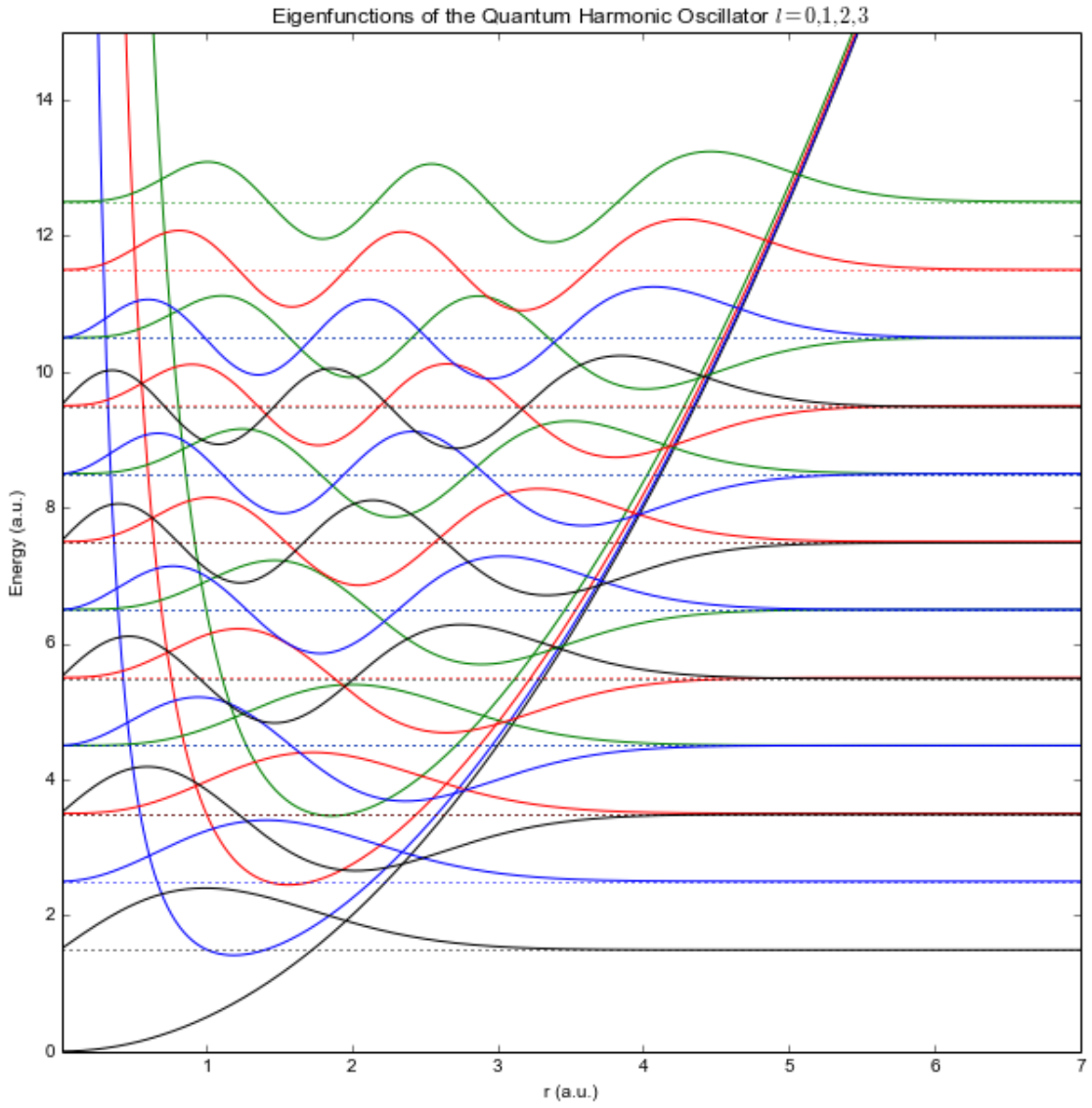

```
In [52]: print E3[0],E3[1],E3[2],E3[3],E3[4]
         A = E3-3/2.
         print A[0],A[1],A[2],A[3],A[4]
```

```
4.49999286871 6.49996656009 8.49991575099 10.4998404557 12.4997413142
2.99999286871 4.99996656009 6.99991575099 8.99984045567 10.9997413142
```

0.0.7 All together

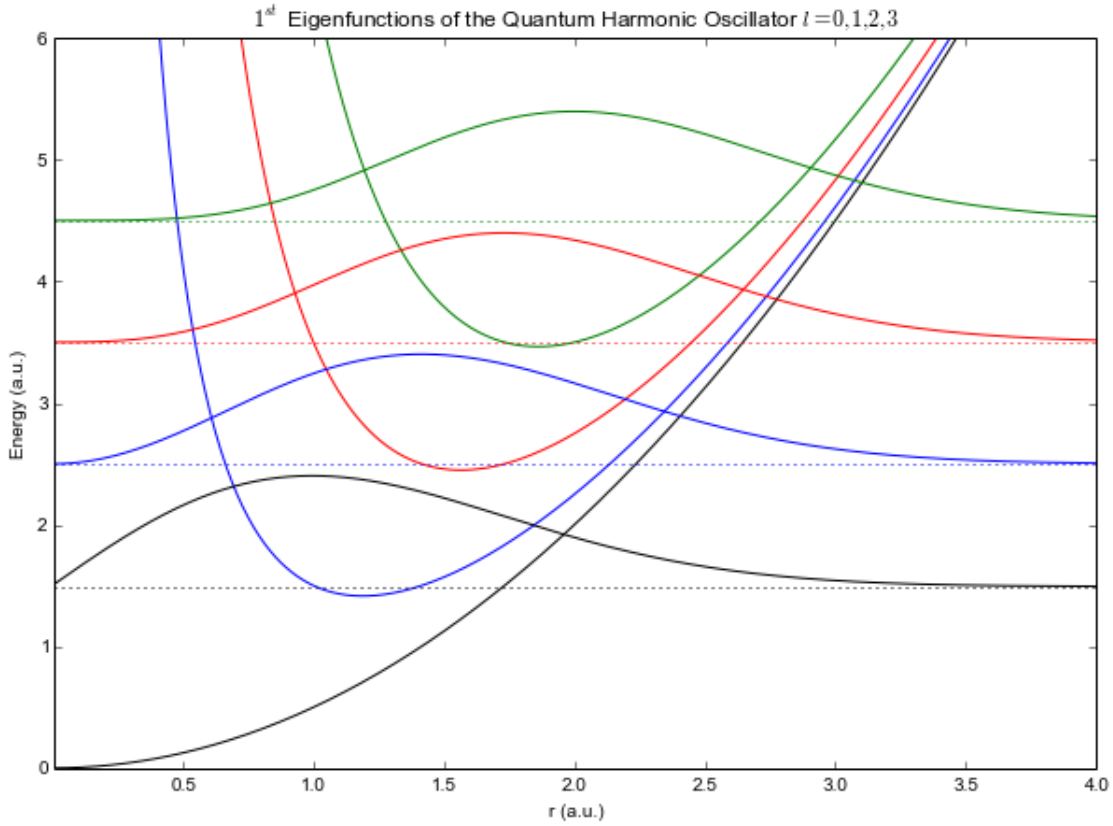
```
In [61]: #define plot size in inches (width, height) & resolution(DPI)
         fig = figure(figsize=(10, 10), dpi=100)

         # Plot the Harmonic potential
         plot(x,V,color='k')
         plot(x1,V1,color='b')
         plot(x1,V2,color='r')
         plot(x1,V3,color='g')

         for i in range(nfunctions):
```

```
        axhline(y=E[i],color='k',ls=":")
        plot(x,U[:,i]+E[i], color='k')
        axhline(y=E1[i],color='b',ls=":")
        plot(x1,U1[:,i]+E1[i], color='b')
        axhline(y=E2[i],color='r',ls=":")
        plot(x1,U2[:,i]+E2[i], color='r')
        axhline(y=E3[i],color='g',ls=":")
        plot(x1,U3[:,i]+E3[i], color='g')
    axis([xmin,xmax,0,15])
    title("Eigenfunctions of the Quantum Harmonic Oscillator $l=0,1,2,3$")
    xlabel("r (a.u.)")
    ylabel("Energy (a.u.)")
    show()
```



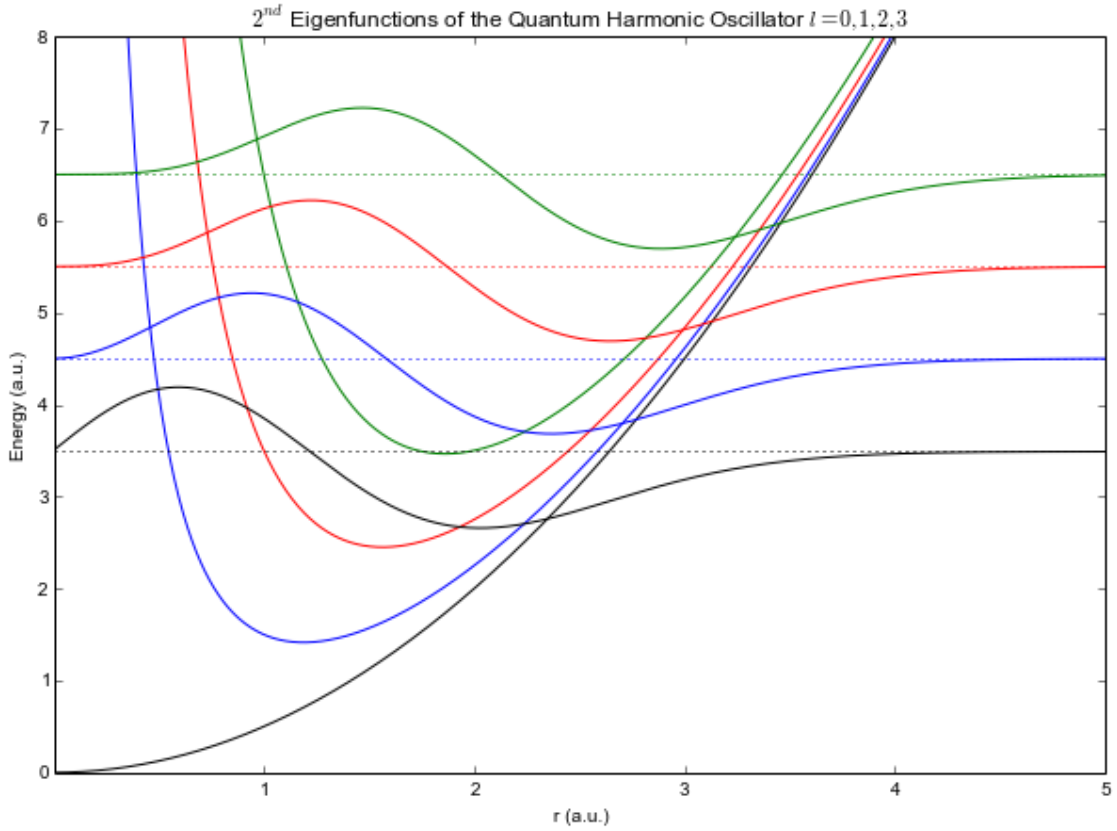
In [65]: *#define plot size in inches (width, height) & resolution(DPI)*

```
fig = figure(figsize=(10, 7), dpi=100)

# Plot the Harmonic potential
plot(x,V,color='k')
plot(x1,V1,color='b')
plot(x1,V2,color='r')
plot(x1,V3,color='g')

for i in range(0,1):
    axhline(y=E[i],color='k',ls=":")
    plot(x,U[:,i]+E[i], color='k')
    axhline(y=E1[i],color='b',ls=":")
    plot(x1,U1[:,i]+E1[i], color='b')
    axhline(y=E2[i],color='r',ls=":")
    plot(x1,U2[:,i]+E2[i], color='r')
    axhline(y=E3[i],color='g',ls=":")
    plot(x1,U3[:,i]+E3[i], color='g')
axis([xmin,4,0,6])
title("$1^{st}$ Eigenfunctions of the Quantum Harmonic Oscillator $l=0,1,2,3$")
xlabel("r (a.u.)")
ylabel("Energy (a.u.)")
show()
```



In [69]:
```
#define plot size in inches (width, height) & resolution(DPI)
fig = figure(figsize=(10, 7), dpi=100)
```

```
# Plot the Harmonic potential
plot(x,V,color='k')
plot(x1,V1,color='b')
plot(x1,V2,color='r')
plot(x1,V3,color='g')

for i in range(1,2):
    axhline(y=E[i],color='k',ls=":")
    plot(x,U[:,i]+E[i], color='k')
    axhline(y=E1[i],color='b',ls=":")
    plot(x1,U1[:,i]+E1[i], color='b')
    axhline(y=E2[i],color='r',ls=":")
    plot(x1,U2[:,i]+E2[i], color='r')
    axhline(y=E3[i],color='g',ls=":")
    plot(x1,U3[:,i]+E3[i], color='g')
axis([xmin,5,0,8])
title("$2^{nd}$ Eigenfunctions of the Quantum Harmonic Oscillator $l=0,1,2,3$")
xlabel("r (a.u.)")
ylabel("Energy (a.u.)")
show()
```



In [ ]:

In [ ]:

1 One-Dimensional Harmonic Oscillator using Special Functions

In [ ]:
```
from sympy import hermite
from math import gamma, exp, pi, sqrt

# Solución Analítica

def oscillator(n,x):

    arg = 2**n*gamma(n+1)*sqrt(pi)
    psi = 1.0/sqrt(arg) * exp(-x**2/2.0) * hermite(n, x)
    return psi
```

In [ ]:
```
from numpy import  linspace , zeros
from matplotlib.pyplot import plot, title, legend, show, axhline, \
                               xlabel, ylabel, axis, figure

nfunctions = 5

# array definitions
nsize = 100
xmin=-3
xmax=3
x = linspace(xmin,xmax,nsize)
psi = zeros(nsize)

# Plot wavefunctions

#define plot size in inches (width, height) & resolution(DPI)
fig = figure(figsize=(7, 6), dpi=100)

for n in range(nfunctions):
    # For each of the first few solutions, plot the energy level:
    axhline(y=E[n],color='k',ls=":")
    # as well as the eigenfunction, displaced by the energy level
    # so they don't all pile up on each other:
    plot(x,U[:,n]+E[n], ls="--")

    for i in range(0,nsize):
        psi[i] = (-1)**n*oscillator(n,x[i])+E[n]

    plot(x,psi)


xlabel('x (a.u.)')
ylabel(r'$\psi(x)$')
title("Comparison of numeric and analytic solutions to\
 the Harmonic Oscillator",size=14)
#legend()
axis([xmin,xmax,0,6])
show()
```

Chequeo de Normalización (analítico)

```
In [ ]: import numpy as np
        import matplotlib as mp
        import sympy as sy

        from sympy import *

        x  = symbols('x ')
        k, m, n = symbols('k m n', integer=True)
        g1 = symbols('g1', cls=Function)
```

```
In [ ]: def oscillator(n,x):
            psi = 1.0/sqrt((2**n*gamma(n+1)*sqrt(pi))) * exp(-x**2/2.0) * hermite(n, x)
            return psi
```

```
In [ ]: integrate(oscillator(0,x)**2,(x,-oo,oo))
```

```
In [ ]: integrate(oscillator(0,x)*oscillator(3,x),(x,-oo,oo))
```

```
In [ ]: oscillator(1,3.9).evalf()
```

```
In [ ]: g1 = lambda x: oscillator(1,x).evalf()
```

```
In [ ]: for x in range(-5,5):
            print x,g1(x)
```

```
In [ ]: mpmath.plot(g1,[-5,5])
```

```
In [ ]: import this
```