ex04

May 18, 2023

1 Numerov algroithm for the Schrödinger equation

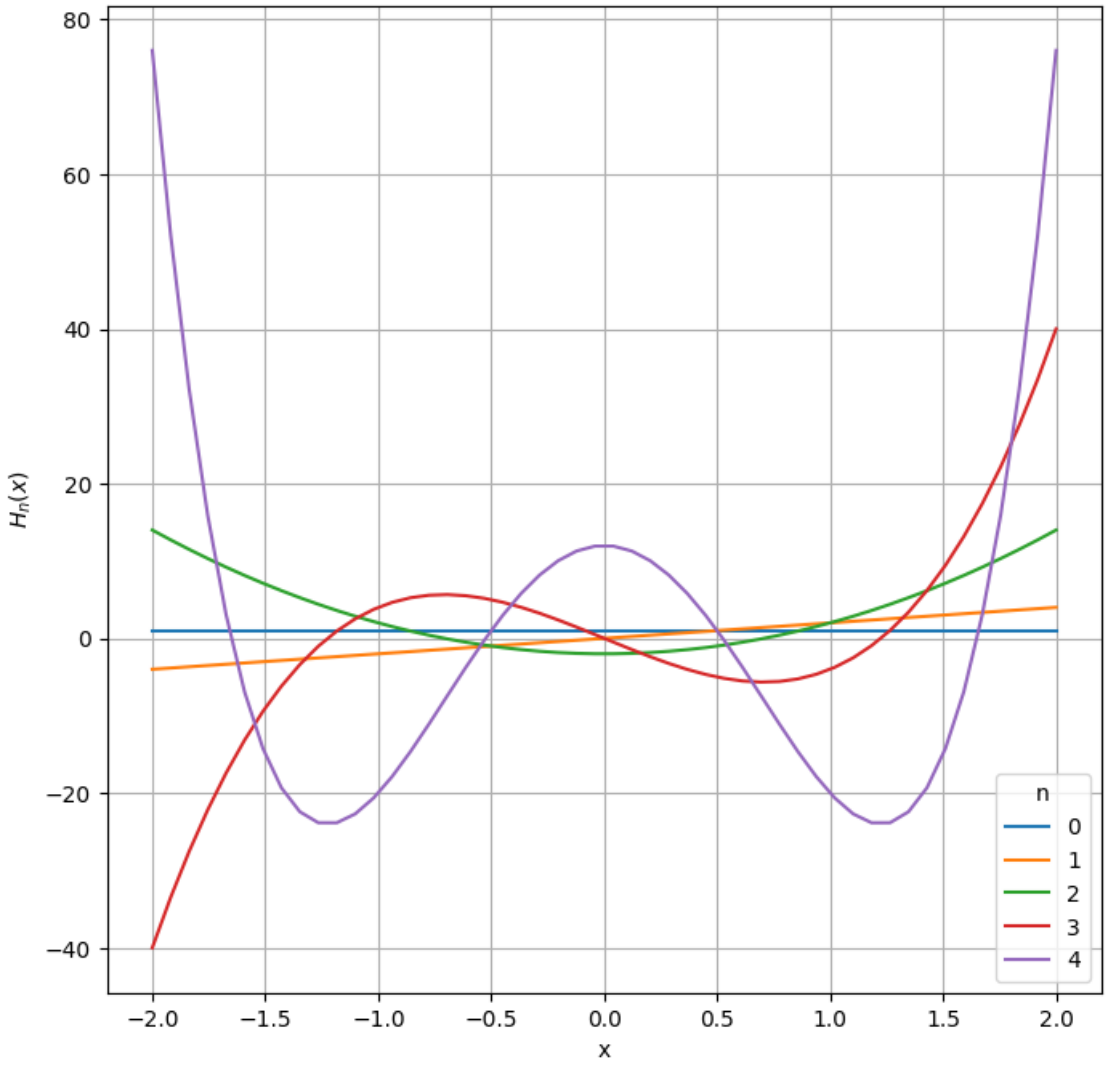
1.0.1 Q' inich Figueroa Coc: Gruppe 5

1.0.2 Paris J. Huth: Gruppe 1

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import argrelextrema
```

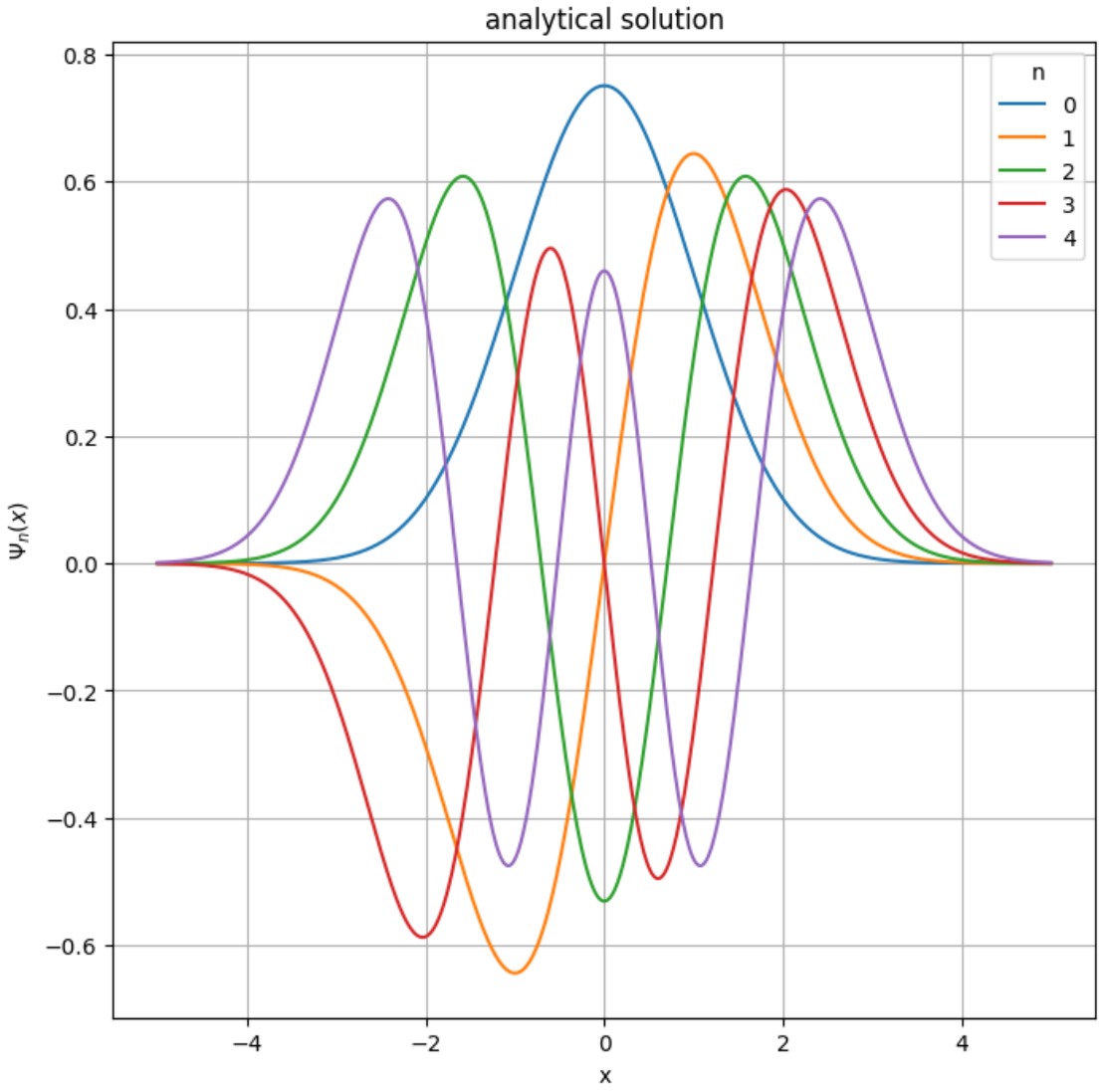
[ ]:
```
# define hermitian form of polynomial
def hermit_poly(n,x):
    if n==0:
        out = np.zeros(x.shape[0])
        out[:]=1
        return out
    elif n==1:
        return(2*x)
    else:
        return(2*x*hermit_poly(n-1,x)-2*(n-1)*hermit_poly(n-2,x))
```

[ ]:
```
# plot first five hermitian polynomials
test_x = np.linspace(-2,2)
f, ax = plt.subplots(1,1, figsize=(8,8))
for n in range(5):
    ax.plot(test_x, hermit_poly(n,test_x),label=n)
    ax.set(xlabel='x',ylabel='$H_n(x)$')
ax.legend(title='n')
ax.grid()
```



[ ]:
```
from scipy.special import factorial

# define function for analytical solution of psy
def Psy(n,x):
    norm = 2**n * factorial(n) * np.sqrt(np.pi)
    norm = np.sqrt(norm)
    return(hermit_poly(n,x)/norm * np.exp(-x**2/2))

# plot psy with the first five hermit polynomials
test_x = np.linspace(-5,5,1000)
f, ax = plt.subplots(1,1, figsize=(8,8))
for n in range(5):
    ax.plot(test_x, Psy(n, test_x), label=n)
```

```
ax.set(xlabel='x', ylabel='$\\Psi_n(x)$', title='analytical solution')
ax.legend(title='n')
ax.grid()
```

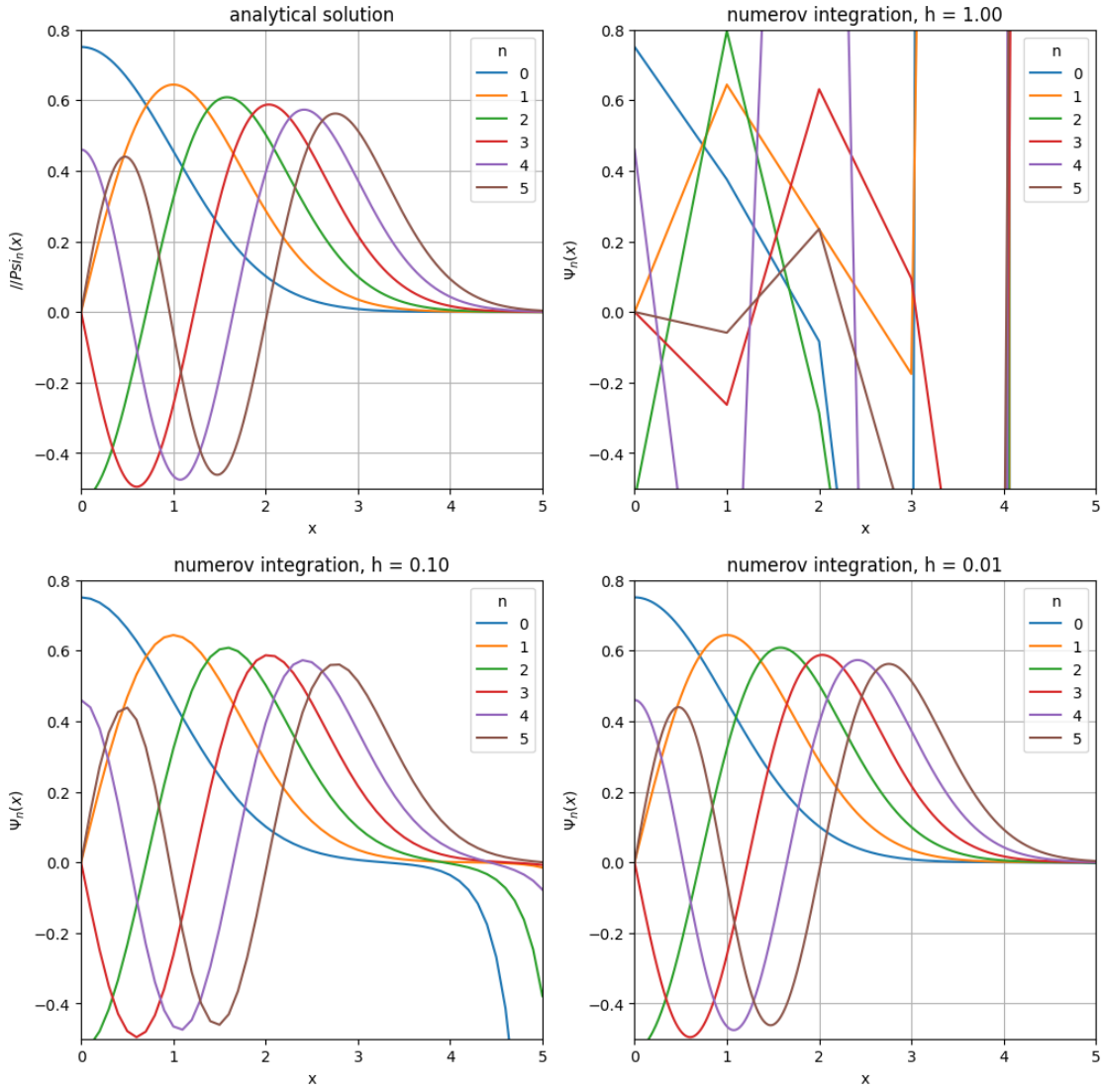


[ ]:
```
# k(x) in the muerov formalism for this problem
def k_harmOsci(x, epsilon):
    return(2*epsilon - x**2)

# numerov algorithm

def numerov(x0, y0, y1, k, n_step, h, k_args={}):
    output = np.zeros((n_step + 2, 2))
    output[0, 0] = x0
```

```
    output[0, 1] = y0
    output[1, 0] = x0 + h
    output[1, 1] = y1

    for i in np.arange(2, n_step + 2):
        output[i, 0] = output[i - 1, 0] + h
        y_i = (2. - 5. / 6. * k(output[i - 1, 0], **k_args) * h ** 2) *␣
 ↪output[i - 1, 1]
        y_i = y_i - (1. + 1. / 12. * k(output[i - 2, 0], **k_args) * h ** 2) *␣
 ↪output[i - 2, 1]
        y_i = y_i / (1. + 1. / 12. * k(output[i, 0], **k_args) * h ** 2)
        output[i, 1] = y_i

    return output



def numerov_harm0sci(n, x_max, h):
    # setup energy
    epsilon = n + 0.5   # energy eigenvalues

    # symmetric solutions if n is even
    if n % 2 == 0:
        a = Psy(n, np.array([0]))  #Psi(0)
        y0 = a
        y1 = y0 - h * h * k_harm0sci(0,epsilon) * y0*0.5

    # Asymmetric if n is odd
    else:
        a = Psy(n,np.array([h]))    # Psy(h)
        y0 = 0.
        y1= a

    # number of steps
    n_step = int(x_max/h) - 1

    return(numerov(x0=0., y0=y0, y1=y1, k=k_harm0sci, n_step=n_step, h = h,␣
 ↪k_args={'epsilon' : epsilon}))
```

[ ]:
```
# apply numerov integration to schroedinger eq of harmonic oscillator
h = [1., 0.1, 0.01]
n_i = range(6)
x_max = 5.

# setup plot
f, ax = plt.subplots(2,2, figsize=(12,12))
x_as = np.arange(0,x_max + h[2], h[2])
```

```
for n in n_i:
    ax[0,0].plot(x_as, Psy(n,x_as), label=n)
    ax[0,0].legend(title = 'n')
ax[0,0].grid()
ax[0,0].set(xlabel='x',ylabel='$//Psi_n(x)$', title = 'analytical solution',
xlim=(0,5), ylim=(-0.5, 0.8))
plot_coords = [[0,1],[1,0],[1,1]]

for i in range(len(h)):
    h_i = h[i]
    p=plot_coords[i]
    for n in n_i:
        num= numerov_harm0sci(n=n, x_max=x_max,h=h_i)
        ax[p[0],p[1]].plot(num[:,0],num[:,1], label=n)
    ax[p[0],p[1]].legend(title='n')
    ax[p[0],p[1]].set(xlabel='x', ylabel='$\\Psi_n(x)$',
        title='numerov integration, h = %.2f' % h_i, xlim=(0,5),ylim=(-0.5,0.8))

ax[p[0],p[1]].grid()
```



1.1 Neutrons in gravitational field

In order to solve the equation numerically, we first normalize the given equation:

$$\psi''(z) + \frac{2m}{\hbar}(E - mgz)\psi(z) = 0$$

by introducing:

$$x = \frac{z}{z_0}$$

in this way we get:

$$\frac{d^2\psi(x \cdot z_0)}{dx^2}\frac{dx^2}{dz^2} + \frac{2m}{\hbar}(E - mgz_0 \cdot x)\psi(x \cdot z_0) = 0$$

due to the linearity of the function ψ we take the factor z_0 out of the argument of the equation and get:

$$z_0 \frac{d^2\psi(x)}{dx^2}\frac{1}{z_0^2} + \frac{2m}{\hbar}(E - mgz_0 \cdot x)z_0 \cdot \psi(x) = 0$$

$$\iff \frac{d^2\psi(x)}{dx^2} + \frac{2m}{\hbar}(E - mgz_0 \cdot x)z_0^2 \cdot \psi(x) = 0$$

$$\iff \frac{d^2\psi(x)}{dx^2} + (\frac{2m}{\hbar}Ez_0^2 - \frac{2m^2g}{\hbar}z_0^3 \cdot x)\psi(x) = 0$$

from this variable transformation we get:

$$z_0 = \left(\frac{2m^2g}{\hbar}\right)^{1/3}$$

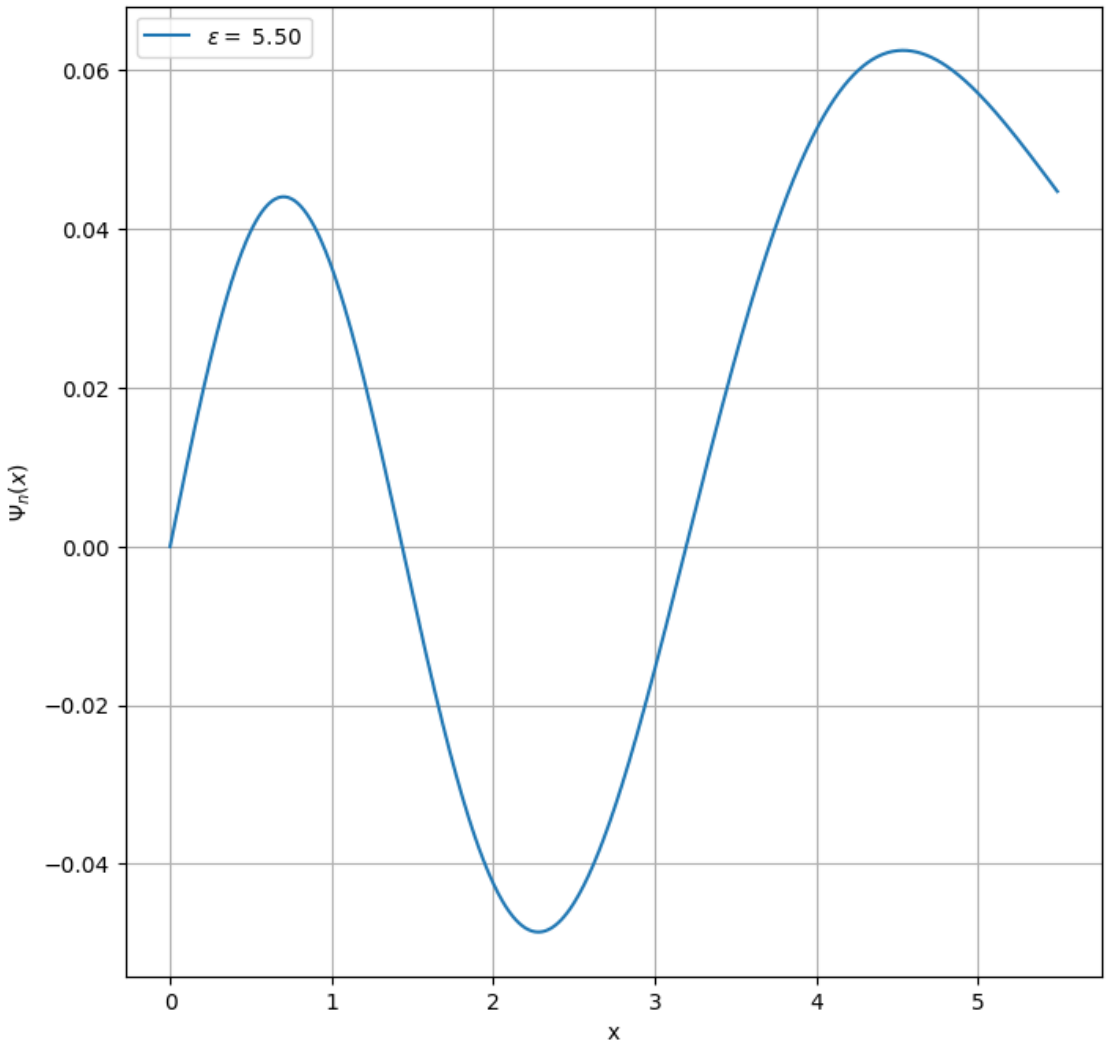
$$\varepsilon = \frac{2m}{\hbar}Ez_0^2.$$

The normalized function read as followed:

$$\psi''(x) + (\varepsilon - x)\psi(x) = 0$$

[ ]:
```
def neuNum(eps,x0, h, N,a):
    y=np.zeros(N)
    x=np.zeros(N)
    x[0]=x0
    for i in range(1,N):
        x[i]=x[i-1]+h

    y[0]=0
    y[1]=a

    for i in range(1,N-1):
        a=2*y[i]*(1-((5/12)*(h**2)*(eps-x[i])))
        b=y[i-1]*(1+((1/12)*(h**2)*(eps-x[i])))
        c=1+((1/12)*(h**2)*(eps-x[i]))
        y[i+1]=(a-b)/c

    return (x,y)

a=0.001
eps=5.5
h=0.01

N=int(np.around(1*eps/h))
result=neuNum(eps,0,h,N,a)
f, ax = plt.subplots(1,1, figsize=(8,8))
ax.plot(result[0],result[1],label='$\epsilon=$ %.2f' % eps)
ax.set(xlabel="x",ylabel="$\Psi_n (x)$")
ax.legend()
```

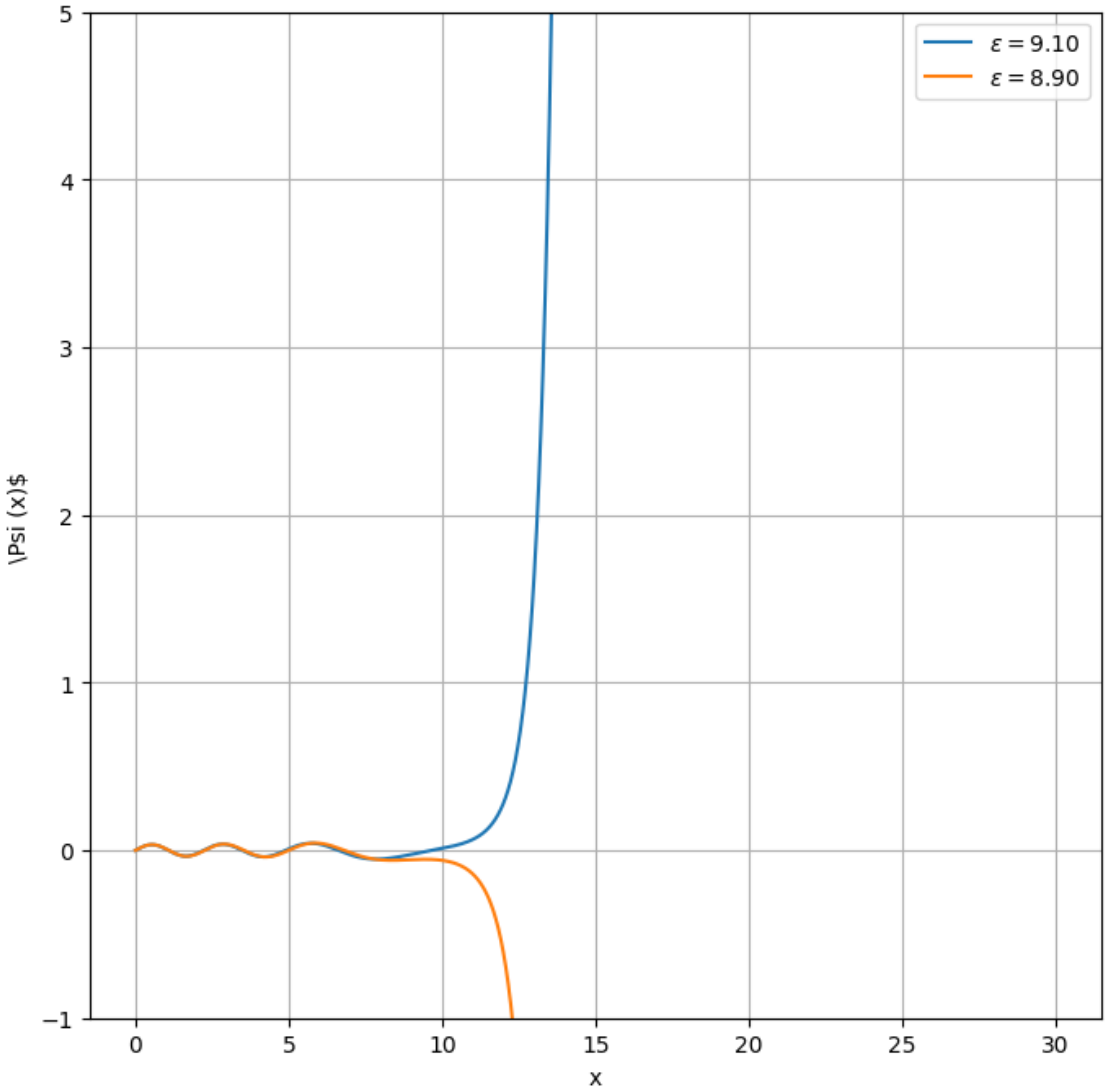
```
ax.grid()
```



[ ]:
```
#now we plot into the classical forbidden area
a=0.001
eps1=9.1
eps2=8.9
h=0.01
N=3000
result1=neuNum(eps1,0,h,N,a)
result2=neuNum(eps2,0,h,N,a)

f, ax = plt.subplots(1,1, figsize=(8,8))
ax.plot(result1[0],result1[1],label="$\epsilon=$%.2f" % eps1)
ax.plot(result2[0],result2[1],label="$\epsilon=$%.2f" % eps2)
```

```
ax.set(ylim=(-1,5),xlabel='x',ylabel='\Psi (x)$')
ax.grid()
ax.legend()
```

[ ]: <matplotlib.legend.Legend at 0x7f70e07ca710>

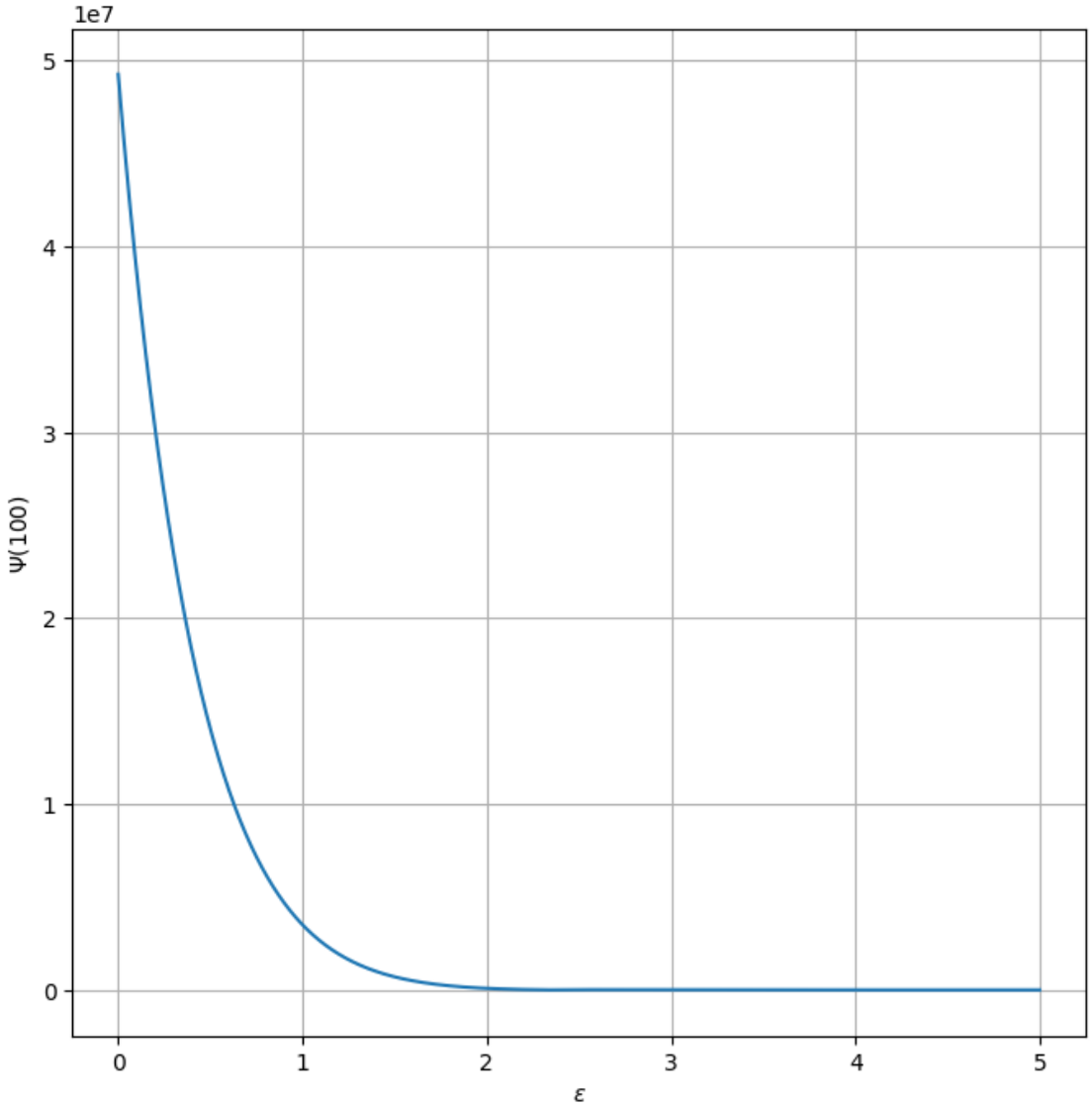


1.1.1 b)

[ ]:
```
a=0.001
h=0.01
N=1000
eps=np.linspace(0,5,1000)
psi=np.ones(eps.size)
```

```
for i in range(eps.size):
    result=neuNum(eps[i],0,h,N,a)
    psi[i]=np.abs(result[1][-1])

f, ax = plt.subplots(1,1, figsize=(8,8))

ax.plot(eps,psi)
ax.set(ylabel='$\Psi(100)$', xlabel='$\epsilon$')
ax.grid()
minima=argrelextrema(psi, np.less)
print(eps[minima])
```

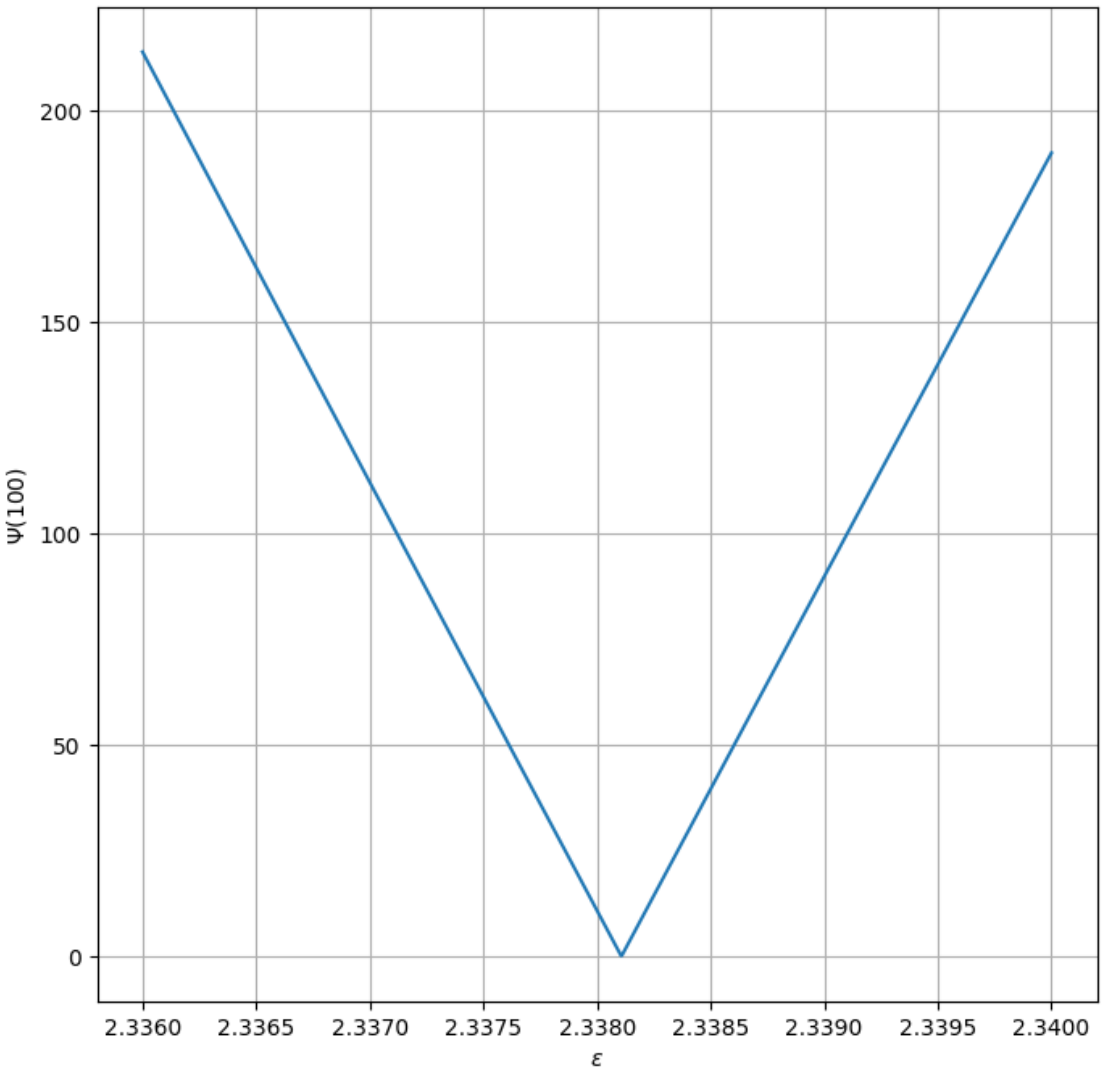
```
[2.33733734 4.08908909]
```



Since the eigenvalues correspond to a change of sign, we take the absolute value and find the minima in a given intervall.

[ ]:
```
eps=np.linspace(2.336,2.34,1000)
psi=np.ones(eps.size)
for i in range(eps.size):
    result=neuNum(eps[i],0,h,N,a)
    psi[i]=np.abs(result[1][-1])


f, ax = plt.subplots(1,1, figsize=(8,8))

ax.plot(eps,psi)
ax.set(ylabel='$\Psi(100)$', xlabel='$\epsilon$')
ax.grid()
minima=argrelextrema(psi, np.less)
print("First mimia is: "+str(eps[minima]))
```

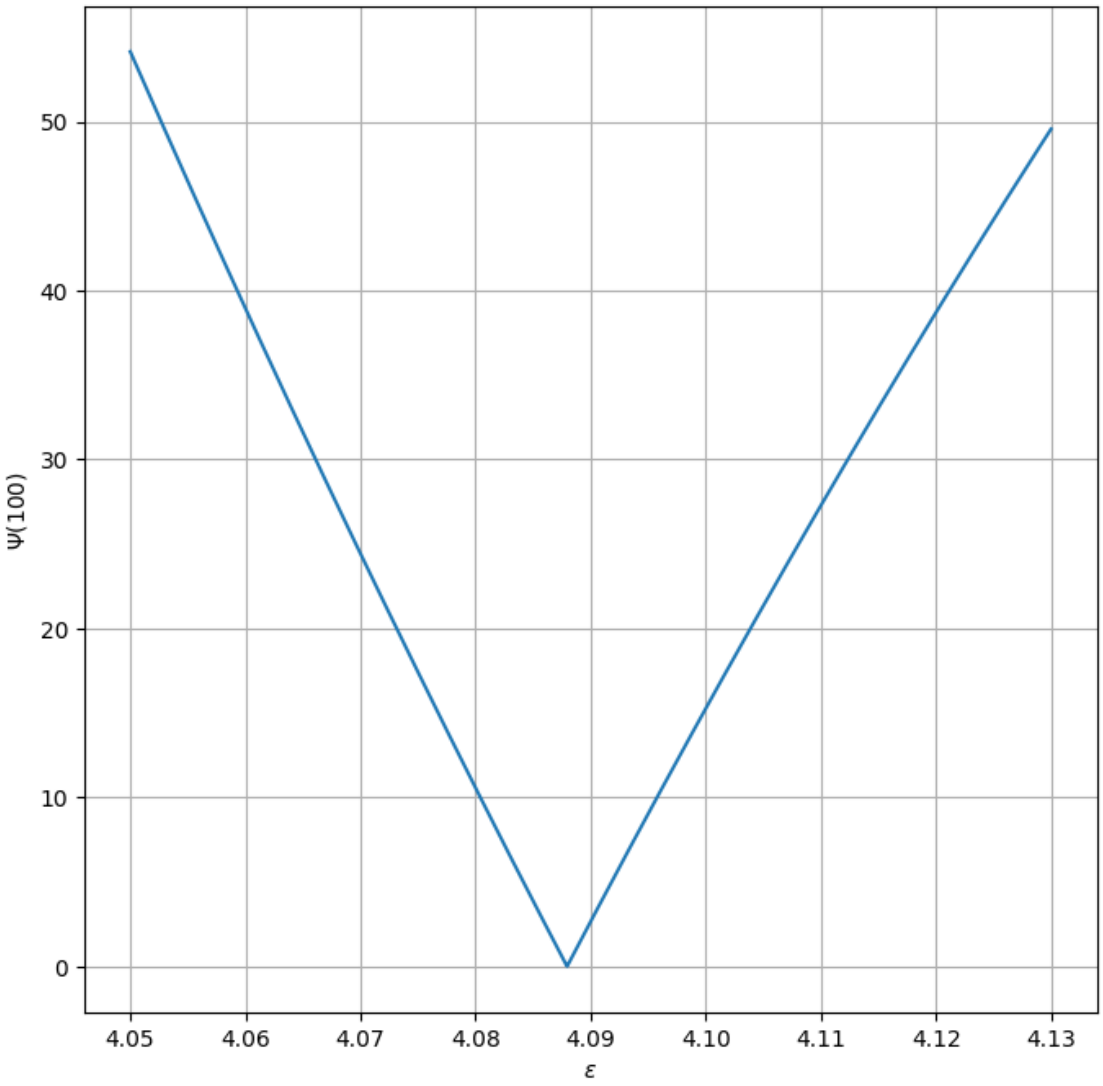
```
First mimia is: [2.33810611]
```



[ ]:
```
eps=np.linspace(4.05,4.13,1000)
psi=np.ones(eps.size)
for i in range(eps.size):
    result=neuNum(eps[i],0,h,N,a)
    psi[i]=np.abs(result[1][-1])

f, ax = plt.subplots(1,1, figsize=(8,8))

ax.plot(eps,psi)
ax.set(ylabel='$\Psi(100)$', xlabel='$\epsilon$')
ax.grid()

minima=argrelextrema(psi, np.less)
```

```
print("Second minima is: "+str(eps[minima]))
```

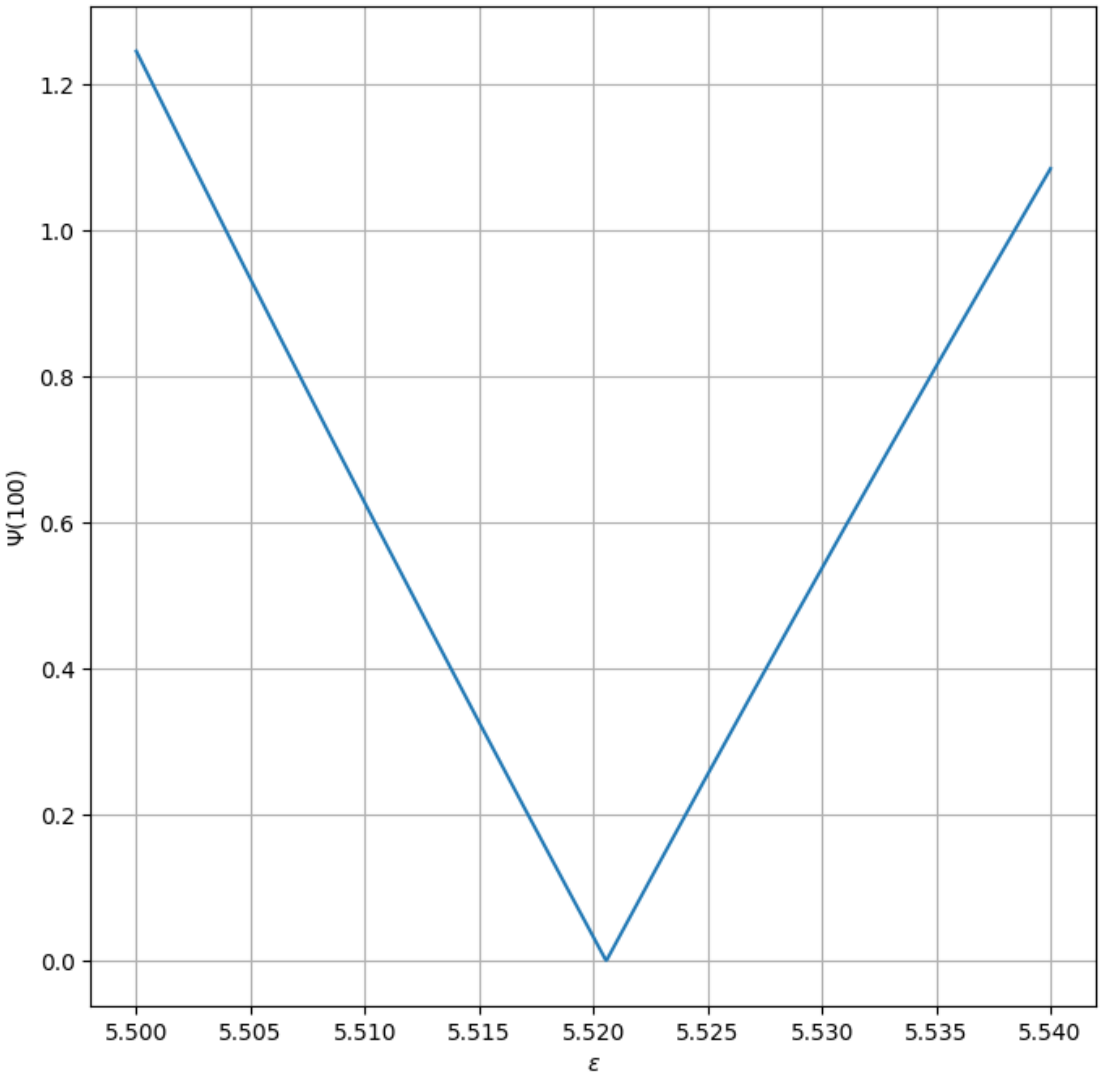
```
Second minima is: [4.08795796]
```



[ ]:
```
eps=np.linspace(5.5,5.54,1000)
psi=np.ones(eps.size)
for i in range(eps.size):
    result=neuNum(eps[i],0,h,N,a)
    psi[i]=np.abs(result[1][-1])


f, ax = plt.subplots(1,1, figsize=(8,8))
```

```
ax.plot(eps,psi)
ax.set(ylabel='$\Psi(100)$', xlabel='$\epsilon$')
ax.grid()

minima=argrelextrema(psi, np.less)
print("Third minima is: "+str(eps[minima]))
```

```
Third minima is: [5.52054054]
```



[ ]: