NAME: ANURAG DAS

ROLL No.: 2020PHY1116

QUANTUM MECHANICS (LAB)

SEMESTER - V

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate
import pandas as pd
from scipy.special import hermite
from scipy.stats import linregress


u_odd = [0, 0.001]
u_even=[1,1.001]

# POTENTIAL FUNCTION
def V2(x):
    V = (x**2)/2
    return V

# NUMEROV
def numerov(a,b,e,i_cs, n, V):

    x = np.linspace(a, b, n)
    u0, u1 = i_cs[0], i_cs[1]

    Vx = []
    for i in x:
        Vx.append(V(i))
    Vx = np.array(Vx)
    alpha = 2*(-Vx + e + 0.5)
    h=x[1]-x[0]
    #print(h)
    ci = 1 + (((h ** 2) / 12) * alpha)
    p = 0
    u = [u0, u1]
    for i in range(1,len(x)-1):
        p+=1
        u2 = (((12 - 10*ci[i]) * u[i]) - (ci[i - 1] * u[i-1])) / (ci[i + 1])
        u.append(u2)
        if p==len(x)-2:
            break
    # NORMALIZATION
    c = integrate.simps(np.array(u) ** 2, x)
    N = u / (np.sqrt(c))
```

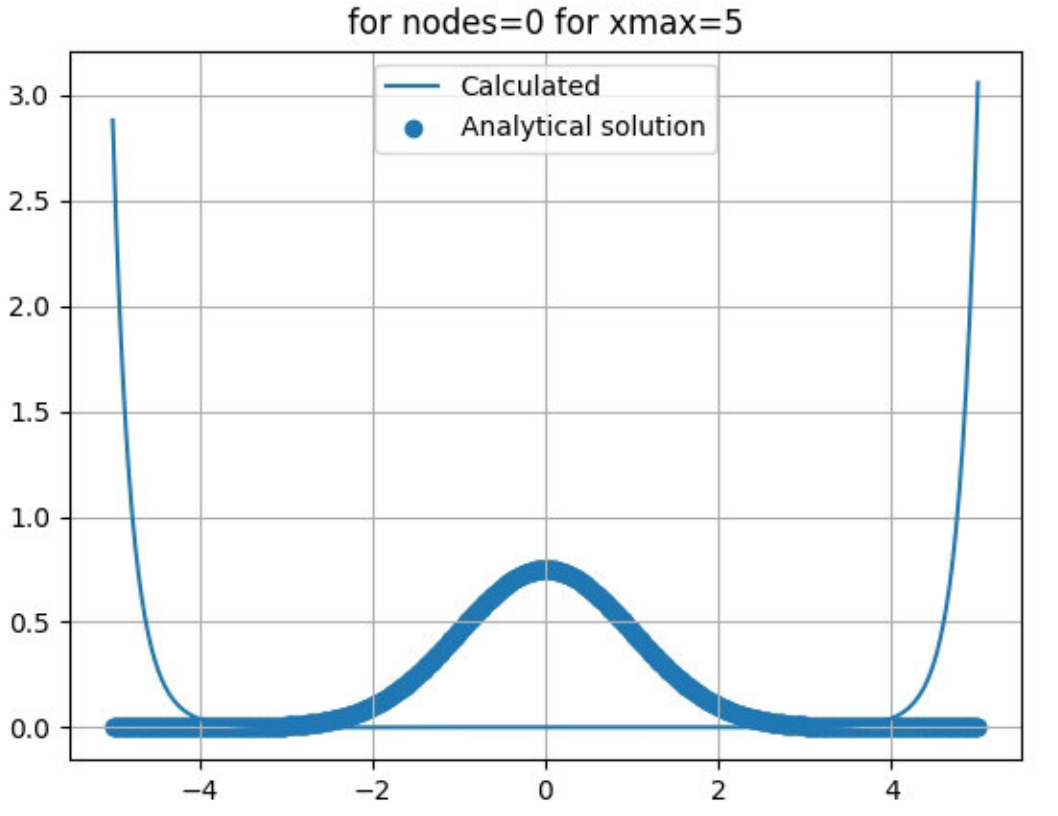
```
    #x.append(t[-1])
    return x,np.array(N)

# FINDING EIGENVALUE FOR REQ NODES

def energy(u,ics,emax,nodes,tol):
    emin=0
    p=0
    while abs(emax-emin)>=tol:
        p+=1
        u2=u[:-1]
        u3=u[1:]
        count=0
        #print("emin",emin)
        for i,j in zip(u2,u3):
            if i*j<0:
                count+=1

        #print(int(nodes/2))
        if count<=int(nodes/2):
            emin=(emax+emin)/2
        else:
            emax=(emax+emin)/2
        e_n=(emax+emin)/2
        u=numerov(0,4,e_n,ics,400,V2)[1]
    return (emax+emin)/2 + 0.5,p,emin,emax

#PARITY
def parity(n,u):
    if n%2==0:
        p_u=u[::-1]
        p_u1=p_u[1:]
    else:
        p_u=-1*u[::-1]
        p_u1 =p_u[1:]
    return p_u1

def u_total(n,u):
    return np.concatenate((parity(n,u),u))

def analytic(x,n):
```

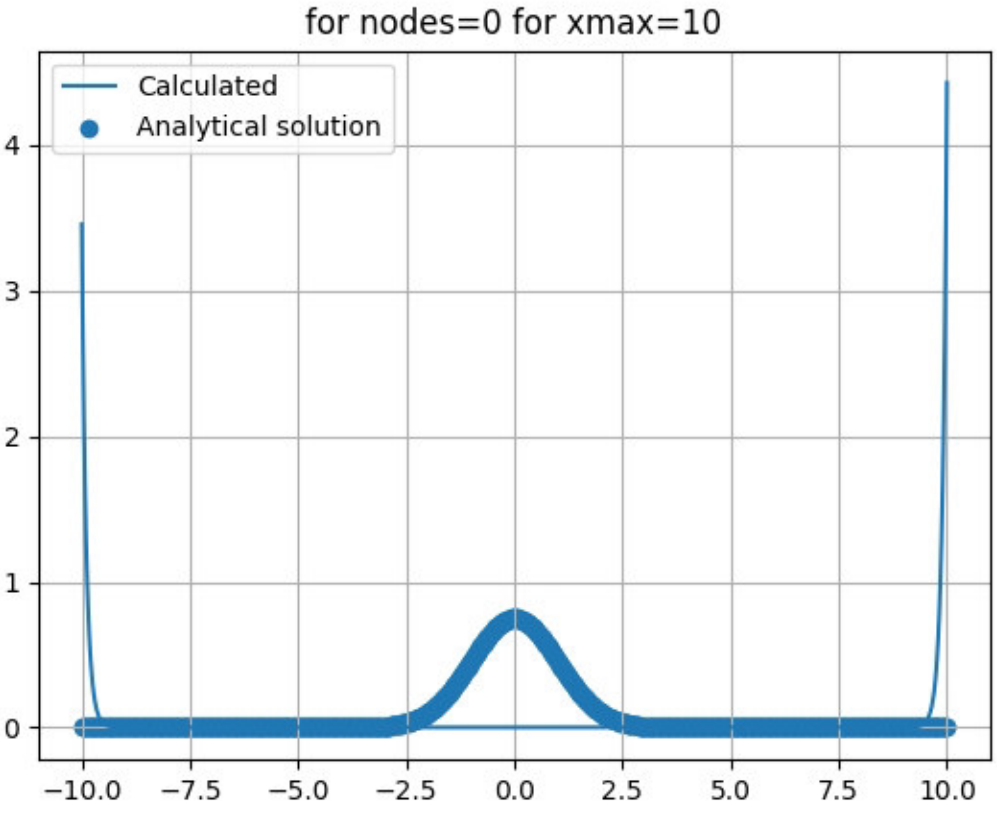
```
    Hn=hermite(n)
    psi=(np.e)**((-(x**2))/2) * Hn(x)
    c = integrate.simps(psi ** 2, x)
    psi_n= psi/np.sqrt(c)
    return psi_n

# A-(ii)
nodes=[0,1,2,3,4,5]
e_n=[]
for i in nodes:
    if i%2==0:
        u=numerov(0,8,4,u_even,400,V2)[1]
        e_n.append(energy(u,u_even,8,i,0.5*(10**(-10)))[0])
    else:
        u=numerov(0,8,4,u_odd,400,V2)[1]
        e_n.append(energy(u, u_odd, 8, i, 0.5 * (10 ** (-10)))[0])

# (a)- (iv)
e_analytic=[0.5,1.5,2.5,3.5,4.5,5.5]
print(pd.DataFrame({'Nodes':nodes,'Eigen Values':e_n,'Analytical Values':
e_analytic}))

# (a)-(iii)
plt.plot(nodes,e_n)
plt.grid()
plt.xlabel('nodes')
plt.ylabel('e_n')
plt.title("e_n VS nodes")
plt.show()
slope_n = linregress(np.array(nodes), np.array(e_n))[0]
intercept_n = linregress(np.array(nodes), np.array(e_n))[1]
print('slope for en vs n: ',slope_n)
print('intercept for en vs n: ',intercept_n)

# (b)

plt.plot(np.array(nodes)**2,e_n)
plt.grid()
plt.xlabel('Nodes Square')
plt.ylabel('e_n')
plt.title("e_n VS nodes Square")
```

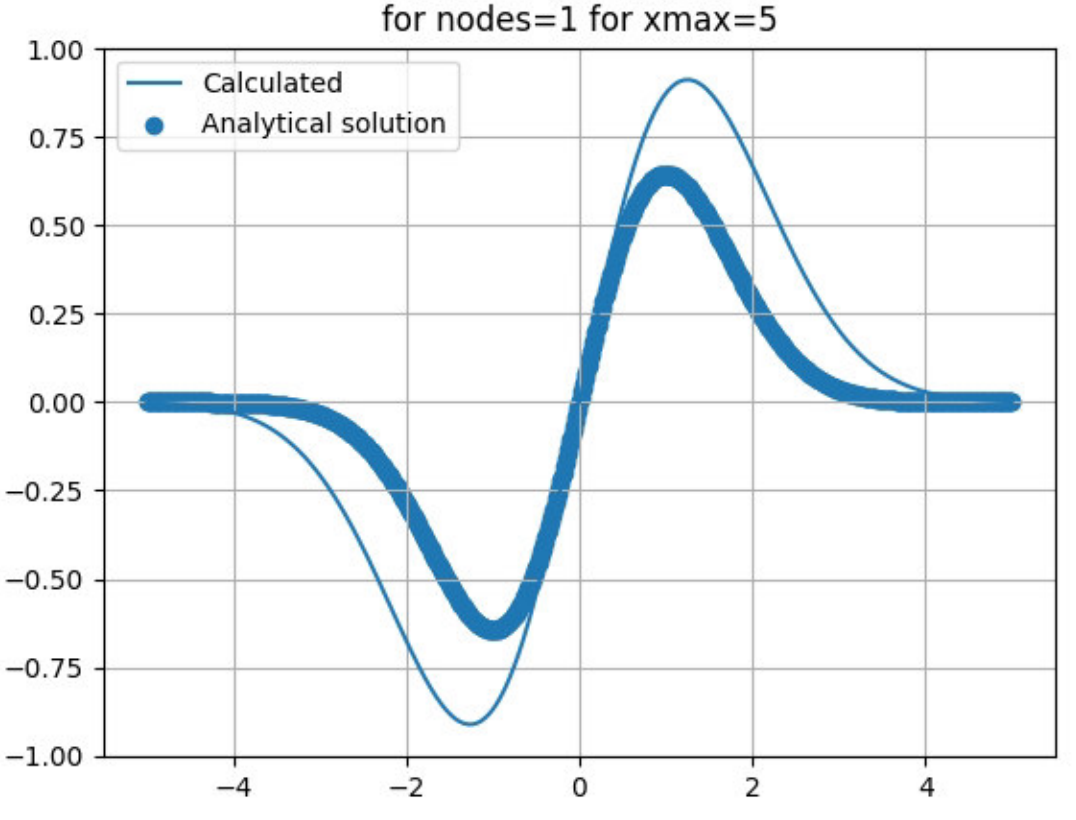
```
slope_n2 = linregress(np.array(nodes)**2, np.array(e_n))[0]
intercept_n2 = linregress(np.array(nodes)**2, np.array(e_n))[1]

fitted_en = slope_n2*(np.array(nodes)**2 + intercept_n2)
plt.scatter(np.array(nodes)**2, fitted_en, label='fitted curve')
plt.show()

print('slope for en vs n**2: ',slope_n2)
print('intercept for en vs n**2: ',intercept_n2)

#(c)

def plot(power, xmax):
    x = np.linspace(-xmax, xmax, 799)
    for i in nodes:
        if i % 2 == 0:
            u = numerov(0, xmax, i, u_even, 400, V2)[1]
            #u_total(i,u)
            if i==0 or 4:
                plt.plot(x,u_total(i,u)**power,label='Calculated')
            else:
                plt.plot(x, (-u_total(i, u))**power,label='Calculated')
            plt.scatter(x,analytic(x,i)**power,label='Analytical solution')
            plt.title('for nodes='+str(i)+ ' for xmax='+ str(xmax))
            plt.grid()
            plt.legend()
            plt.savefig('psi' + str(power) + ' nodes=' + str(i) + 'xmax=' + str(xmax))
            plt.show()
        else:
            u = numerov(0, 4, i, u_odd, 400, V2)[1]
            if i==3:
                plt.plot(np.linspace(-xmax,xmax,799),(-u_total(i,u))**power,label='Calculated')
            else:
                plt.plot(np.linspace(-xmax,xmax,799),u_total(i,u)**power,label='Calculated')
            plt.scatter(x, analytic(x, i)**power,label='Analytical solution')
            plt.title('for nodes=' + str(i)+' for xmax='+ str(xmax))
            plt.grid()
            plt.legend()
            plt.savefig('psi' + str(power) + ' nodes=' + str(i) + 'xmax=' + str(xmax))
            plt.show()
```

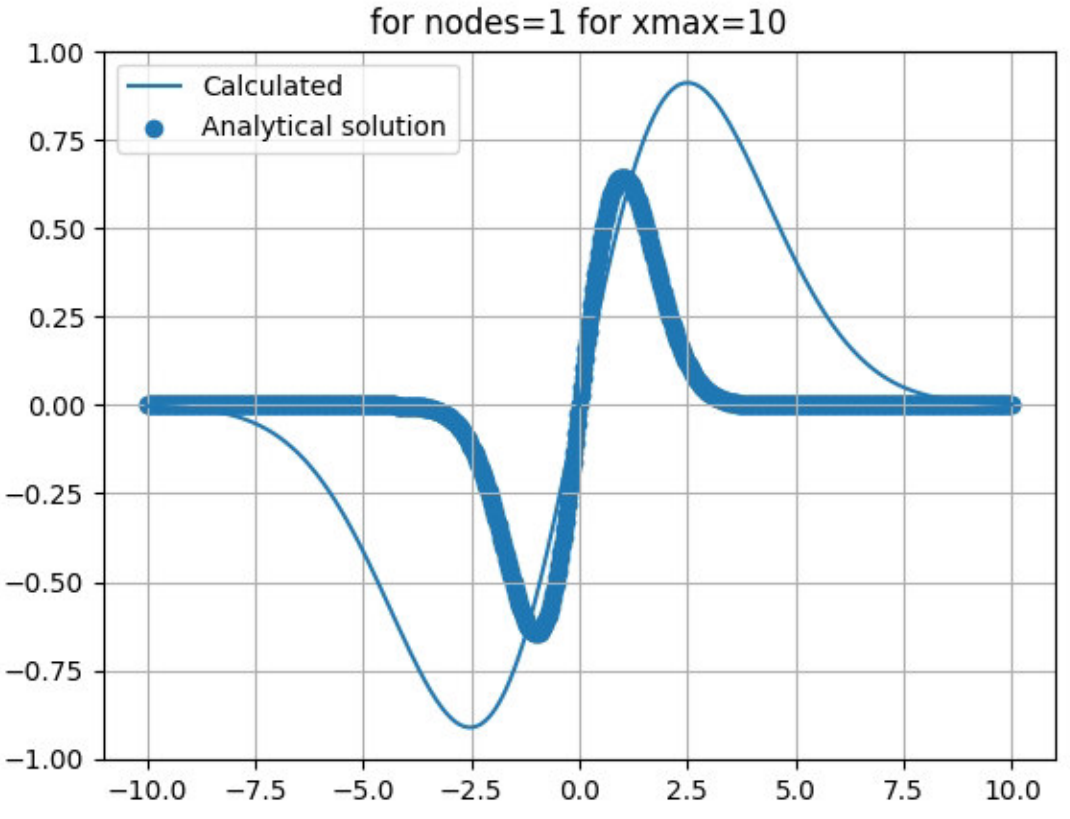
```
plot(1,5)
plot(2,5)
plot(1,10)
plot(2,10)
```

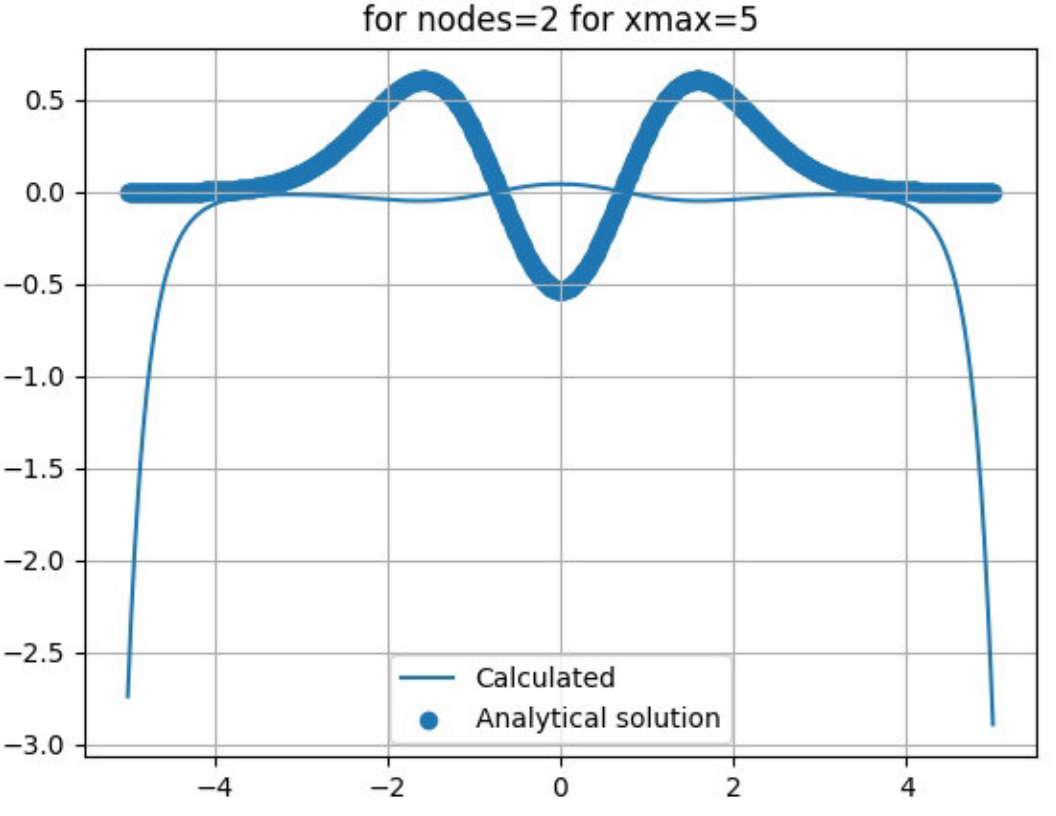
```
C:¥Users¥anura¥AppData¥Local¥Programs¥Python¥Python38¥
python.exe E:/SEM-5/A6/A6.3.py
   Nodes  Eigen Values  Analytical Values
0      0      0.557046                0.5
1      1      1.500015                1.5
2      2      2.535256                2.5
3      3      3.501692                3.5
4      4      4.540692                4.5
5      5      5.539422                5.5
slope for en vs n:  1.0000097950379963
intercept for en vs n:  0.5289958665761376
slope for en vs n**2:  0.18468954620903338
intercept for en vs n**2:  1.3360328472549892

Process finished with exit code 0
```

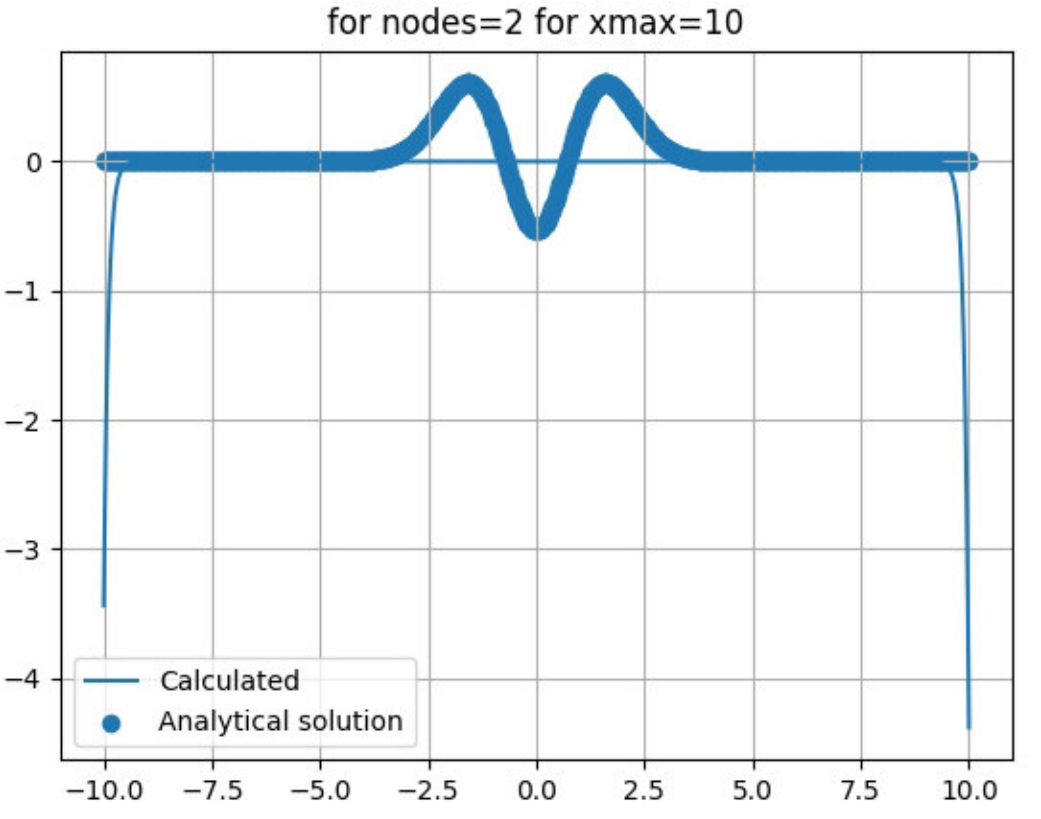

for nodes=0 for xmax=5
Calculated
Analytical solution
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−4
−2
0
2
4

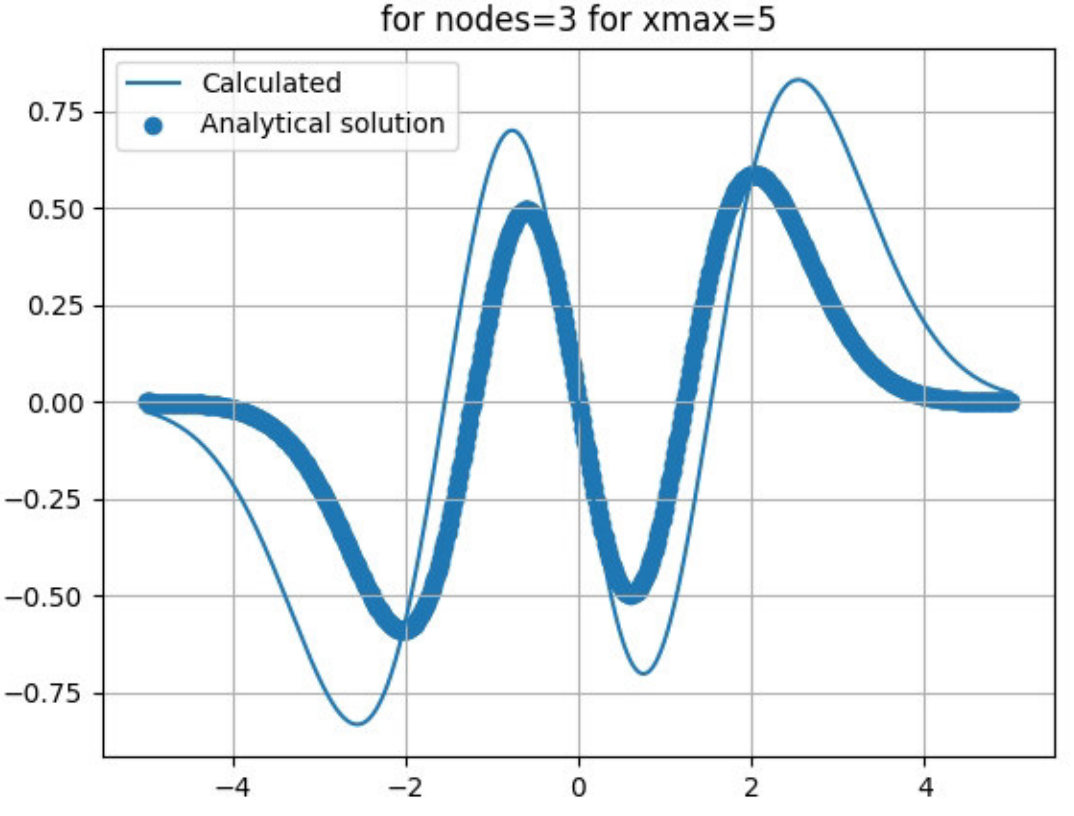

for nodes=0 for xmax=10
Calculated
Analytical solution
4
3
2
1
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

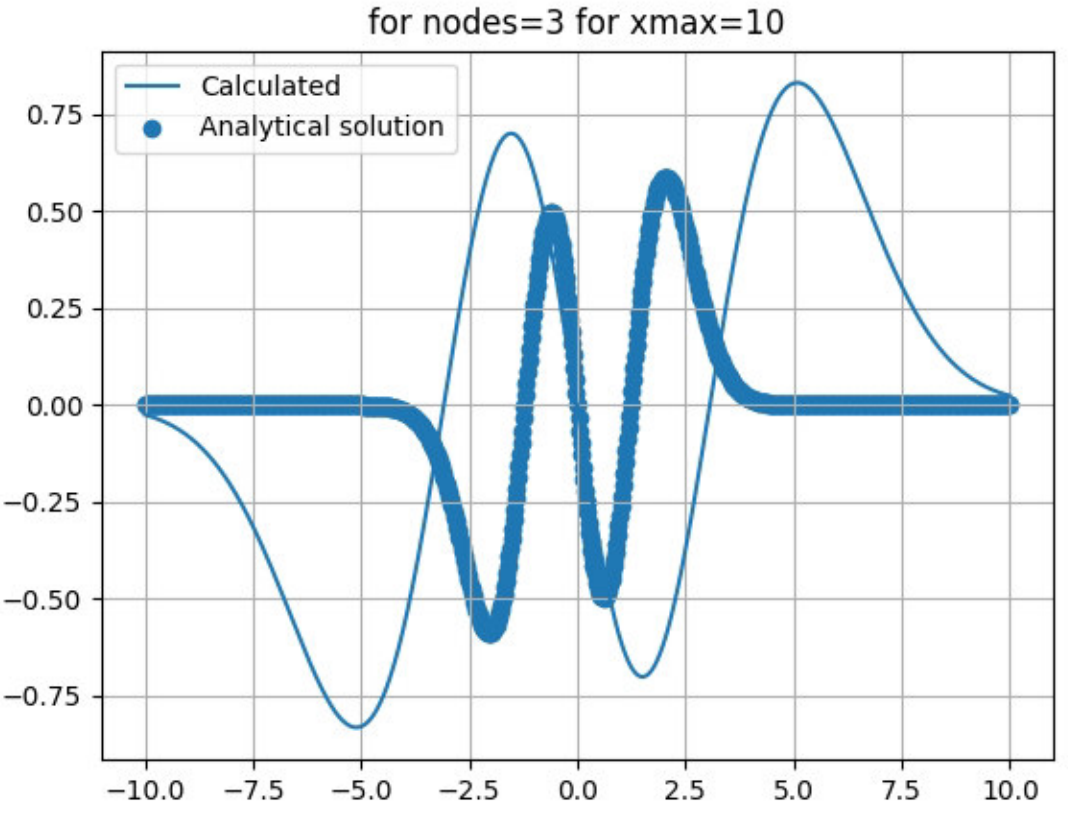

for nodes=1 for xmax=5
Calculated
Analytical solution
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−4
−2
0
2
4

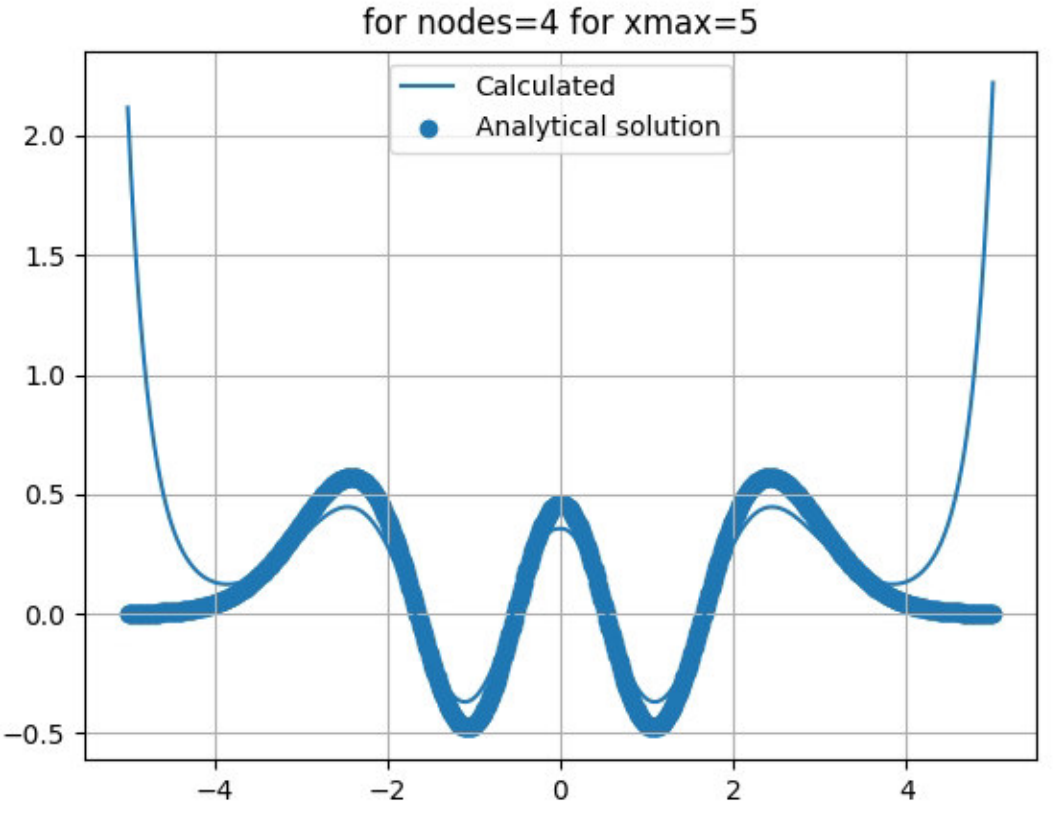

for nodes=1 for xmax=10
Calculated
Analytical solution
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

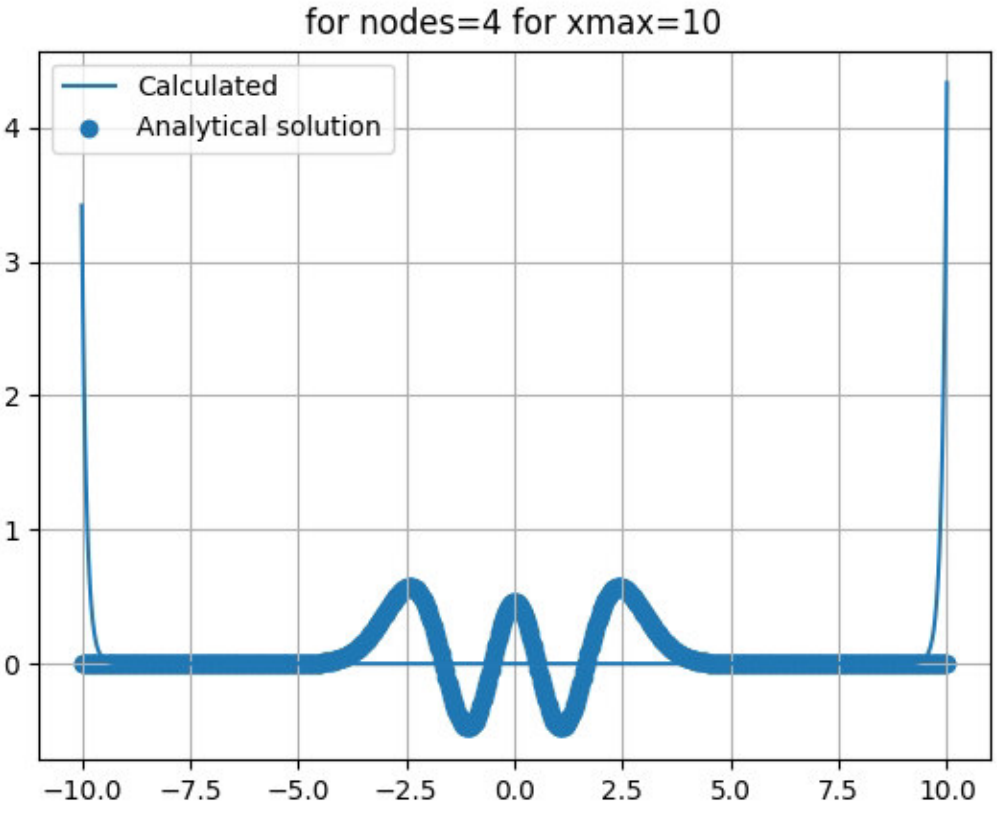

for nodes=2 for xmax=5
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
−4
−2
0
2
4
Calculated
Analytical solution

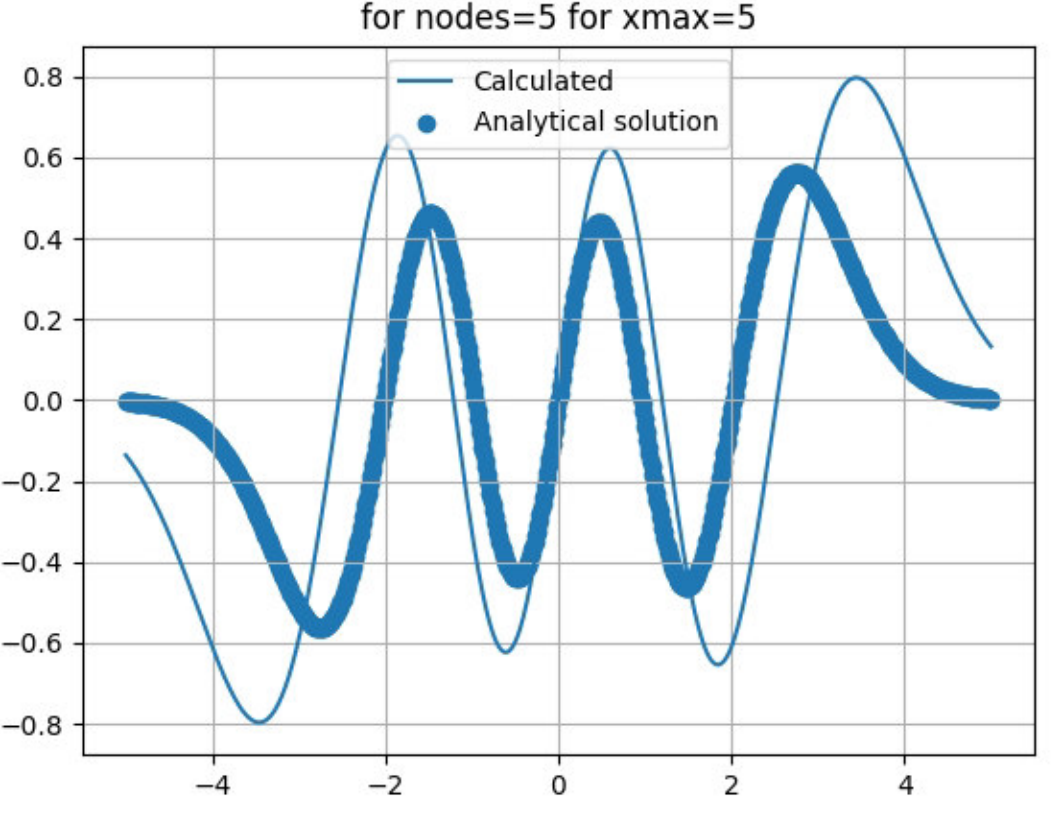

for nodes=2 for xmax=10
0
−1
−2
−3
−4
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Calculated
Analytical solution

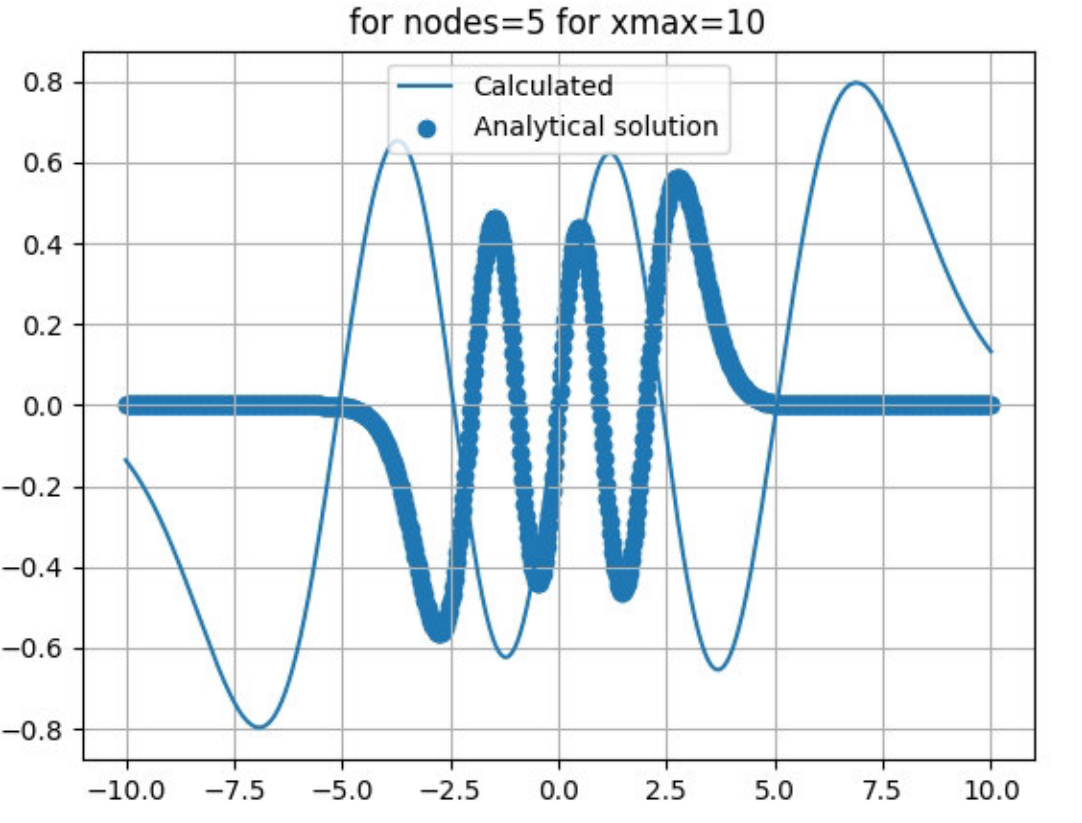

for nodes=3 for xmax=5
Calculated
Analytical solution
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−4
−2
0
2
4

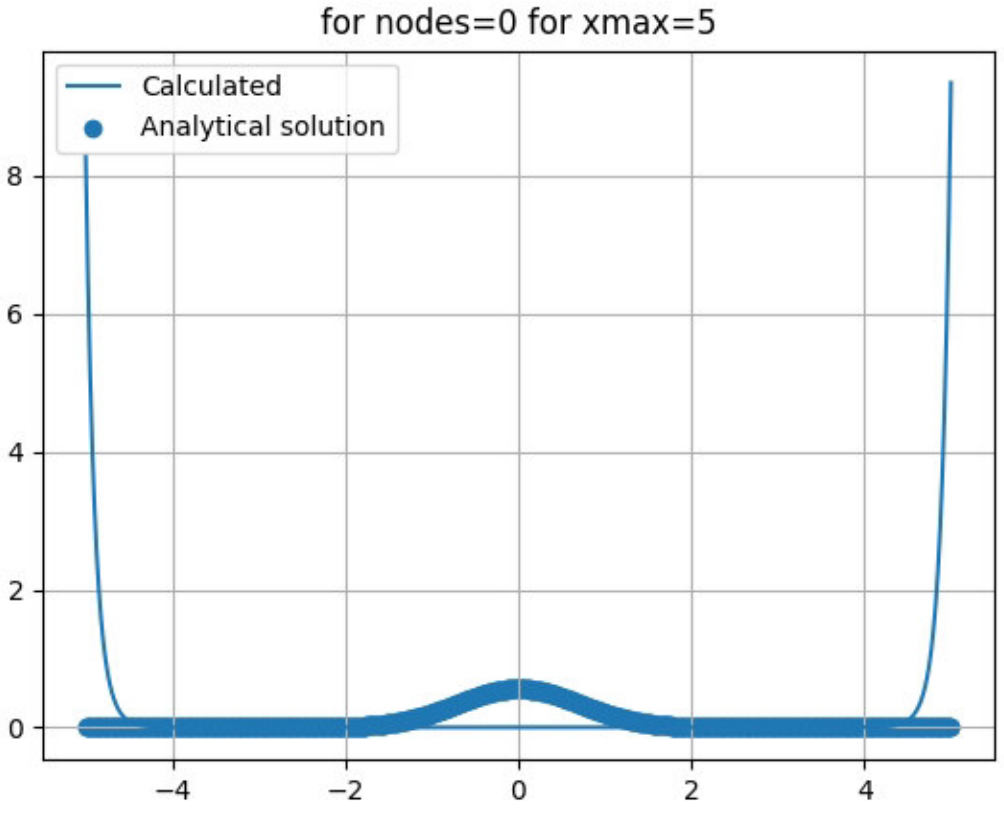

for nodes=3 for xmax=10
Calculated
Analytical solution
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

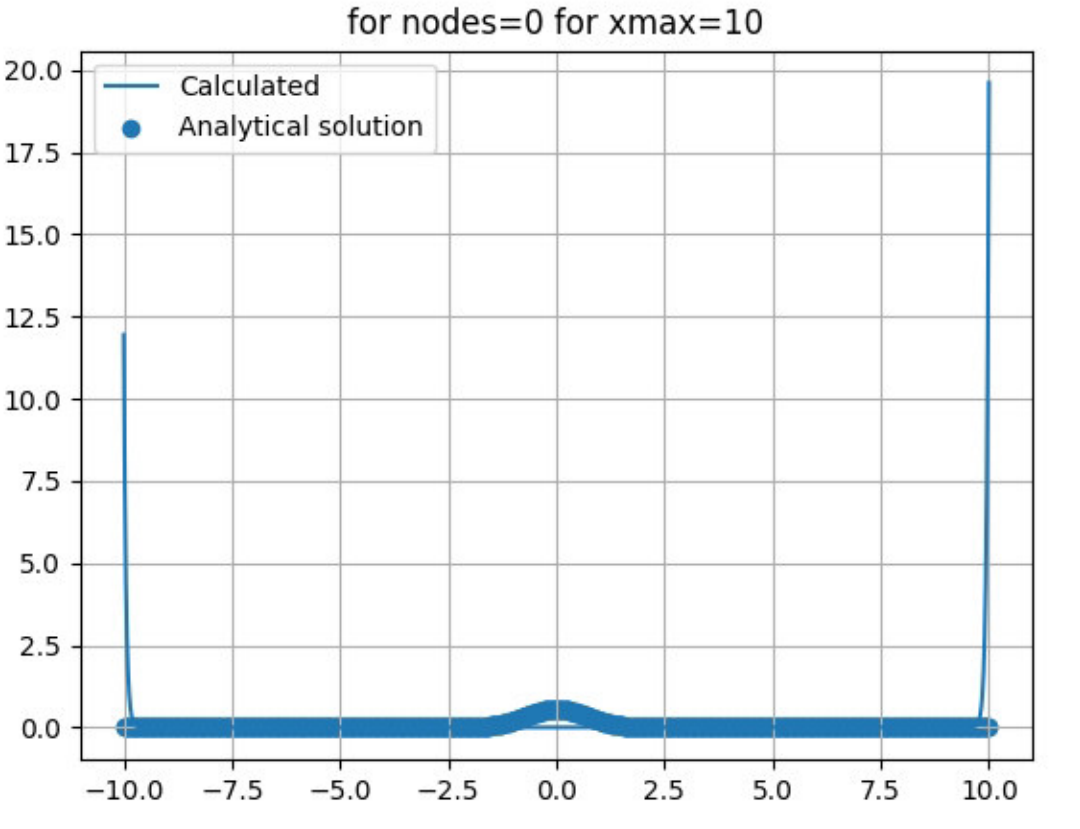

for nodes=4 for xmax=5
Calculated
Analytical solution
2.0
1.5
1.0
0.5
0.0
−0.5
−4
−2
0
2
4

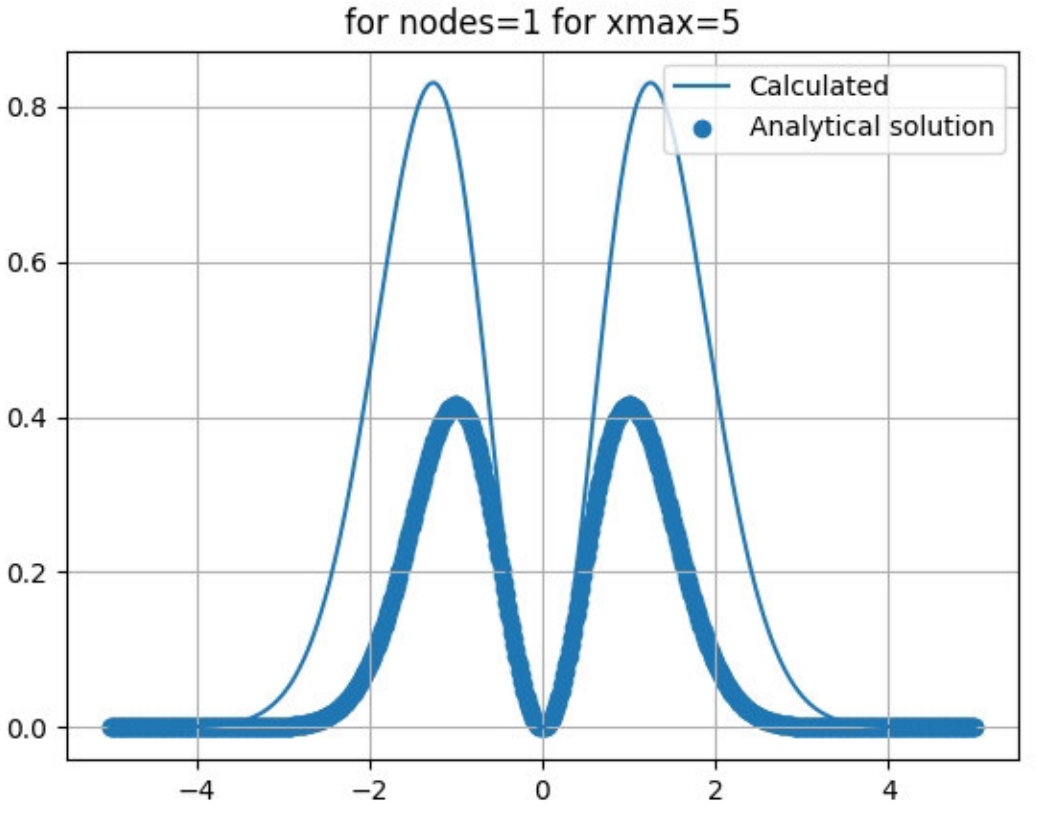

for nodes=4 for xmax=10
Calculated
Analytical solution
4
3
2
1
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

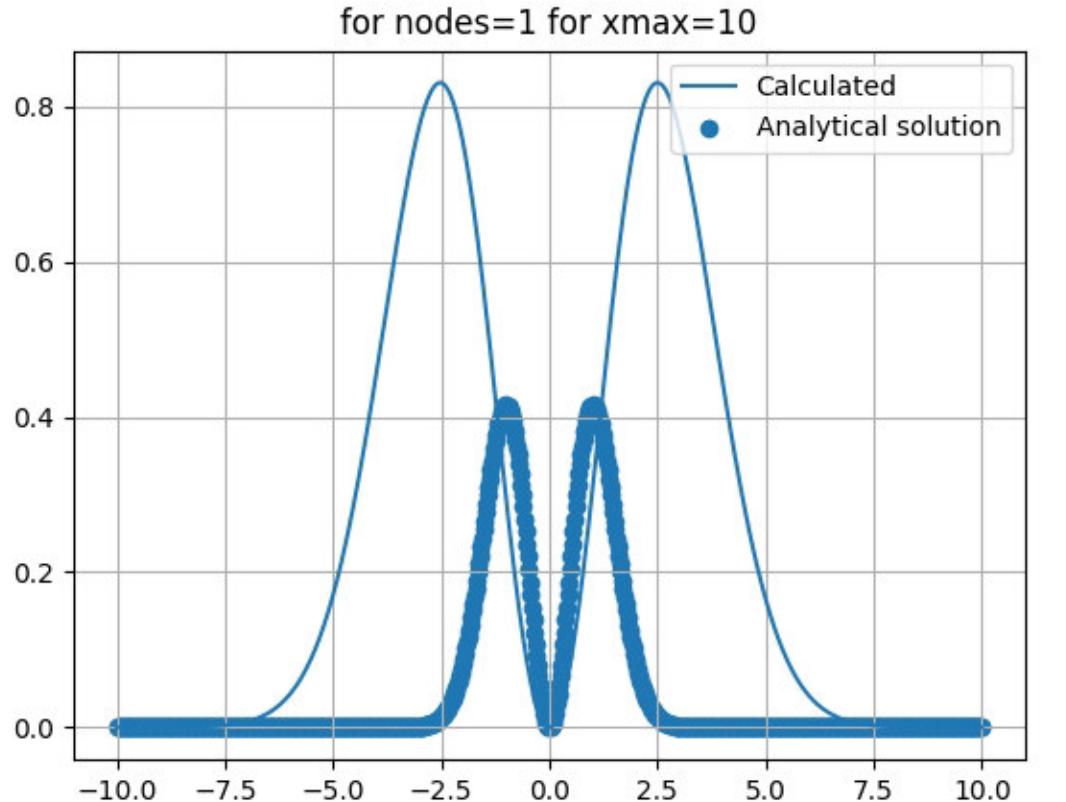

for nodes=5 for xmax=5
Calculated
Analytical solution
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−4
−2
0
2
4

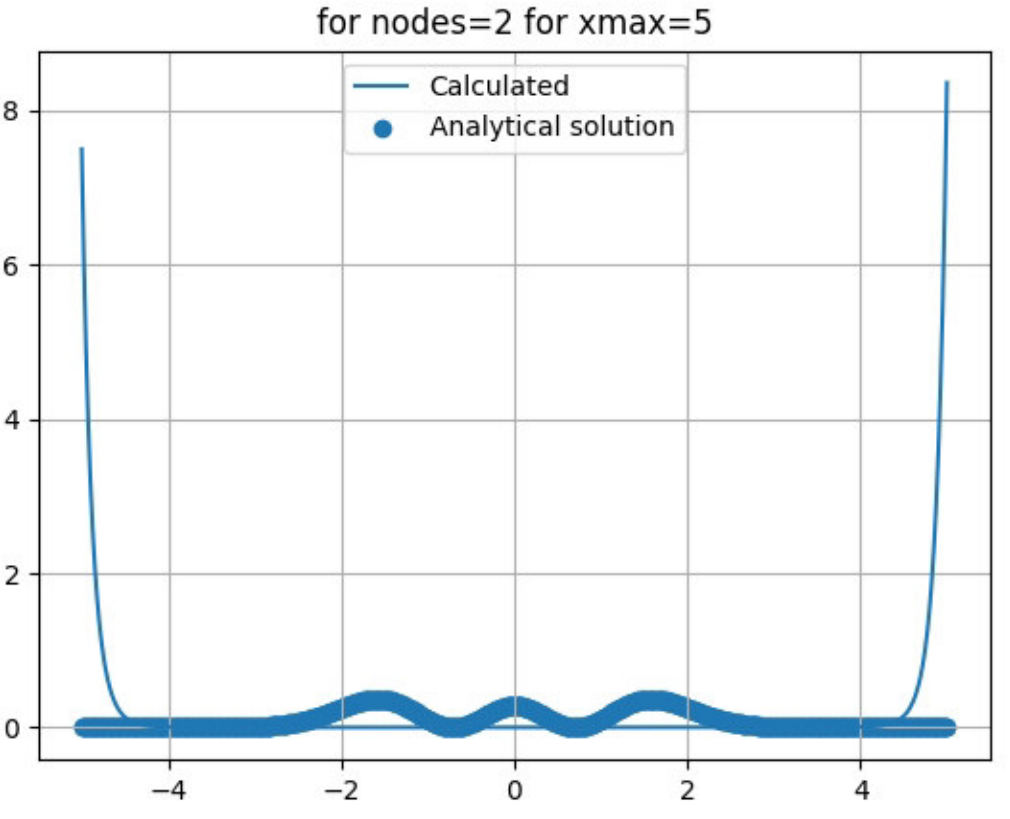

for nodes=5 for xmax=10
Calculated
Analytical solution
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

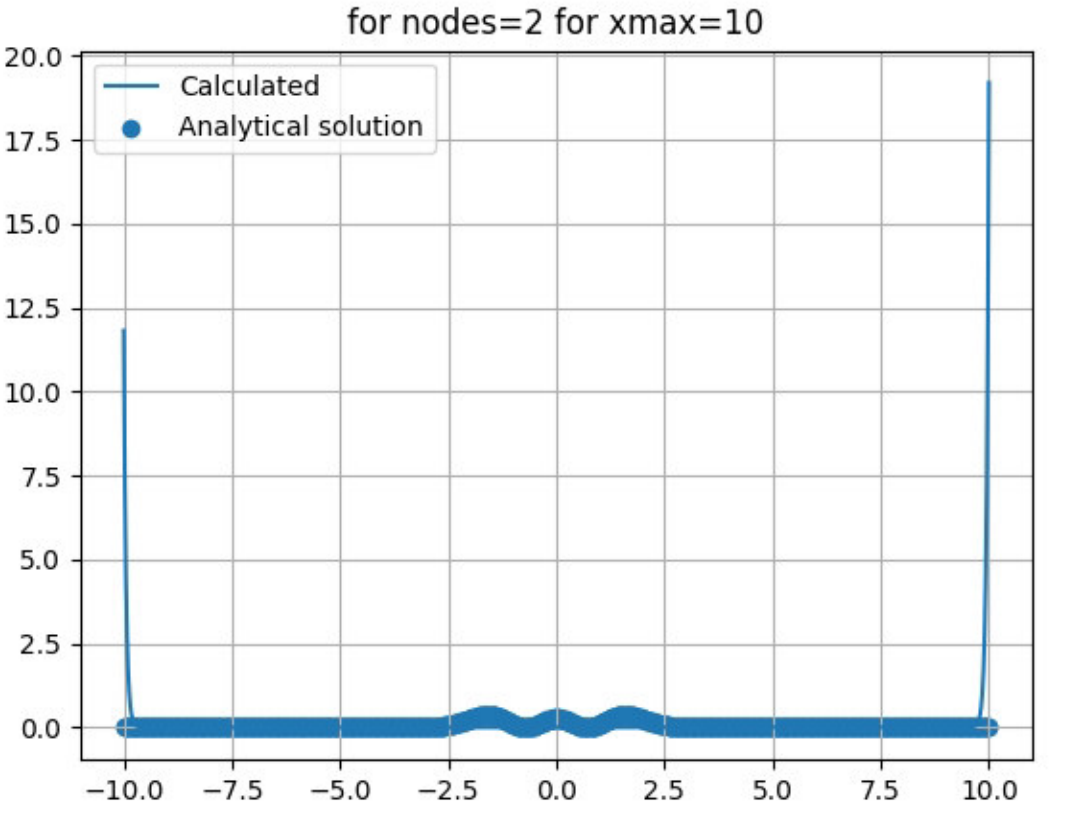

for nodes=0 for xmax=5
Calculated
Analytical solution
8
6
4
2
0
−4
−2
0
2
4

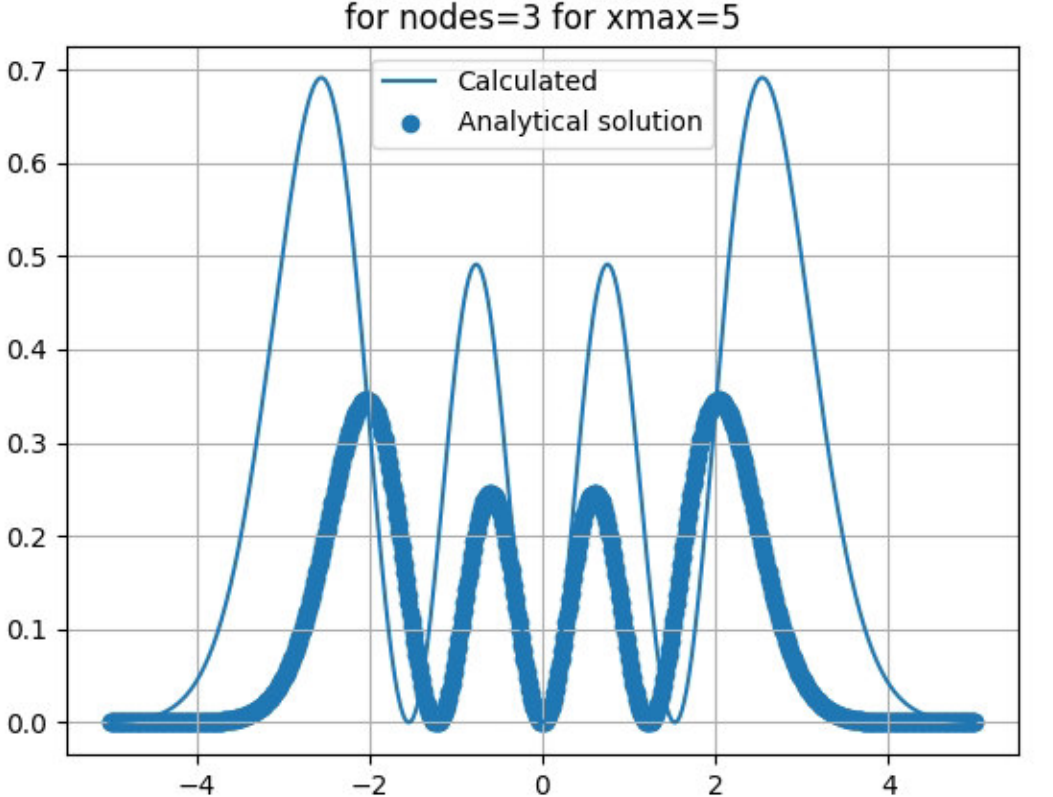

for nodes=0 for xmax=10
Calculated
Analytical solution
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

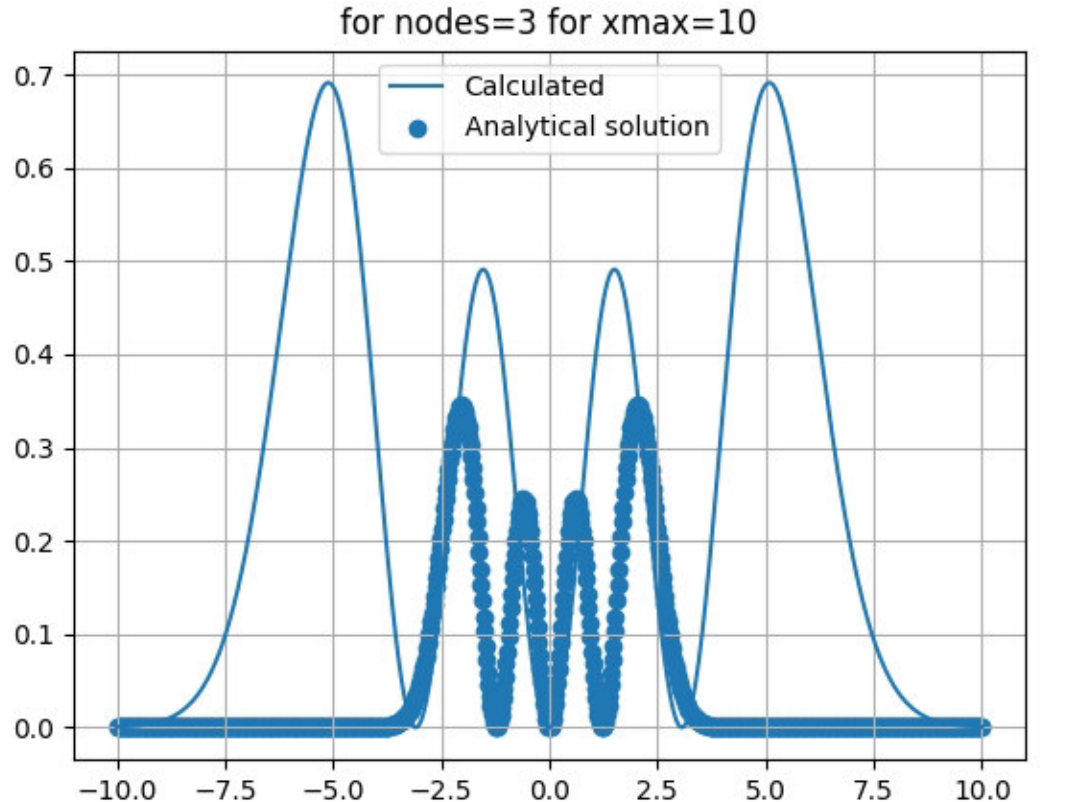

for nodes=1 for xmax=5
Calculated
Analytical solution
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

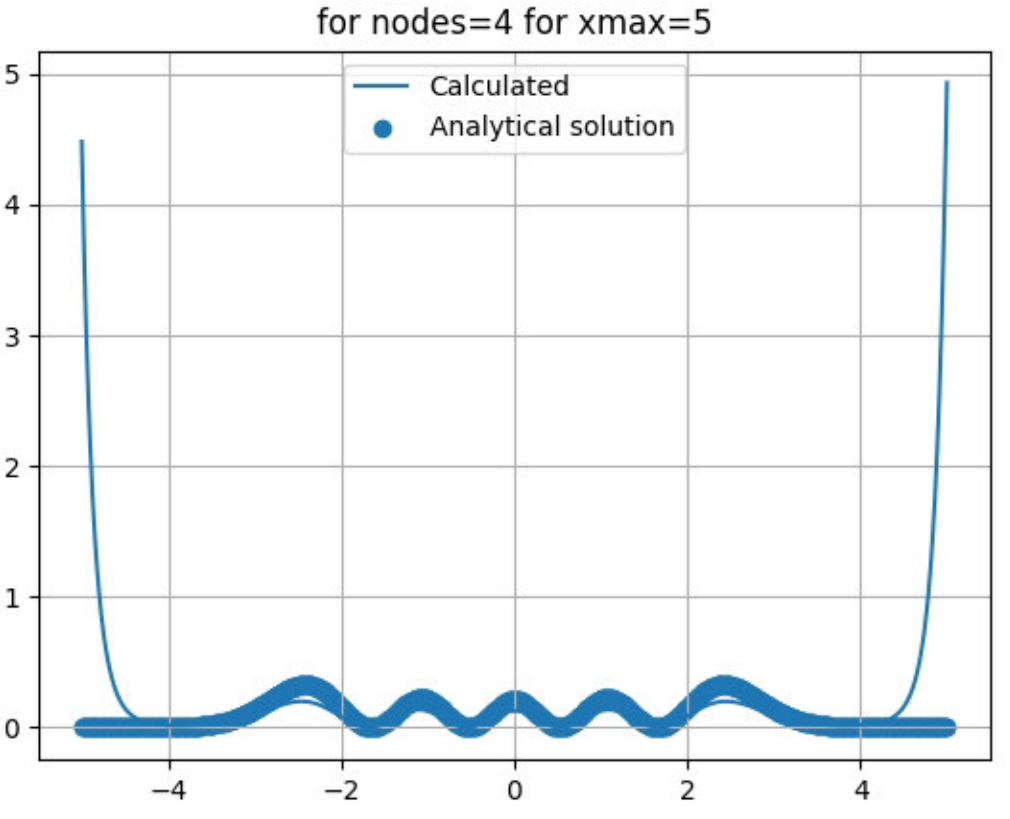

for nodes=1 for xmax=10
Calculated
Analytical solution
0.8
0.6
0.4
0.2
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

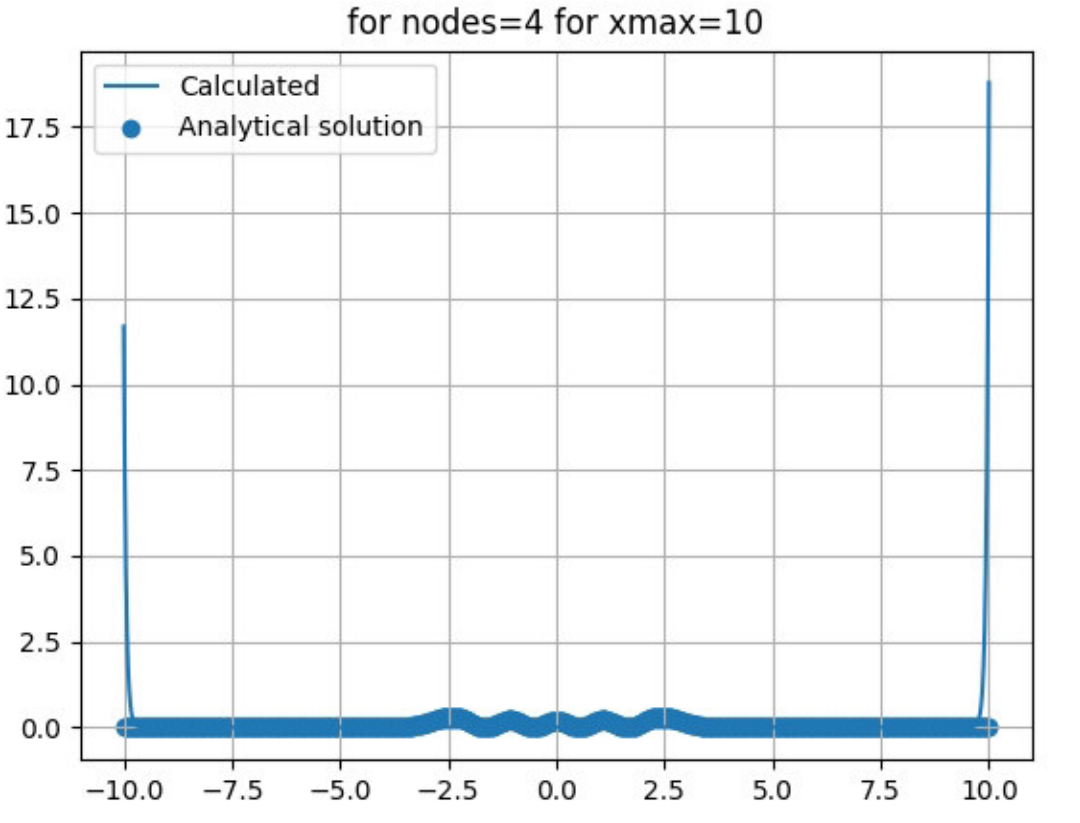

for nodes=2 for xmax=5
Calculated
Analytical solution
8
6
4
2
0
−4
−2
0
2
4

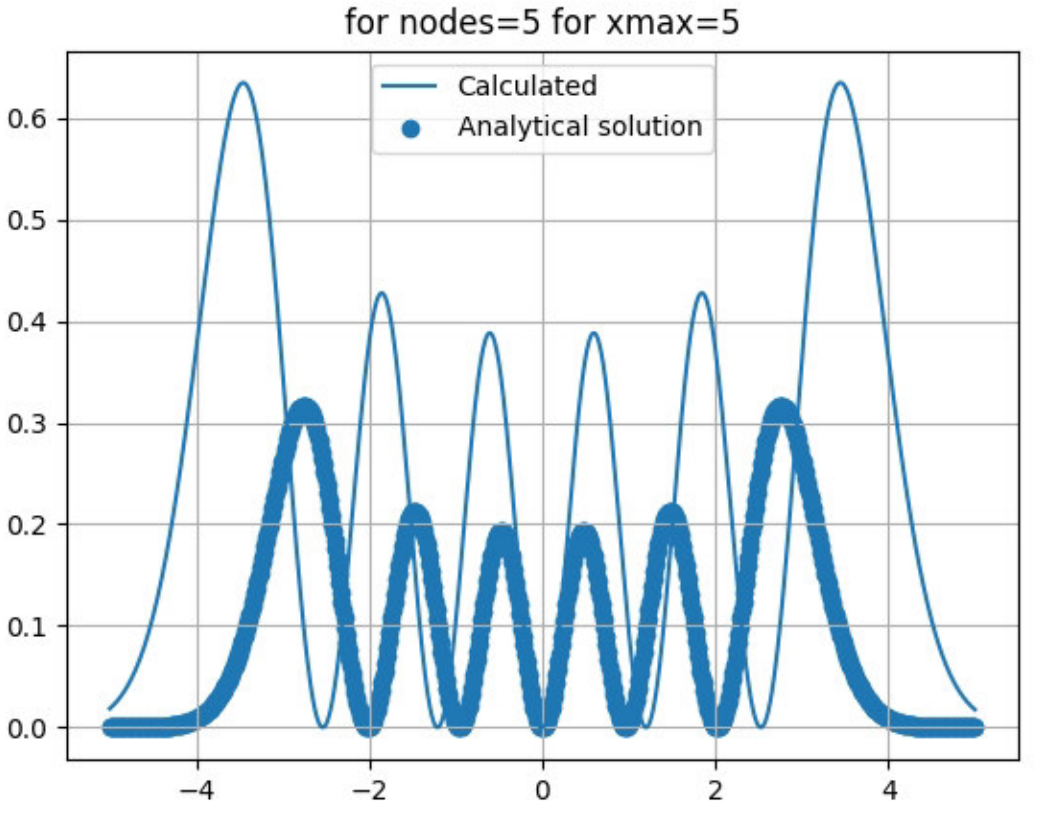

for nodes=2 for xmax=10
Calculated
Analytical solution
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

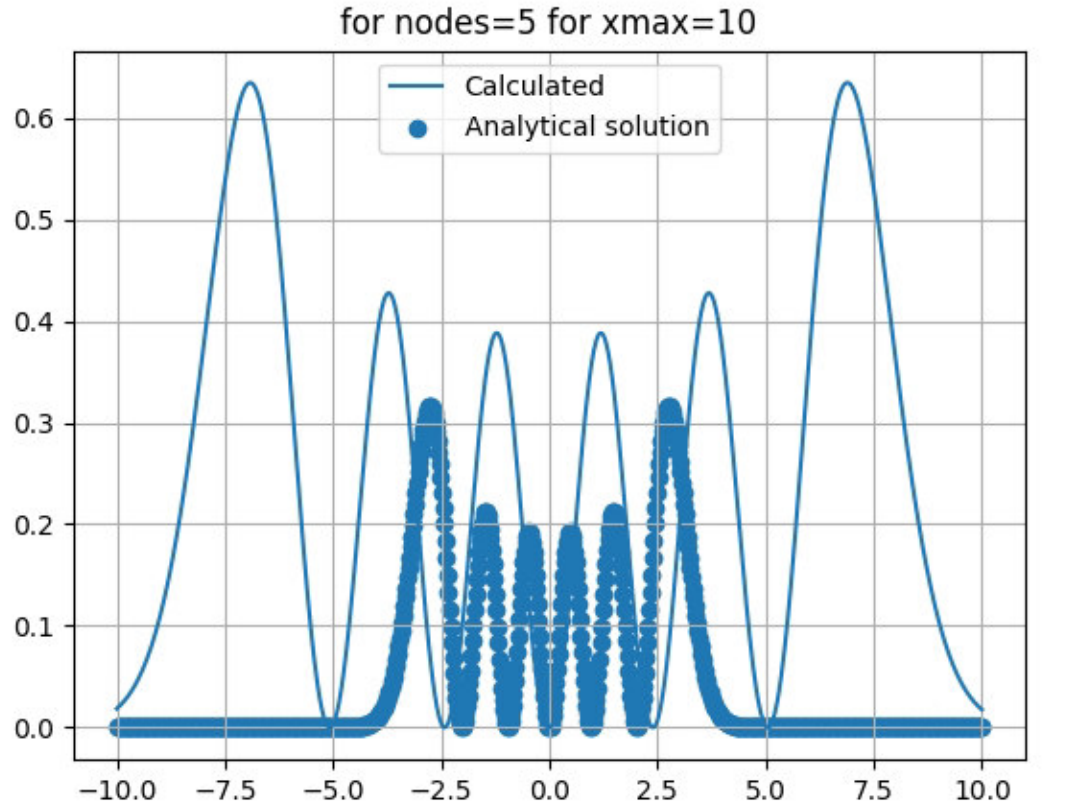

for nodes=3 for xmax=5
Calculated
Analytical solution
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−4
−2
0
2
4


for nodes=3 for xmax=10
Calculated
Analytical solution
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


for nodes=4 for xmax=5
Calculated
Analytical solution
5
4
3
2
1
0
−4
−2
0
2
4


for nodes=4 for xmax=10
Calculated
Analytical solution
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


for nodes=5 for xmax=5
Calculated
Analytical solution
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−4
−2
0
2
4


for nodes=5 for xmax=10
Calculated
Analytical solution
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0