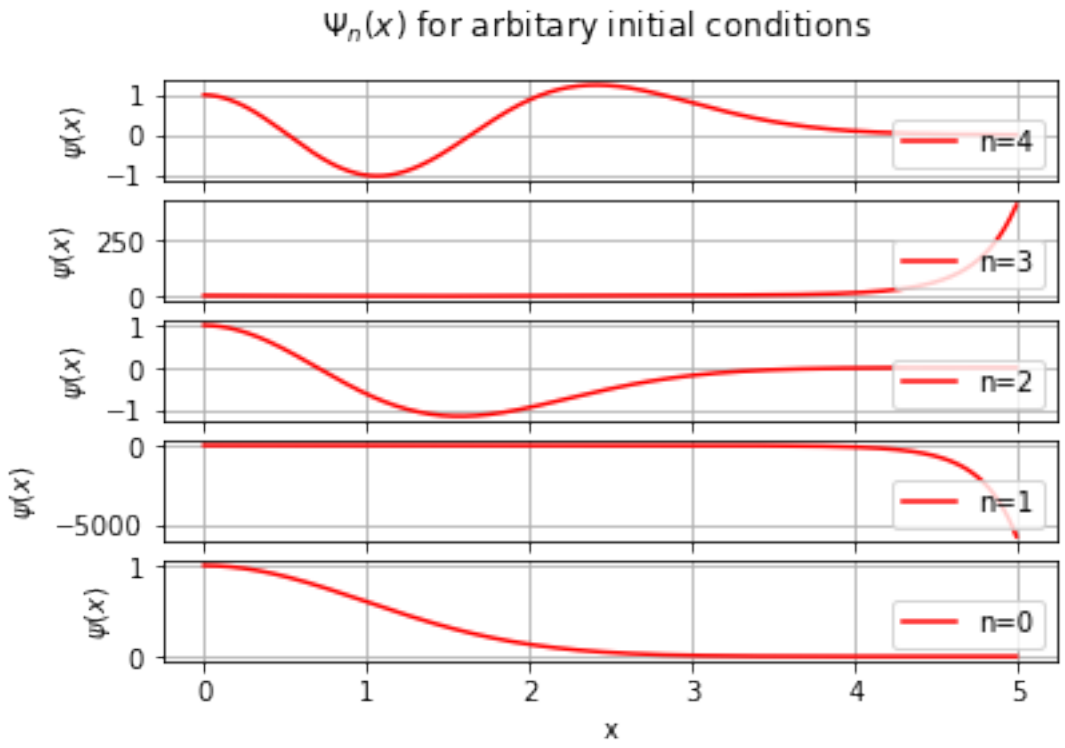
Listing 6.1: Harmonic oscilator Wave functions
Petridis Petros

June 19, 2022

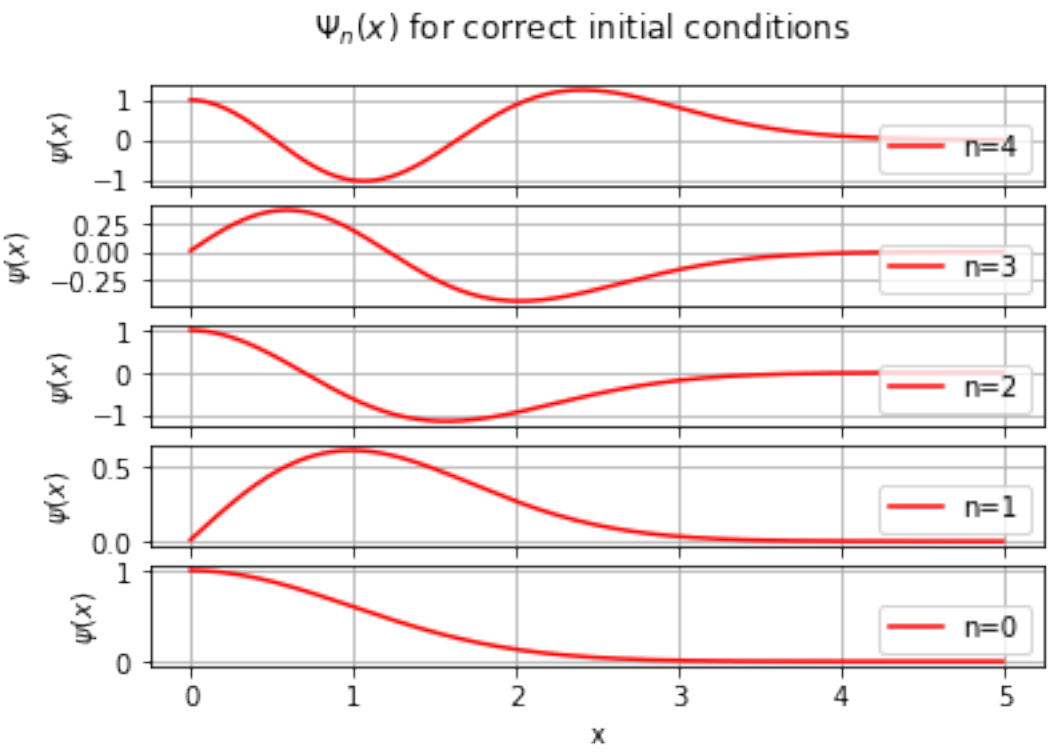
[1]:
```
import numpy as np, matplotlib.pylab as plt
from scipy import special as s
from scipy.integrate import simpson
import warnings
warnings.filterwarnings('ignore')


def rk4Algor(t,h,N,y,f):
            k1=np.zeros(N)
            k2=np.zeros(N)
            k3=np.zeros(N)
            k4=np.zeros(N)
            k1 = h*f(t,y)
            k2 = h*f(t+h/2.,y+k1/2.)
            k3 = h*f(t+h/2.,y+k2/2.)
            k4 = h*f(t+h,y+k3)
            y=y+(k1+2*(k2+k3)+k4)/6.
            return y
def f(x,y):
    fVec[0]=y[1] #z(x)=y'(x)
    fVec[1]=-(2*n+1-x**2)*y[0] #z'(x)=-(2n+1-x^2)*y(x)
    return fVec

fVec=np.zeros(2)
y=np.zeros((2))

ns=[0,1,2,3,4]
```

[2]:
```
#Question 1
#arbitary initial conditions
#integration 0->5
fig1,ax1=plt.subplots(len(ns))
j=len(ns)-1
rVec=np.zeros((int(5/0.01)),float)
psiVec=np.zeros((int(5/0.01)),float)
for n in ns:
```

```
    y[0]=1
    y[1]=10**(-8)

    i=0
    f(0.0,y)
    dr=0.01

    for r in np.arange(0,5,dr):
        rVec[i]=r
        y=rk4Algor(r,dr,2,y,f)

        psiVec[i]=y[0]
        i=i+1

    ax1[j].plot(rVec,psiVec, label="n=%d"%n,color="red")
    ax1[j].grid()

    ax1[j].legend(loc="lower right")
    j-=1
fig1.suptitle(r"$\Psi_n(x)$"" for arbitary initial conditions")
for axis in ax1.flat:
    axis.set(xlabel="x", ylabel="$\psi(x)$")
```

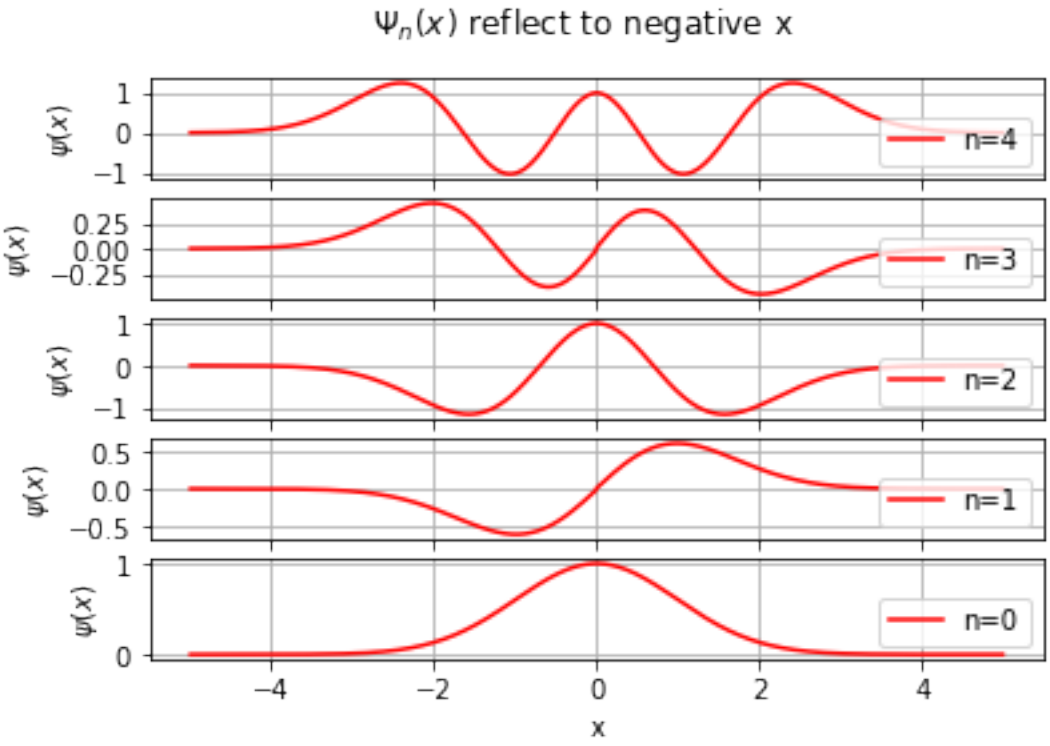


[3]:
```
#Question 2
#correct initial conditions
#integration 0->5
fig2,ax2=plt.subplots(len(ns))
rVec=np.zeros((int(5/0.01)),float)
psiVec=np.zeros((int(5/0.01)),float)
j=len(ns)-1
for n in ns:
    #parity
    if (n%2==0):
        y[0]=1
        y[1]=10**-8
    else:
        y[0]=10**-8
        y[1]=1

    i=0
    f(0.0,y)
    dr=0.01

    for r in np.arange(0,5,dr):
        rVec[i]=r
        y=rk4Algor(r,dr,2,y,f)

        psiVec[i]=y[0]
        i=i+1

    ax2[j].plot(rVec,psiVec, label="n=%d"%n,color="red")
    ax2[j].grid()

    ax2[j].legend(loc="lower right")
    j-=1
fig2.suptitle(r"$\Psi_n(x)$"" for correct initial conditions")
for axis in ax2.flat:
    axis.set(xlabel="x", ylabel="$\psi(x)$")
```



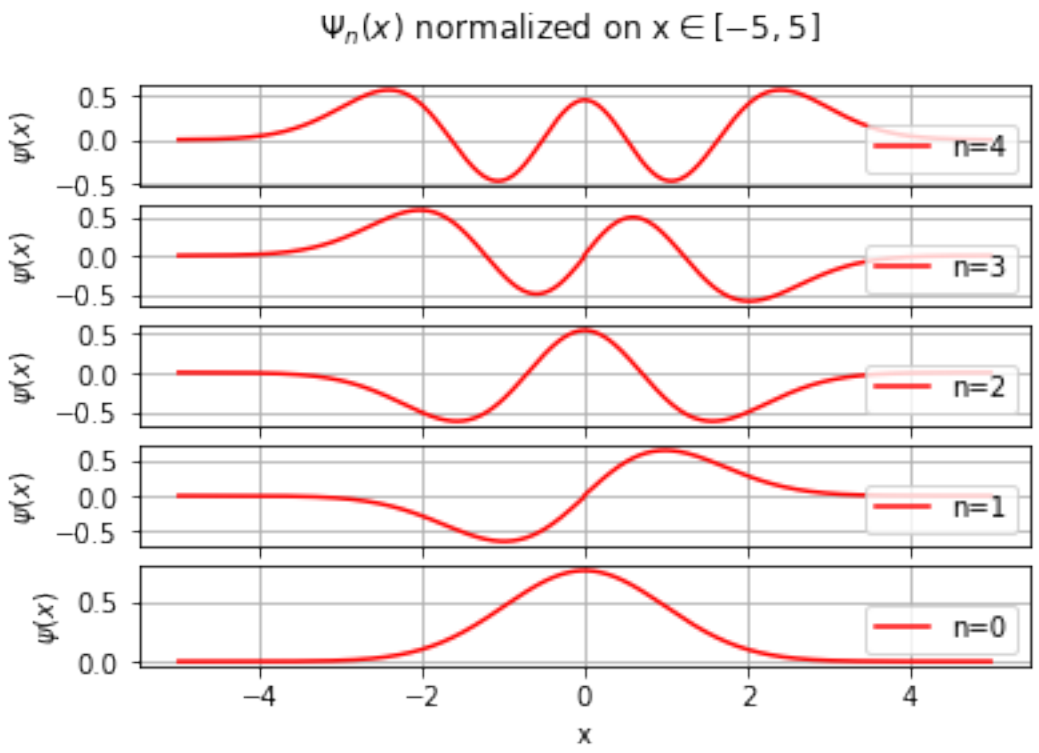
[4]:
```
#Question 3
#reflect the (x)   x,
fig3,ax3=plt.subplots(len(ns))
rVec=np.zeros((int(5/0.01)),float)
psiVec=np.zeros((int(5/0.01)),float)
j=len(ns)-1
for n in ns:
    #parity
    if (n%2==0):
        y[0]=1
        y[1]=10**-8
    else:
        y[0]=10**-8
        y[1]=1

    i=0
    f(0.0,y)
    dr=0.01

    for r in np.arange(0,5,dr):
        rVec[i]=r
        y=rk4Algor(r,dr,2,y,f)
```

```
        psiVec[i]=y[0]
        i=i+1

    if n%2==0:
        ar1=np.concatenate((-np.flip(rVec[1::]),rVec))
        ar2=np.concatenate((np.flip(psiVec[1::]),psiVec))
    else:
        ar1=np.concatenate((-np.flip(rVec[1::]),rVec))
        ar2=np.concatenate((-np.flip(psiVec[1::]),psiVec))

    ax3[j].plot(ar1,ar2,label="n=%d"%n,color="red")
    ax3[j].legend(loc="lower right")
    ax3[j].grid(True)
    j-=1
fig3.suptitle(r"$\Psi_n(x)$"" reflect to negative x")
for axis in ax3.flat:
    axis.set(xlabel="x", ylabel="$\psi(x)$")
```



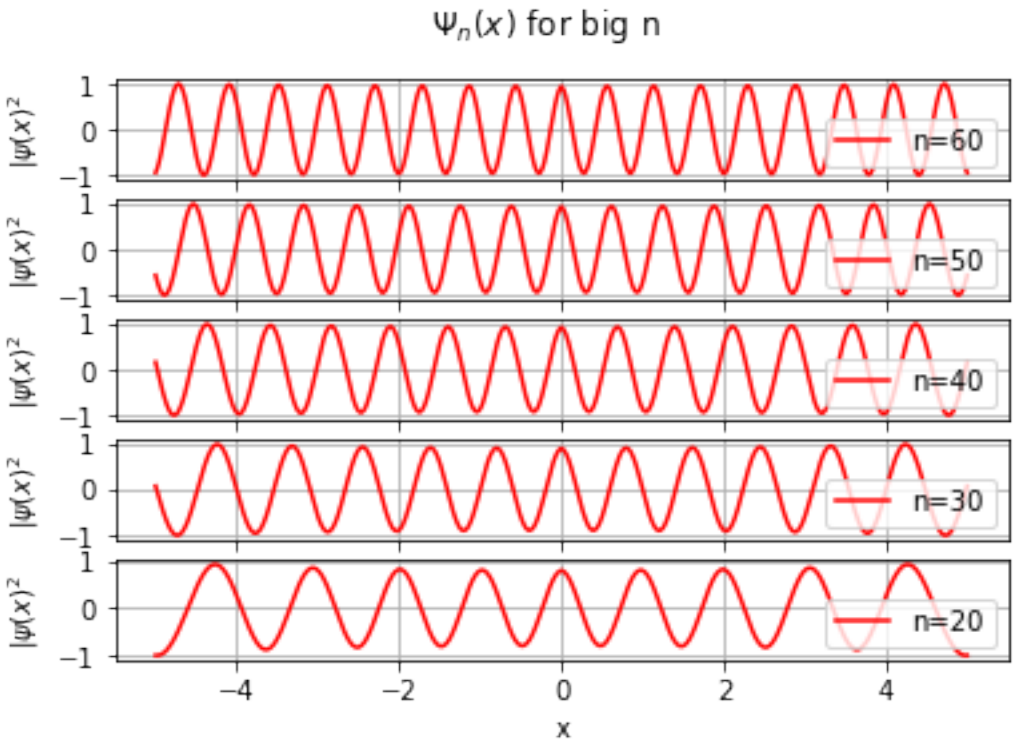
[5]:
```
#Question 4
#normalization
fig4,ax4=plt.subplots(len(ns))
```

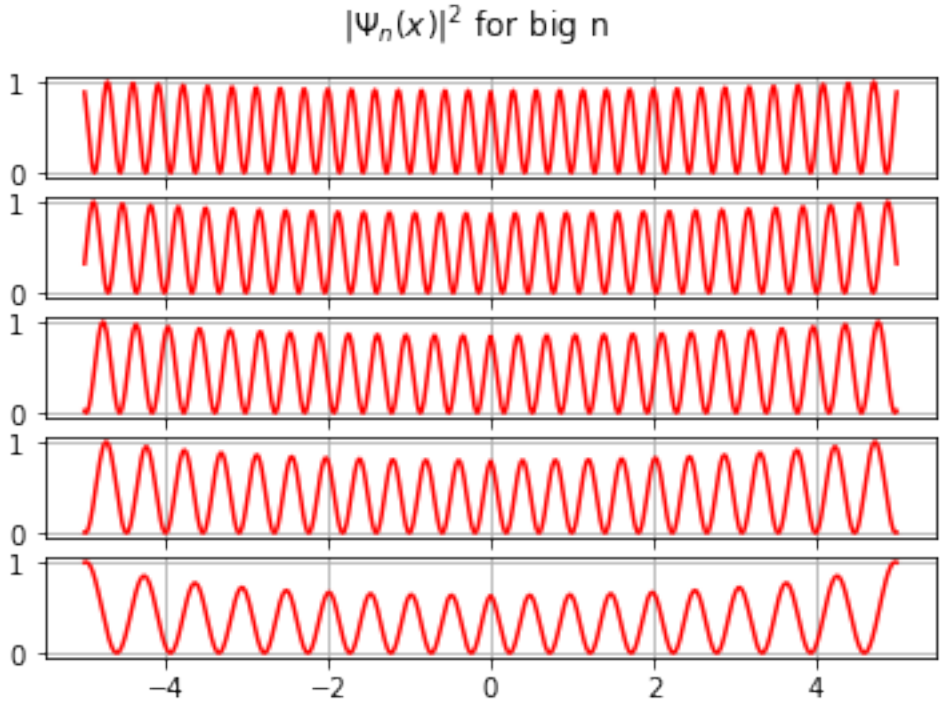
```
rVec=np.zeros((int(5/0.01)),float)
psiVec=np.zeros((int(5/0.01)),float)
j=len(ns)-1
for n in ns:
    #parity
    if (n%2==0):
        y[0]=1
        y[1]=10**-8
    else:
        y[0]=10**-8
        y[1]=1

    i=0
    f(0.0,y)
    dr=0.01

    for r in np.arange(0,5,dr):
        rVec[i]=r
        y=rk4Algor(r,dr,2,y,f)

        psiVec[i]=y[0]
        i=i+1

    if n%2==0:
        ar1=np.concatenate((-np.flip(rVec[1::]),rVec))
        ar2=np.concatenate((np.flip(psiVec[1::]),psiVec))
        I=simpson(abs(ar2)**2,ar1)
    else:
        ar1=np.concatenate((-np.flip(rVec[1::]),rVec))
        ar2=np.concatenate((-np.flip(psiVec[1::]),psiVec))
        I=simpson(abs(ar2)**2,ar1)

    ar2=ar2/np.sqrt(I)
    ax4[j].plot(ar1,ar2,label="n=%d"%n,color="red")
    ax4[j].grid()
    ax4[j].legend(loc="lower right")
    j-=1
fig4.suptitle(r"$\Psi_n(x)$"" normalized on x"r"$\in[-5,5]$" )
for axis in ax4.flat:
    axis.set(xlabel="x", ylabel="$\psi(x)$")
```



[6]:
```
#Question 5
#n large: n=20,30,40,50,60, n  |n|^2
fig5,ax5=plt.subplots(len(ns))
fig6,ax6=plt.subplots(len(ns))
rVec=np.zeros((int(5/0.01)),float)
psiVec=np.zeros((int(5/0.01)),float)
ns_2=[20,30,40,50,60]
j=len(ns_2)-1
for n in ns_2:
    #parity
    if (n%2==0):
        y[0]=1
        y[1]=10**-8
    else:
        y[0]=10**-8
        y[1]=1

    i=0
    f(0.0,y)
    dr=0.01

    for r in np.arange(0,5,dr):
```

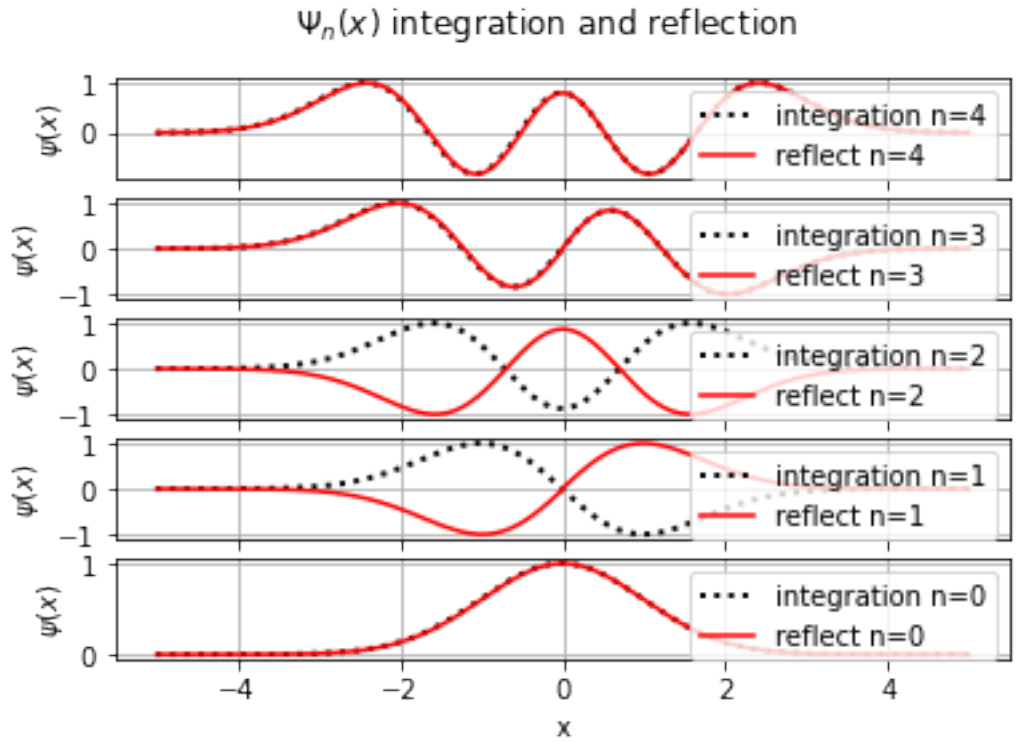
```
        rVec[i]=r
        y=rk4Algor(r,dr,2,y,f)

        psiVec[i]=y[0]
        i=i+1

    psiVec=psiVec/max(abs(psiVec))

    if n%2==0:
        ar1=np.concatenate((-np.flip(rVec[1::]),rVec))
        ar2=np.concatenate((np.flip(psiVec[1::]),psiVec))
    else:
        ar1=np.concatenate((-np.flip(rVec[1::]),rVec))
        ar2=np.concatenate((-np.flip(psiVec[1::]),psiVec))

    ax5[j].plot(ar1,ar2,label="n=%d"%n,color="red")
    ax5[j].grid()
    ax6[j].plot(ar1,abs(ar2)**2,label="n=%d"%n,color="red")
    ax6[j].grid()

    ax5[j].legend(loc="lower right")
    j-=1
fig5.suptitle(r"$\Psi_n(x)$"" for big n")
fig6.suptitle(r"$|\Psi_n(x)|^2$"" for big n")
for axis in ax5.flat:
    axis.set(xlabel="x", ylabel="$\psi(x)$")
for axis in ax5.flat:
    axis.set(xlabel="x", ylabel="$|\psi(x)^2$")
```


$\Psi_n(x)$ for big n
$|\psi(x)|^2$
n=60
n=50
n=40
n=30
n=20
1
0
−1
−4
−2
0
2
4
x

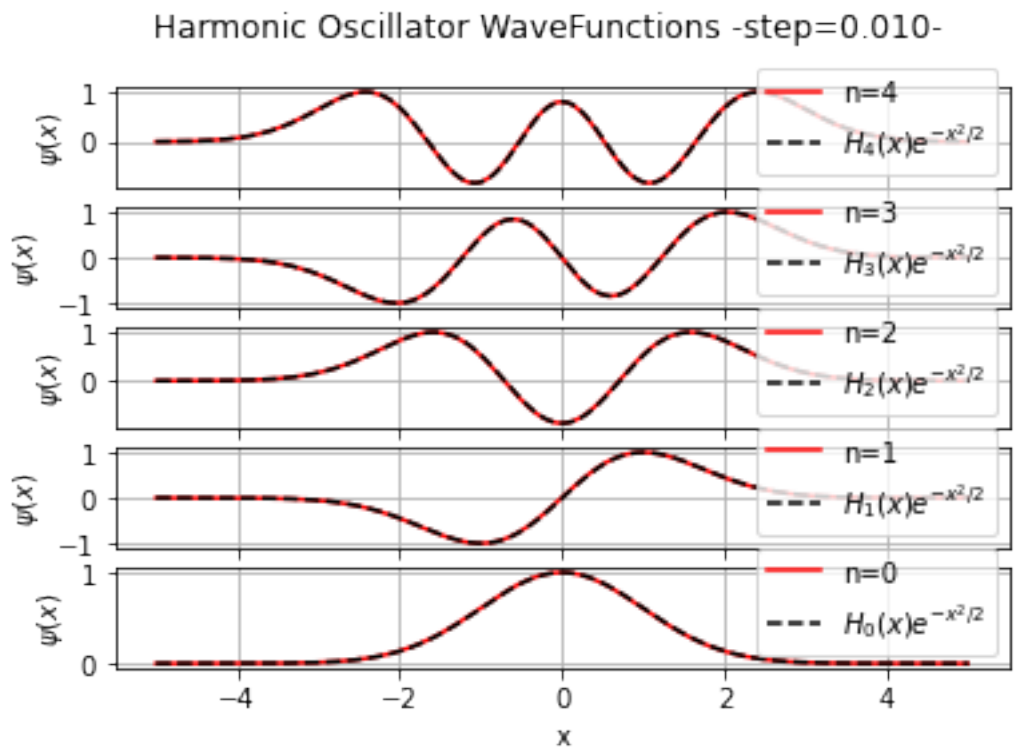

$|\Psi_n(x)|^2$ for big n
1
0
−4
−2
0
2
4

[7]:
```
#Question 6
#Integrate -5->5: same parity for reflection?
fig7,ax7=plt.subplots(len(ns))
rVec=np.zeros((int(10/0.01)),float)
psiVec=np.zeros((int(10/0.01)),float)
j=len(ns)-1
for n in ns:
    if (n%2==0):
        y[0]=10**-8
        y[1]=1
    else:
        y[0]=-10**-8
        y[1]=1

    i=0
    f(0.0,y)
    dr=0.01

    for r in np.arange(-5,5,dr):
        rVec[i]=r
        y=rk4Algor(r,dr,2,y,f)
        psiVec[i]=y[0]
        i=i+1

    ax7[j].plot(rVec,psiVec/max(abs(psiVec)),linewidth=2, label="integration␣
 ↪n=%d"%n,color="black",linestyle="dotted")
    j-=1

rVec=np.zeros((int(5/0.01)),float)
psiVec=np.zeros((int(5/0.01)),float)
j=len(ns)-1
for n in ns:
    #parity
    if (n%2==0):
        y[0]=1
        y[1]=10**-8
    else:
        y[0]=10**-8
        y[1]=1


    i=0
    f(0.0,y)
    dr=0.01

    for r in np.arange(0,5,dr):
        rVec[i]=r
```

```
        y=rk4Algor(r,dr,2,y,f)

        psiVec[i]=y[0]
        i=i+1

    if n%2==0:
        ar1=np.concatenate((-np.flip(rVec[1::]),rVec))
        ar2=np.concatenate((np.flip(psiVec[1::]),psiVec))
    else:
        ar1=np.concatenate((-np.flip(rVec[1::]),rVec))
        ar2=np.concatenate((-np.flip(psiVec[1::]),psiVec))

    ax7[j].plot(ar1,ar2/max(abs(ar2)),label="reflect n=%d"%n,color="red")
    ax7[j].grid()
    ax7[j].legend(loc="upper right")
    j-=1

fig7.suptitle("$\Psi_n(x)$"" integration and reflection")
for axis in ax7.flat:
    axis.set(xlabel="x", ylabel="$\psi(x)$")
```



[9]:
```
#Question 7:  n=Hn Hermite
fig8,ax8=plt.subplots(len(ns))
rVec=np.zeros((int(10/0.01)),float)
psiVec=np.zeros((int(10/0.01)),float)

j=len(ns)-1
for n in ns:
    if (n%2==0):
        y[0]=10**-8
        y[1]=10**-8
    else:
        y[0]=-10**-8
        y[1]=-10**-8

    i=0
    f(0.0,y)
    dr=0.01

    for r in np.arange(-5,5,dr):
        rVec[i]=r
        y=rk4Algor(r,dr,2,y,f)
        psiVec[i]=y[0]
        i=i+1

    psiVec=psiVec/max(abs(psiVec))

    ax8[j].plot(rVec,psiVec, label="n=%d"%n,color="red")
    ax8[j].grid()

    hermite=s.hermite(n,monic=False)
    rng=np.linspace(-5,5,1000)

    ax8[j].plot(rng,hermite(rng)*np.exp(-rng*rng/2)/max(abs(hermite(rng)*np.
 ↪exp(-rng*rng/2))),
                "k--", label="$H_%d(x)e^{-x^2/2}$"%n)
    ax8[j].legend(loc="lower right")
    j-=1
fig8.suptitle(f"Harmonic Oscillator WaveFunctions -step=%.3f-"%dr)
for axis in ax8.flat:
    axis.set(xlabel="x", ylabel="$\psi(x)$")
```


Harmonic Oscillator WaveFunctions -step=0.010-
n=4
$H_4(x)e^{-x^2/2}$
n=3
$H_3(x)e^{-x^2/2}$
n=2
$H_2(x)e^{-x^2/2}$
n=1
$H_1(x)e^{-x^2/2}$
n=0
$H_0(x)e^{-x^2/2}$
$\psi(x)$
x

[ ]: