Assignment 6 - Harmonic Oscillator-II

SGTB Khalsa College, University of Delhi

Preetpal Singh(2020PHY1140)(20068567043)

Unique Paper Code: 32221501

Paper Title: Quantum Mechanics and Applications

Submitted on: August 24, 2022

B.Sc(H) Physics Sem V

Submitted to: Dr. Mamta

Programming

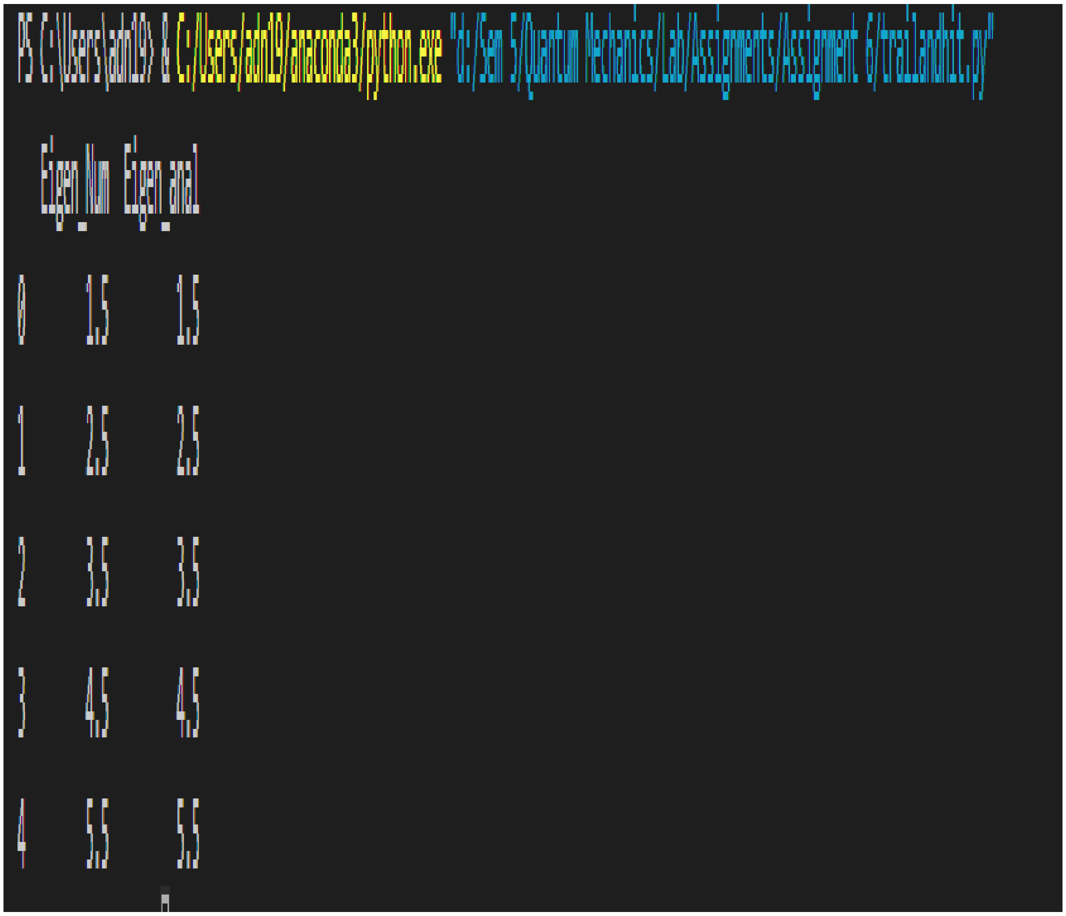
```python
import numpy as np
import matplotlib.pyplot as plt
import math
import scipy.integrate as integrate
from scipy import optimize,stats
import pandas as pd
def numerov(x_min, x_max,N,E_min, E_max  ):
    c_i =[];u=[]
    x = np.linspace(x_min,x_max,N+1)
    h = x[1]-x[0]
    Alpha = 2*(((-x**2)/2)+E_max)
    ddx_12 = (h**2)/12
    for i in range(0,N+1):
        c_i_ = 1 + np.multiply(ddx_12,Alpha[i])
        c_i.append(c_i_)
    if E_max%2==0:
        u_0 = 1
        u_1=(6-5*c_i[0])/c_i[1]
    else:
        u_0 = 0
        u_1=h
    u.append(u_0);u.append(u_1)
    for i in range(2,N+1):
        u_ = (1/c_i[i])*(((12-10*c_i[i-1])*u[i-1])-c_i[i-2]*u[i-2])
        u.append(u_)
    u_norm=u/np.sqrt(integrate.simps(np.power(u,2),x))
    return u_norm,  x

def sub_plot(ax,a,b,d,title,x_label,y_label):
    ax.scatter(a,b,label="Numerical Value",marker="*",color="red")
    ax.plot(a,d,label="Inbuilt Solution")
    ax.set_title(title)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    ax.legend()
    ax.grid(True)

def e_range(u,n_node,E_min,E_max) :
    I = []
    E = (E_min+E_max)/2
    for i in range(len(u)):
        if (u[i-1]*u[i]) < 0:
           I.append(i)
    N_node = len(I)
    if N_node > int(n_node):
       E_max = E
    else:
       E_min = E
    return len(I),E_min,E_max

def E(xi,xf,N,n_node,E_min,E_max,tol):
    for i in range(1000):
        U ,alpha = numerov(xi,xf,N,E_min,(E_min+E_max)/2)
        I ,E_min_new,E_max_new = e_range(U,n_node,E_min,E_max)
        if abs(E_max_new - E_min_new)<tol:
            break
        #    print("Eigen value :",E_max_new, "For n:",n_node)
        #    return E_min_new,E_max_new,U,alpha
        else:
           E_min = E_min_new
           E_max = E_max_new
    return E_min_new,E_max_new,U,alpha

def combine(list1, list2):
    list1_ = np.delete(list1, len(list1)-1)
```

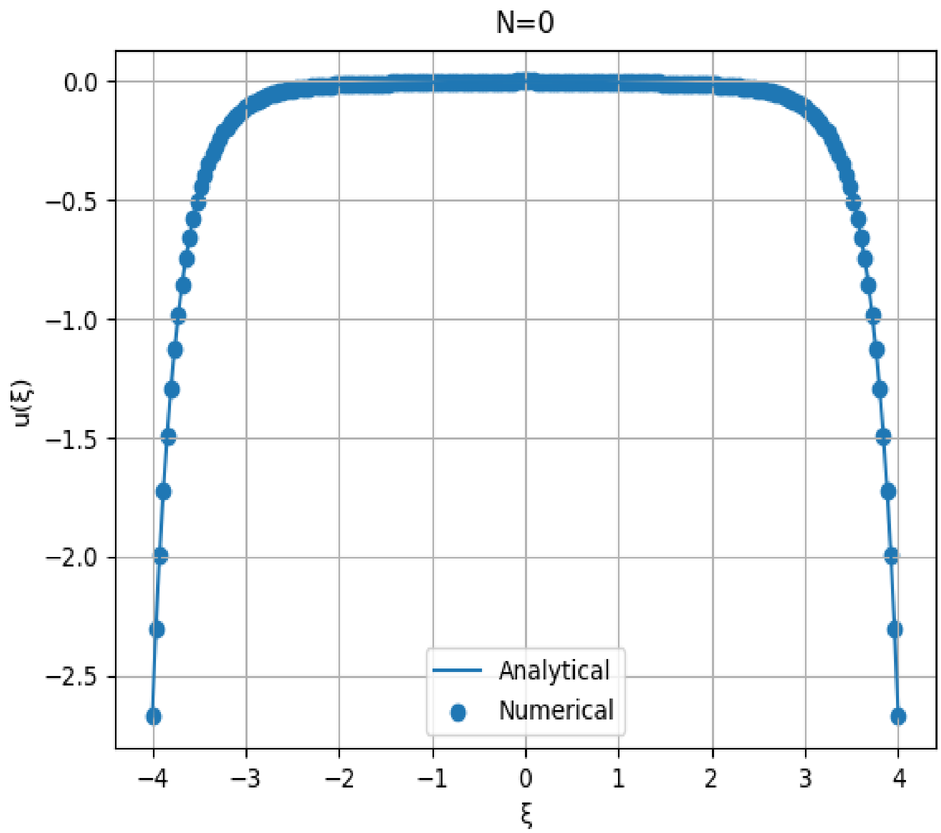
```
    result_list = []
    result_list = list(list1_)
    for item in list2:
        result_list.append(item)
    return result_list


def parity(xi, xf, N,n_node,E_min,E_max,tol):
    p = E(xi,xf,N,n_node,E_min,E_max,tol)
    t = p[2][::-1]
    # t_1 = p[2][::-1]
    array = [];array_1=[]

    if n_node%2 != 0:
        # array_1 = combine((np.multiply(-1,t_1)),p[2])
        array = combine((np.multiply(-1,t)),p[2])
        array_1 = combine((np.multiply(-1,t)),p[2])
    elif n_node%2 == 0:
        # array_1 = combine(t_1, p[2])
        array = combine(t, p[2])
    return array, array
# '''-----------------------------u vs x Plotting
    --------------------------------------'''
# for n_node in range(0,5):
#     xi=0;xf=4;N=100;E_max=n_node+0.5;E_min=0;tol=1e-10
#     if n_node %2 ==0:
#         u_0 = 1
#         u_prime=0
#     else:
#         u_0 = 0
#         u_prime = 1
#     par_1, par_2 = parity(xi,xf,N,n_node,E_min,E_max,tol)
#     if n_node==0 or n_node==3 or n_node==4:
#         plt.scatter(np.linspace(-xf,xf,2*N +1),np.power(np.multiply(-1,par_1),2),
    label="Numerical")
#         plt.plot(np.linspace(-xf,xf,2*N +1),np.power(np.multiply(-1,par_2),2),
    label="Analytical")
#         plt.grid()
#         plt.title(f'N={n_node}')
#         plt.xlabel("\u03BE")
#         plt.ylabel("$u(\u03BE)^2$")
#         plt.legend()
#         plt.show()

#     else:
#         plt.scatter(np.linspace(-xf,xf,2*N +1),np.power(par_1,2),label="Numerical
    ")
#         plt.plot(np.linspace(-xf,xf,2*N +1),np.power(par_2,2),label="Analytical")
#         plt.legend()
#         plt.grid()
#         plt.title(f'N={n_node}')
#         plt.xlabel("\u03BE")
#         plt.ylabel("$u(\u03BE)^2$")
#         plt.show()




# for n_node in range(0,5):
#     xi=0;xf=4;N=100;E_max=n_node+0.5;E_min=0;tol=1e-10
#     if n_node %2 ==0:
#         u_0 = 1
#         u_prime=0
#     else:
#         u_0 = 0
#         u_prime = 1
#     par_1, par_2 = parity(xi,xf,N,n_node,E_min,E_max,tol)
```

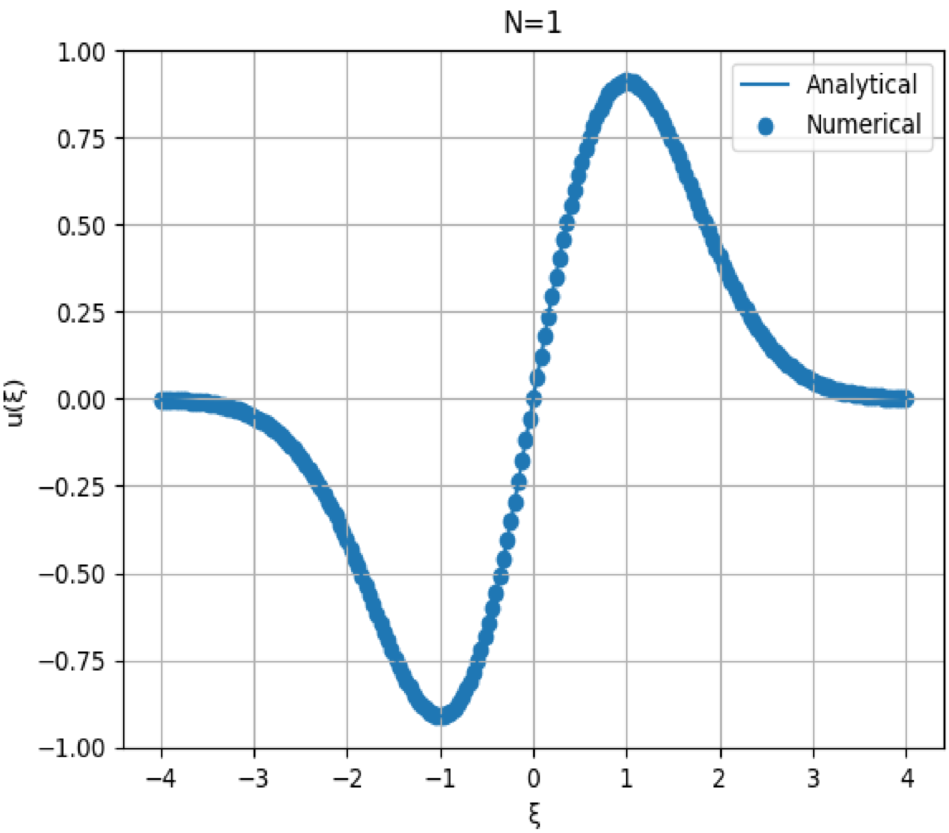
```
#     if n_node==0 or n_node==3 or n_node==4:
#         plt.scatter(np.linspace(-xf,xf,2*N +1),np.multiply(-1,par_1),label="Numerical")
#         plt.plot(np.linspace(-xf,xf,2*N +1),np.multiply(-1,par_2),label="Analytical")
#         plt.grid()
#         plt.title(f'N={n_node}')
#         plt.xlabel("\u03BE")
#         plt.ylabel("u(\u03BE)")
#         plt.legend()
#         plt.show()

#     else:
#         plt.scatter(np.linspace(-xf,xf,2*N +1),par_1,label="Numerical")
#         plt.plot(np.linspace(-xf,xf,2*N +1),par_2,label="Analytical")
#         plt.legend()
#         plt.grid()
#         plt.title(f'N={n_node}')
#         plt.xlabel("\u03BE")
#         plt.ylabel("u(\u03BE)")
#         plt.show()


'''------------------------------------Q_a(ii)---------------------'''
E_num=[];E_anal=[];n=[]
for i in range(1,6):
    xi=0;xf=10;N=100;n_node=i;E_max=i+0.5;E_min=0;tol=1e-10
    E_min_,E_max_,U,x= E(xi,xf,N,n_node,E_min,E_max,tol)
    E_num.append((E_min_ + E_max_)/2)
    E_anal.append(i+1/2)
    n.append(i)
#     plt.plot(x,U)
#     plt.grid()
#     plt.show()

# data = {
#     "Eigen_Num":E_num,
#     "Eigen_anal":E_anal,
# }
# print(pd.DataFrame(data))

'''-----------------------------------Q_a(iii)----------------------'''
# slope, intercept, r, p, se = stats.linregress(n, E_num )
# print("slope",slope)
# plt.scatter(n,E_num,label="Numerical")
# plt.plot(n,np.multiply(slope,n)+intercept,label="Analytical")
# plt.grid()
# plt.legend()
# plt.xlabel("n")
# plt.ylabel("E_n")
# plt.show()
# slope, intercept, r, p, se = stats.linregress(n, E_num )
# print("slope",slope)
'''-----------------------------------Q-b----------------------------'''
slope, intercept, r, p, se = stats.linregress(np.power(n,2), E_num )
print("slope",slope)
plt.scatter(np.power(n,2),E_num,label="Numerical")
plt.plot(np.power(n,2),np.multiply(slope,np.power(n,2))+intercept,label="Analytical")
plt.grid()
plt.legend()
plt.xlabel("n")
plt.ylabel("E_n")
plt.show()
slope, intercept, r, p, se = stats.linregress(np.power(n,2), E_num )
print("slope",slope)
'''--------------------------------Q_c------------------------------'''
```

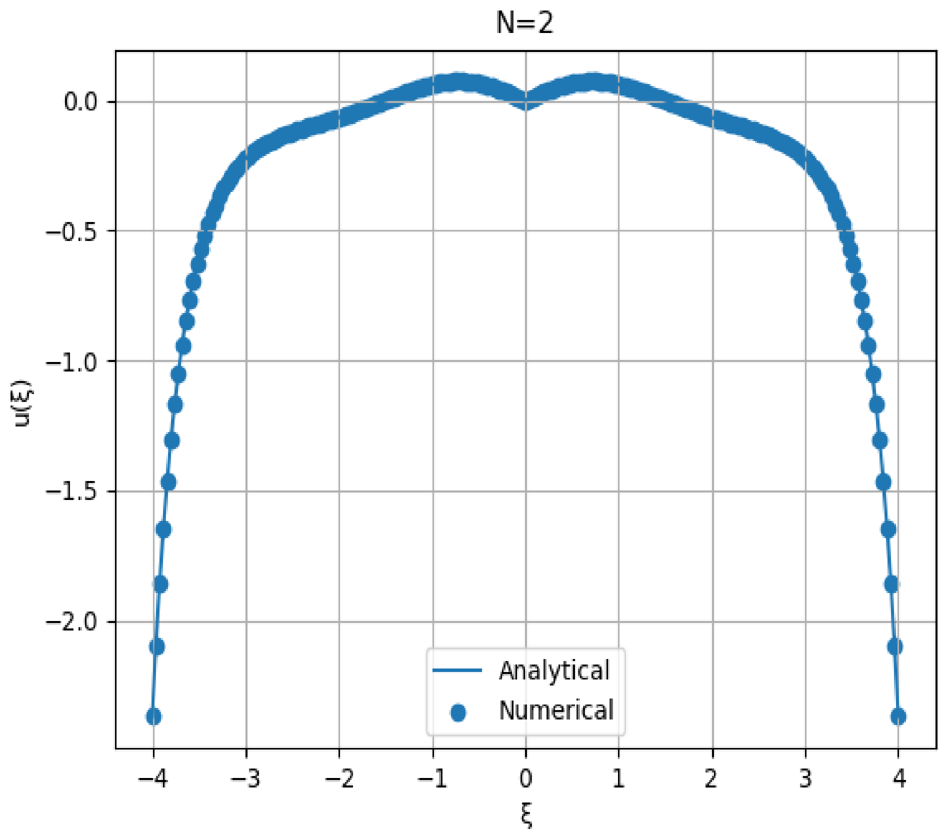
```
# xi=0;xf=3;N=100;n_node=3;E_max=2;E_min=0;tol=1e-10
# E_min_,E_max_,U,x= E(xi,xf,N,n_node,E_min,E_max,tol)
# print( (E_min_ + E_max_)/2)
# plt.plot(x,U)
# plt.grid()
# plt.show()
```

Result and Discussion

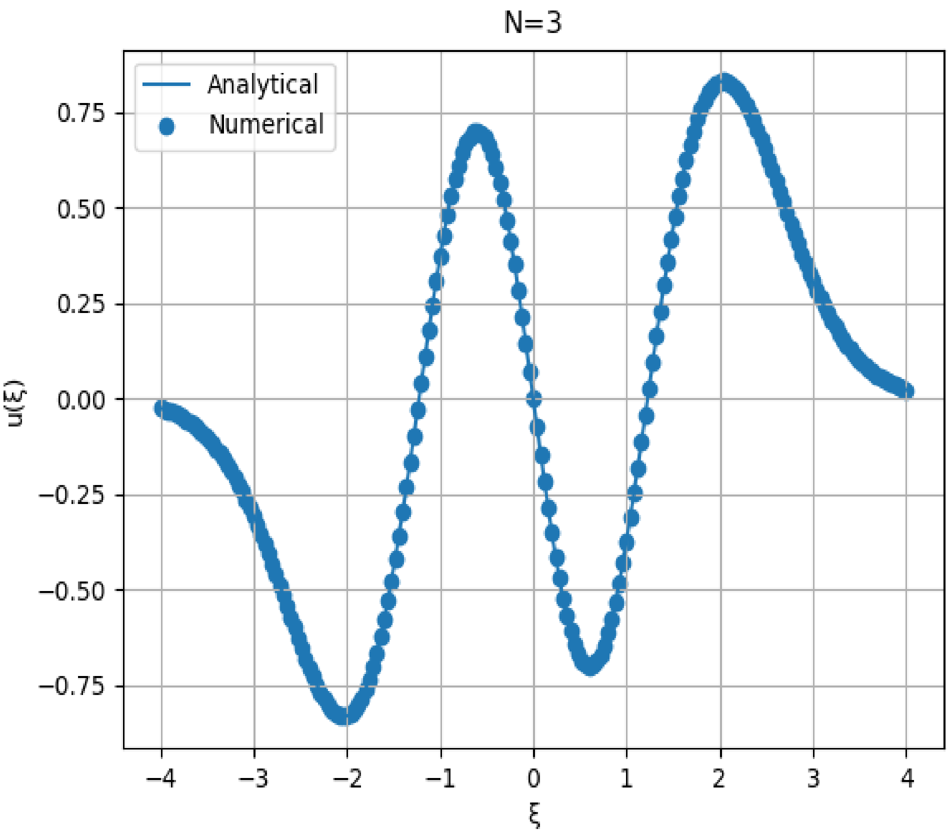


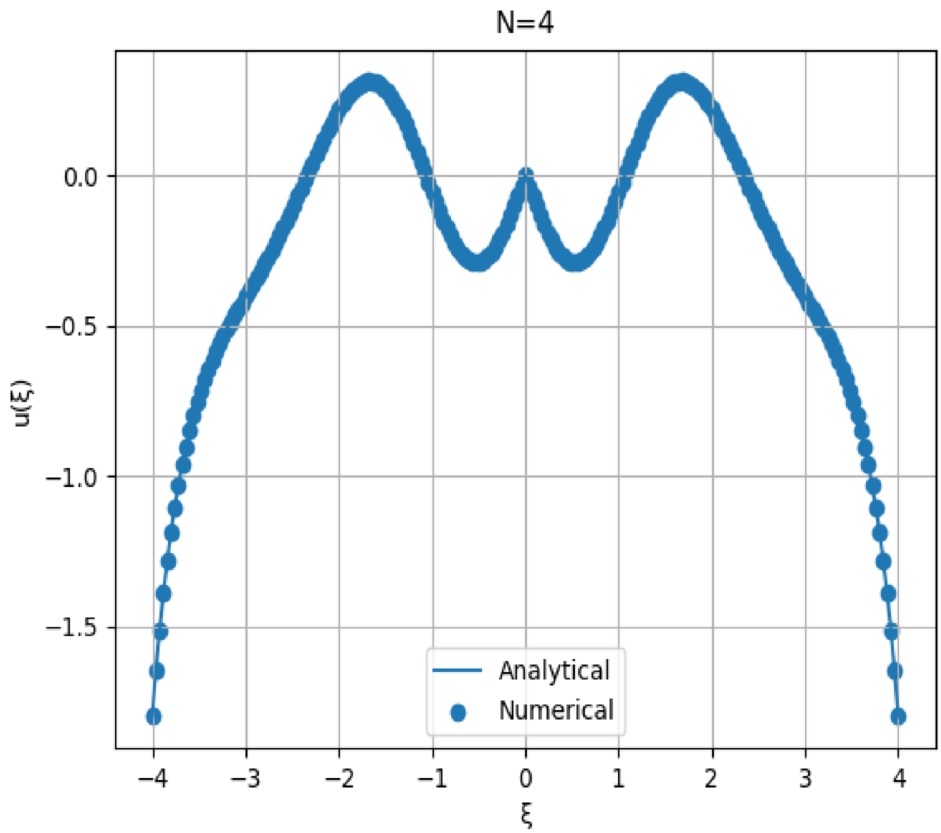

N=0
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
u(ξ)
−4
−3
−2
−1
0
1
2
3
4
ξ
Analytical
Numerical


N=1
Analytical
Numerical
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
u(ξ)
−4
−3
−2
−1
0
1
2
3
4
ξ

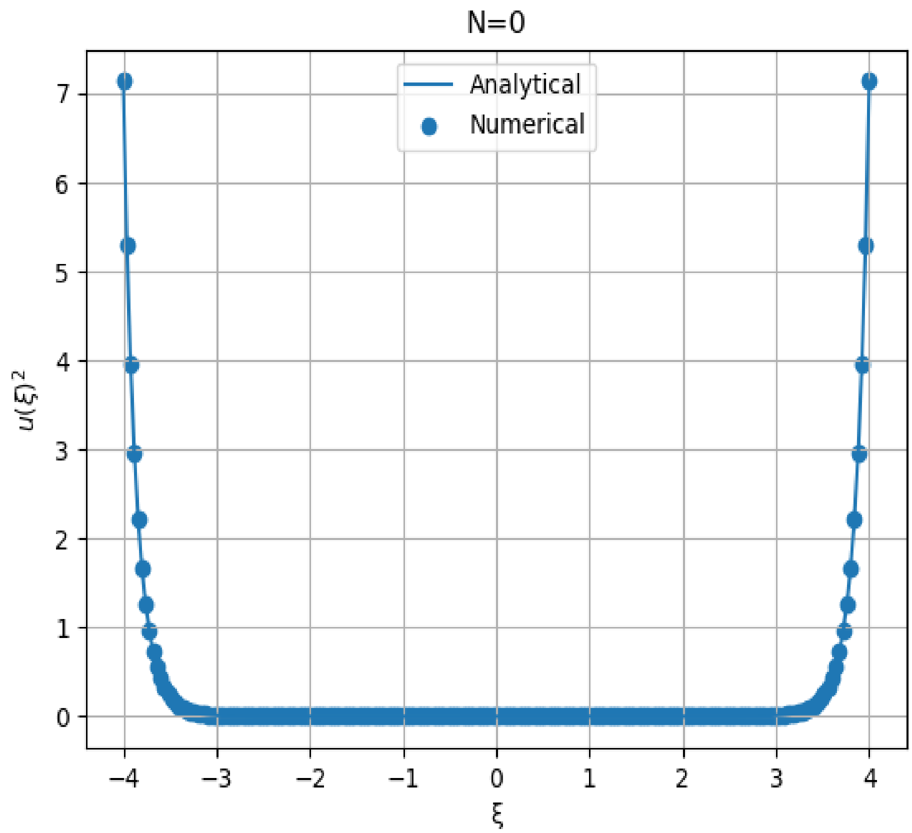


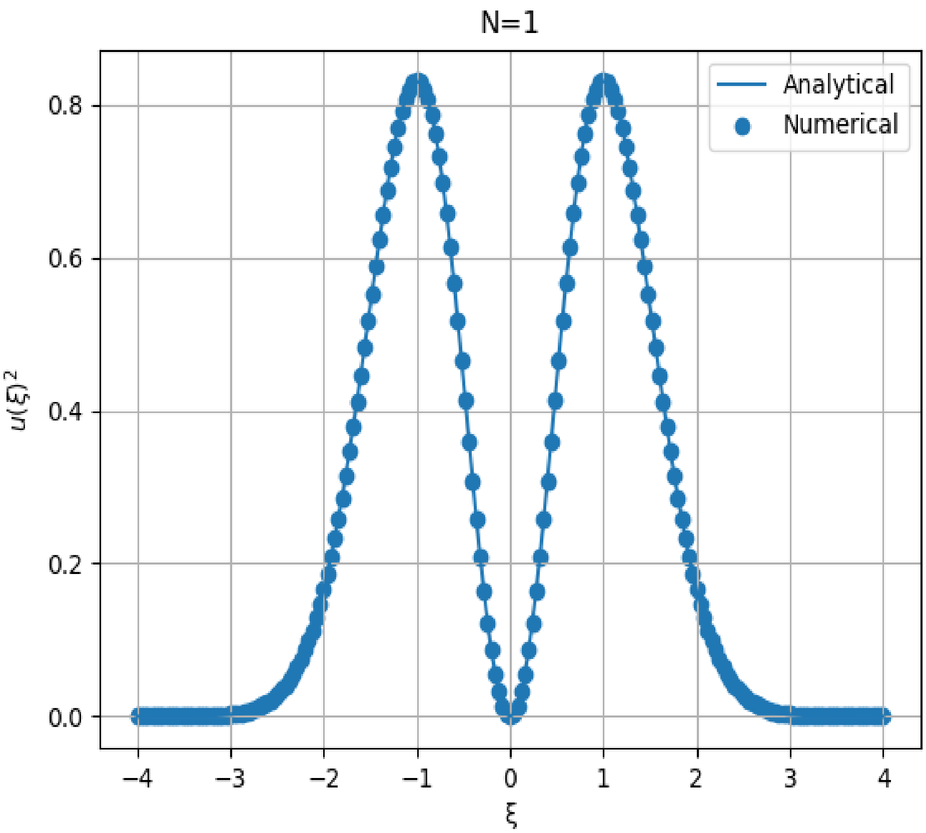
N=2
0.0
−0.5
−1.0
−1.5
−2.0
u(ξ)
−4
−3
−2
−1
0
1
2
3
4
ξ
Analytical
Numerical


N=3
Analytical
Numerical
u(ξ)
ξ
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−4
−3
−2
−1
0
1
2
3
4

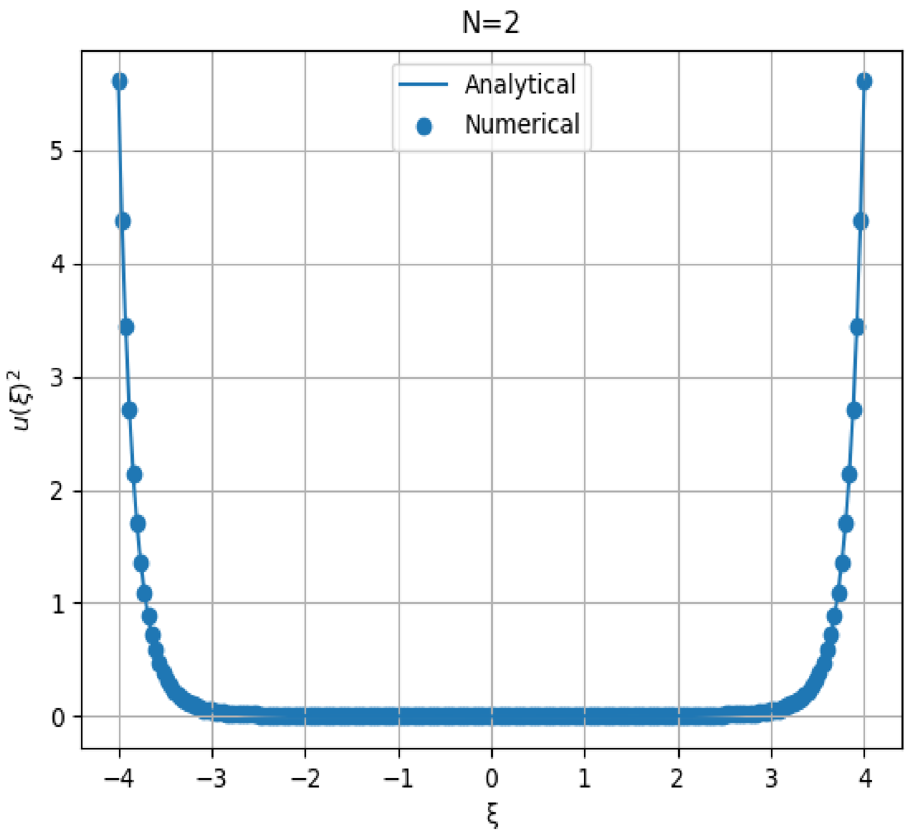


x

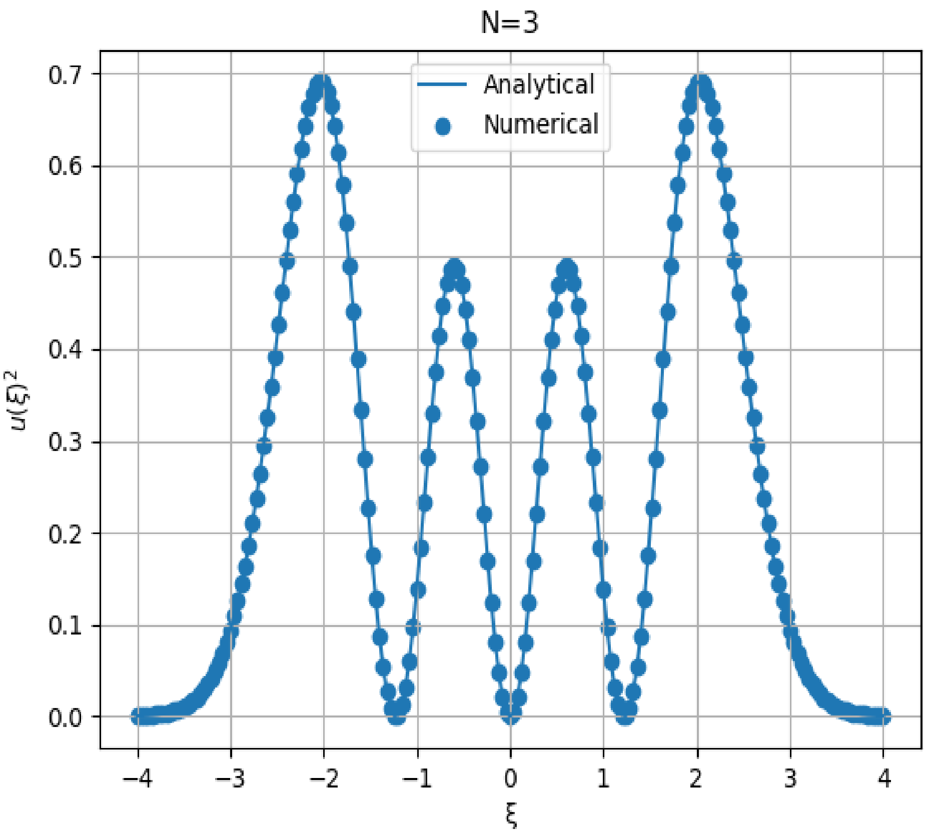

N=0
Analytical
Numerical
$u(\xi)^2$
ξ

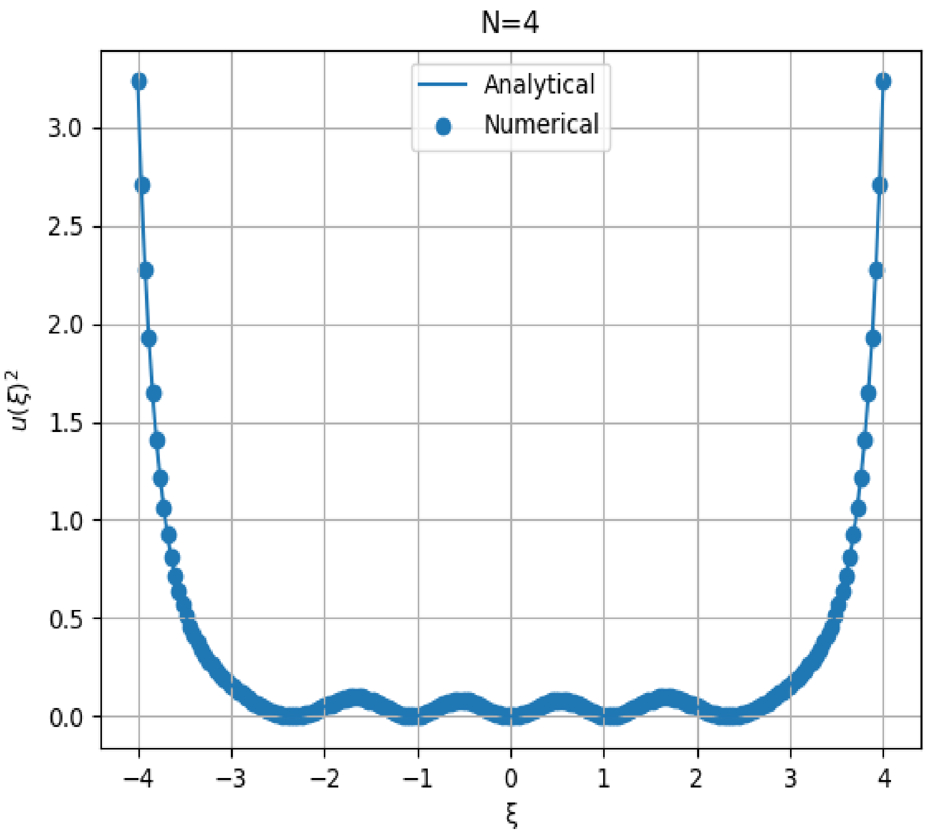


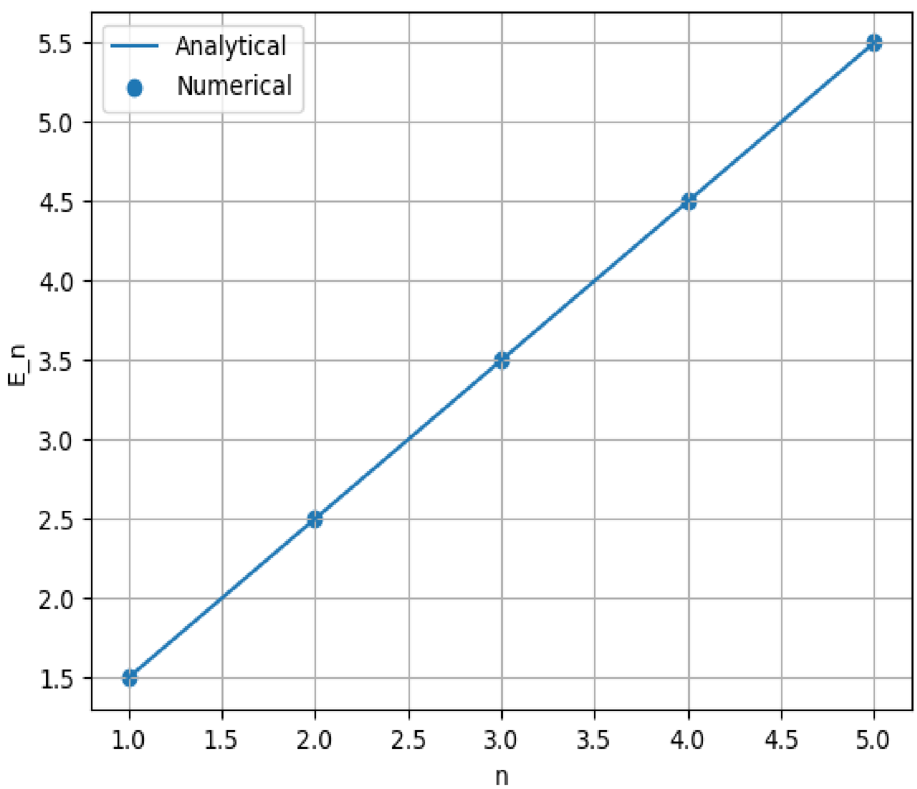
N=1
Analytical
Numerical
$u(\xi)^2$
0.0
0.2
0.4
0.6
0.8
−4
−3
−2
−1
0
1
2
3
4
ξ

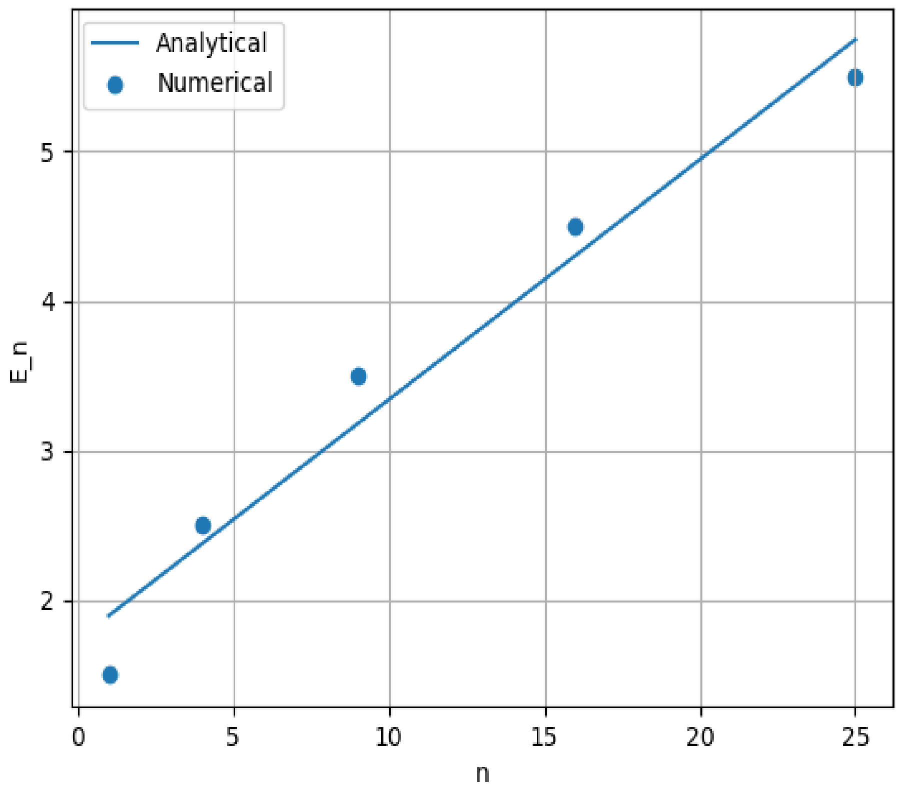


N=2
Analytical
Numerical
$u(\xi)^2$
ξ


N=3
Analytical
Numerical
$u(\xi)^2$
ξ
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
−4
−3
−2
−1
0
1
2
3
4


N=4
Analytical
Numerical
3.0
2.5
2.0
1.5
1.0
0.5
0.0
$u(\xi)^2$
−4
−3
−2
−1
0
1
2
3
4
ξ


Analytical
Numerical
E_n
n
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0



Analytical
Numerical
5
4
3
2
E_n
0
5
10
15
20
25
n