Python code: Legendre Polynomials and Bessel Functions of first kind

Shyamal Bhar
Assistant Professor
Vidyasagar College for Women
Kolkata 700 006

Legendre Equation:

The Legendre equation is given by

$$\left(1-x^2\right)\frac{d^2y}{dx^2}-2x\frac{dy}{dx}+n\left(n+1\right)y=0$$

$P_n\left(x\right)$ satisfy the above equation.

$P_n\left(x\right)$ is called Legendre Polynomial of order n. This $P_n\left(x\right)$ can be obtained from Rodrigue's Formula:

$$P_n\left(x\right)=\frac{1}{n!2^n}\frac{d^n}{dx^n}\left(x^2-1\right)^n$$

Few Legendre Polynomials are:

$$P_0\left(x\right)=1 \qquad P_1\left(x\right)=x$$

$$P_2\left(x\right)=\frac{3x^2-1}{2} \qquad P_3\left(x\right)=\frac{5x^3-3x}{2}$$

Orthogonality of Legendre Polynomials:

$$\int_{-1}^{1}P_m\left(x\right)P_n\left(x\right)dx=\begin{cases}0, & m\neq n\\ \dfrac{2}{2n+1}, & m=n\end{cases}$$

Recurrence Formulae for $P_n\left(x\right)$:

1. $\left(1-x^2\right)P_n'\left(x\right)=\left(n+1\right)xP_n\left(x\right)-\left(n+1\right)P_{n+1}\left(x\right)$
2. $\left(n+1\right)P_{n+1}\left(x\right)=\left(2n+1\right)xP_n\left(x\right)-nP_{n-1}\left(x\right)$
3. $nP_n\left(x\right)=xP_n'\left(x\right)-P_{n-1}'\left(x\right)$
4. $\left(2n+1\right)P_n(x)=P'_{(n+1)}(x)-P'_{(n-1)}(x)$

5. $P_n'(x) = xP_{n-1}'(x) + nP_{n-1}(x)$

Bessel's Equation

The most important differential equations is

$$x^2 \frac{d^2y}{dx^2} + x\frac{dy}{dx} + \left(x^2 - n^2\right) y = 0$$

$J_n(x)$ is a solution of this equation. This is called Bessel's equation of first kind.

Recurrence formulae for $J_n(x)$:

1. $nJ_n(x) + xJ_n'(x) = xJ_{n-1}$
2. $J_n'(x) = \frac{1}{2}\left[J_{n-1}(x) - J_{n+1}(x)\right]$
3. $J_n'(x) = \frac{n}{x}J_n(x) - J_{n+1}(x)$
4. $\frac{d}{dx}\left[x^{-n}J_n(x)\right] = -x^{-n}J_{n+1}(x)$
5. $J_{n+1}(x) = \frac{2n}{x}J_n(x) - J_{n-1}(x)$

An important Property of Bessel functions:

$$J_{-n}(x) = (-1)^n J_n(x)$$

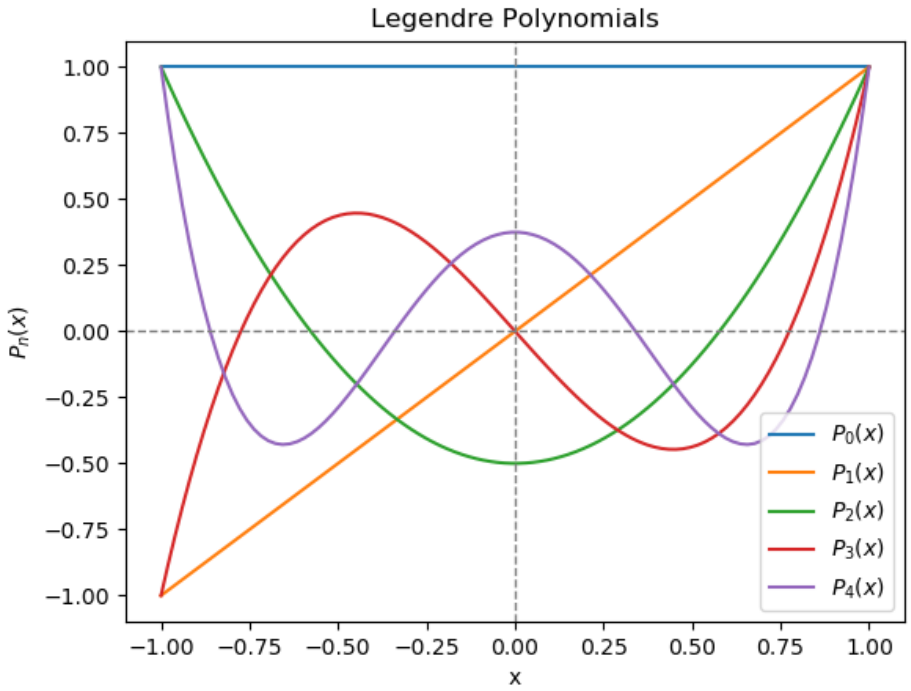
Orthogonalily of Bessel functions:

$$\int_0^1 xJ_n(\alpha x)J_n(\beta x)dx = \begin{cases} 0, & \alpha \neq \beta \\ \dfrac{\left[J_{n+1}(\alpha)\right]^2}{2}, & \alpha = \beta \end{cases}$$

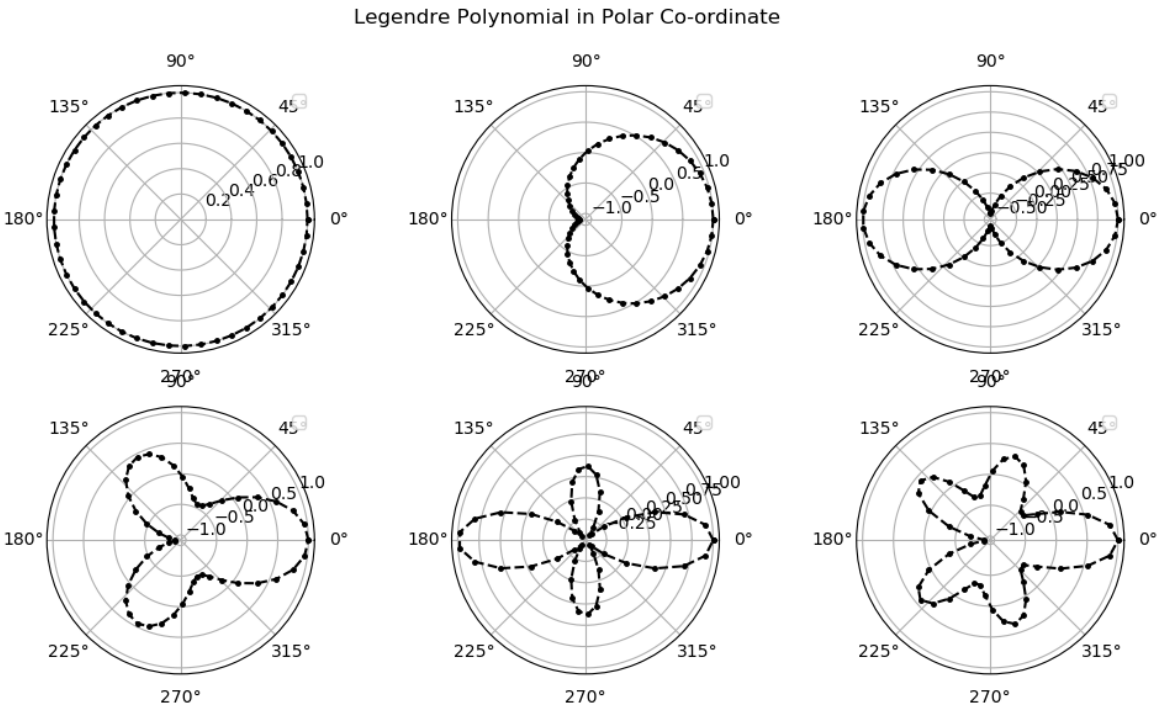
where α and β are roots of $J_n(x) = 0$

Python Code:

```
# Plotting Legendre Polynomials


import numpy as np
from scipy.special import legendre as P
import matplotlib.pyplot as plt

x=np.linspace(-1,1,100) # range of x values

N=4  # highest order of Polynomial

for n in range(N+1):
        plt.plot(x,P(n)(x),label=r"$P_%d(x)$"%(n))

plt.xlabel('x')
plt.ylabel('$P_n(x)$')
plt.title('Legendre Polynomials')
plt.axhline(0,c='gray',ls='--',lw=1)
plt.axvline(0,c='gray',ls='--',lw=1)
plt.legend(loc='best')
plt.show()
```



```
# Plotting of Legendre Polynomials in Polar Co-ordinate system

import matplotlib.pyplot as plt
import numpy as np
from scipy.special import legendre as P

a,b=2,3 # Number of rows and column of the figure
n = a*b ### number of Legendre Polynomials

# Theta values

th=np.linspace(0,2.*np.pi)

fig=plt.figure(figsize=(6*a,4*b))
plt.suptitle("Legendre Polynomial in Polar Co-ordinate")
for i in range(n):
    pn=P(i)(np.cos(th))

    plt.subplot(a,b,i+1, projection='polar')# choice of boxes
for subplots
    plt.plot(th, pn, 'k.',ls='--', ms = 5)
    plt.legend(loc='best',prop={'size':10})
plt.show()
```



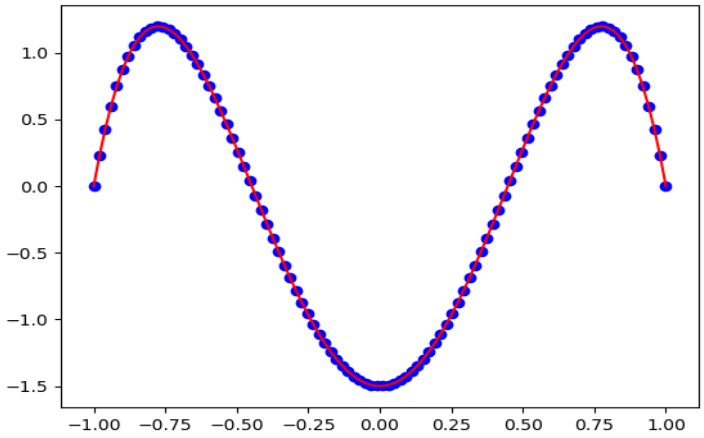
```
# Orthogonality of Legendre Polynomials
# To verify Integration(Pm(x)*Pn(x)dx)=2/(2n+1)delta(mn)


import numpy as np
from scipy.integrate import simps
from scipy.special import legendre as P
import matplotlib.pyplot as plt

m=2  # Values of m

x=np.linspace(-1,1,100) # range of x

# evaluating integration


 print ('m\t n\t delta_mn ')
 print '----------------------'

 for n in range(0,m+2):
         fun=P(m)*P(n)
         result=(2*n+1.)/2.*simps(fun(x),x)

         print ('%d\t%d\t%.4f '%(m,n,result))
```

Result:

```
 m     n    delta_mn
--------------------
 2     0     0.0000
 2     1    -0.0000
 2     2     1.0000
 2     3     0.0000
```

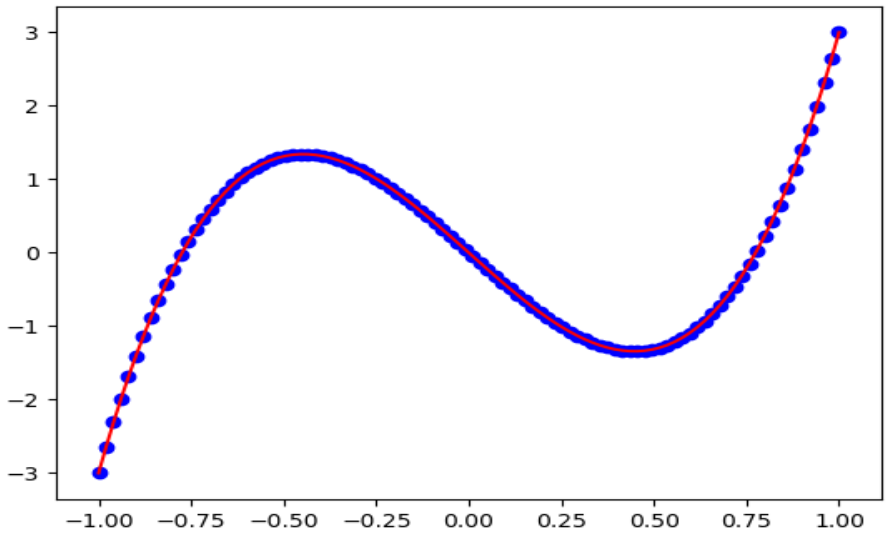
```
# Recurrence formulae for P_n(x)
#(1-x**2)P'_n(x)=[(n+1)xP_n(x)-(n+1)P_(n+1)(x)

import numpy as np
from scipy.special import legendre as P
import matplotlib.pyplot as plt

print 'Enter the order of Polynimial'
n=input()
#n=2  # Values of n
tol=0.0001 # tolerence
flag=0

lhs=np.poly1d([-1,0,1])*np.polyder(P(n))
rhs=(n+1.)*(np.poly1d([1,0])*P(n)-P(n+1))

X=np.linspace(-1,1,100) # range of integration
for x in X:
        if abs(lhs(x)-rhs(x))<tol:
                flag=1
        else:
                flag=0
if flag==1:
        print "Given recurrence fromulae is verified"
else:
        print "Given formulae is NOT verified"
```

```
plt.plot(X,lhs(X),c='red')
plt.scatter(X[::],rhs(X)[::],c='blue',marker='o')
plt.show()
```



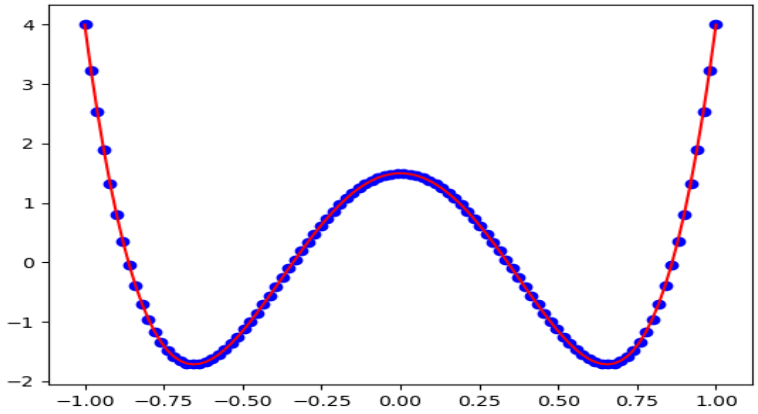
```
# Recurrence formulae for P_n(x)
#(n+1)P_(n+1)(x)=(2n+1)xP_n(x)-nP_(n-1)(x)

import numpy as np
from scipy.special import legendre as P
import matplotlib.pyplot as plt

print 'Enter the order of Polynimial'
n=input()
#n=2  # Values of n
tol=0.0001 # tolerence
flag=0

lhs=(n+1.)*P(n+1)
rhs=(2*n+1.)*np.poly1d([1,0])*P(n)-n*P(n-1)

X=np.linspace(-1,1,100) # range of x
for x in X:
        if abs(lhs(x)-rhs(x))<tol:
                flag=1
        else:
                flag=0
if flag==1:
        print "Given recurrence fromulae is verified"
else:
```

```
        print "Given formulae is NOT verified"

plt.plot(X,lhs(X),c='red')
plt.scatter(X[::],rhs(X)[::],c='blue',marker='o')
plt.show()
```



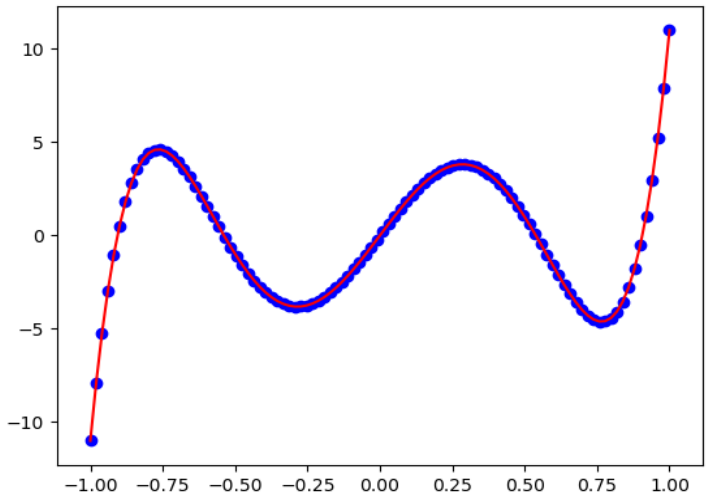
```
# Recurrence formulae for P_n(x)
#nP_n(x)=xP'_n(x)-P'_(n-1)(x)

import numpy as np
from scipy.special import legendre as P
import matplotlib.pyplot as plt

print 'Enter the order of Polynimial'
n=input()
#n=2  # Values of n
tol=0.0001 # tolerence
flag=0

lhs=n*P(n)
rhs=np.poly1d([1,0])*np.polyder(P(n))-np.polyder(P(n-1))

X=np.linspace(-1,1,100) # range of integration
for x in X:
        if abs(lhs(x)-rhs(x))<tol:
                flag=1
        else:
                flag=0
if flag==1:
        print "Given recurrence fromulae is verified"
```

```
else:
        print "Given formulae is NOT verified"

plt.plot(X,lhs(X),c='red')
plt.scatter(X[::],rhs(X)[::],c='blue',marker='o')
plt.show()
```



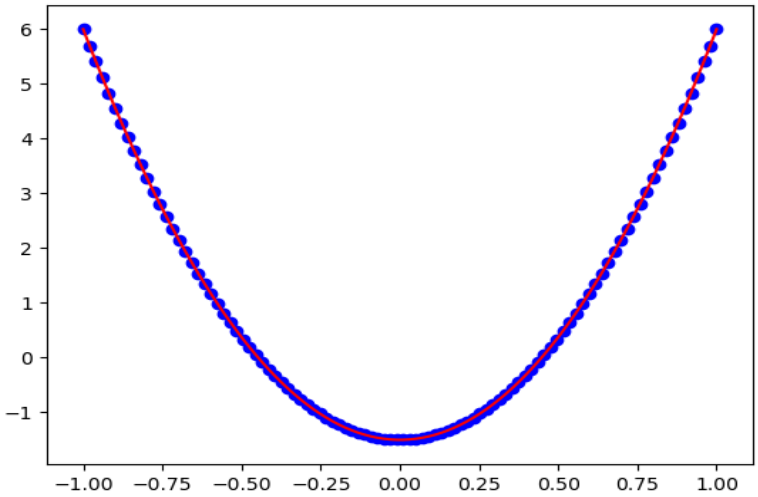
```
# Recurrence formulae for P_n(x)
#(2n+1)P_n(x)=P'_(n+1)(x)-P'_(n-1)(x)

import numpy as np
from scipy.special import legendre as P
import matplotlib.pyplot as plt

print 'Enter the order of Polynimial'
n=input()
#n=2  # Values of n
tol=0.0001 # tolerence
flag=0

lhs=(2*n+1)*P(n)
rhs=np.polyder(P(n+1))-np.polyder(P(n-1))

X=np.linspace(-1,1,100) # range ofx
for x in X:
        if abs(lhs(x)-rhs(x))<tol:
                flag=1
        else:
                flag=0
if flag==1:
        print "Given recurrence fromulae is verified"
else:
        print "Given formulae is NOT verified"
```

```
plt.plot(X,lhs(X),c='red')
plt.scatter(X[::],rhs(X)[::],c='blue',marker='o')
plt.show()
```

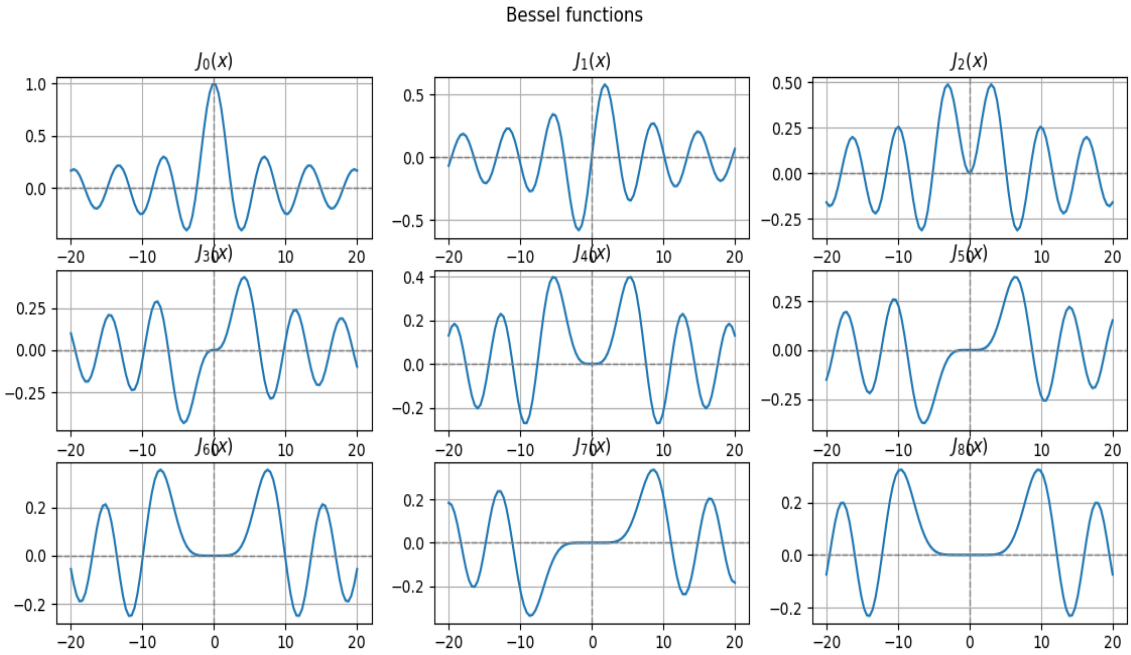


```
# Recurrence formulae for P_n(x)
#P'_n(x)=xP'_(n-1)(x)+nP_(n-1)(x)

import numpy as np
from scipy.special import legendre as P
import matplotlib.pyplot as plt

print 'Enter the order of Polynimial'
n=input()
#n=2  # Values of n
tol=0.0001 # tolerence
flag=0

lhs=np.polyder(P(n))
rhs=np.poly1d([1,0])*np.polyder(P(n-1))+n*P(n-1)

X=np.linspace(-1,1,100) # range of x
for x in X:
        if abs(lhs(x)-rhs(x))<tol:
                flag=1
        else:
                flag=0
if flag==1:
        print "Given recurrence fromulae is verified"
else:
        print "Given formulae is NOT verified"
```

```
plt.plot(X,lhs(X),c='red')
plt.scatter(X[::],rhs(X)[::],c='blue',marker='o')
plt.show()
```

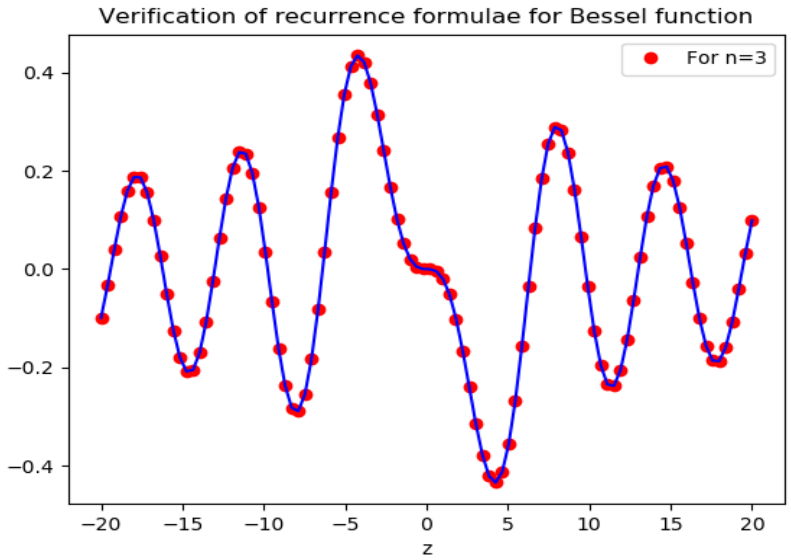


```
# Plotting Bessel functions of first kind


import numpy as np
from scipy.special import jv, jvp
import matplotlib.pyplot as plt

x=np.linspace(-20,20,100) # range of x

n=-3  # highest order of Polynomial
a,b=3,3 # number of plots along x and y axes respectively
        # row=b, column=a

plt.figure(figsize=(6*a,4*b))
plt.suptitle('Bessel functions')
for i in range(0,a*b):
        ax=plt.subplot(b,a,i+1)
        ax.plot(x,jv(i,x),label=r'$J_%d(x)$' %(i))
        ax.set_title('$J_%d(x)$'%(i))
        ax.axhline(0,c='gray',ls='--',lw=1)
        ax.axvline(0,c='gray',ls='--',lw=1)
        ax.grid()
plt.show()
```



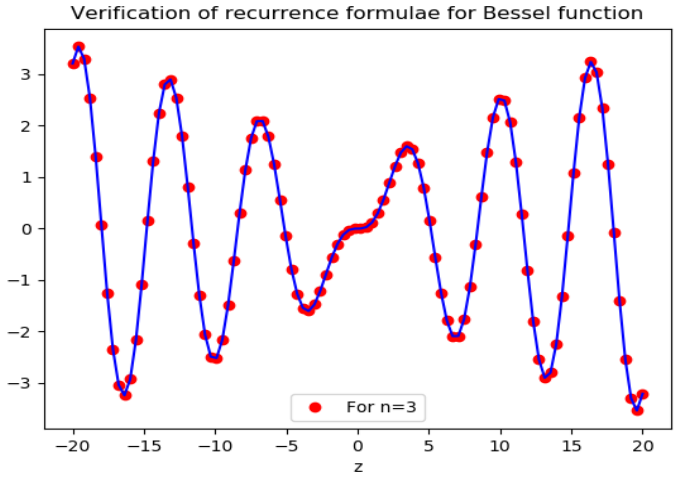
```
# Verification of J_-n(z)=(-1)^nJ_n(z)

import numpy as np
from scipy.special import jv, jvp
import matplotlib.pyplot as plt

z=np.linspace(-20,20,100) # range of x values

n=3  # value of n
tol=0.0001


lhs=jv(-n,z)
rhs=(-1.)**n*jv(n,z)

if abs(lhs-rhs).max()<tol:
        flag=1
else:
        flag=0
if flag==1:
        print "Given recurrence fromulae is verified"
else:
        print "Given formulae is NOT verified"

plt.plot(z,lhs,'ro',markevery=1,label='For n=%d'%(n))
```

```
plt.plot(z,rhs,'b-')
plt.title('Verification of recurrence formulae for Bessel function')
plt.xlabel('z')
plt.legend()
plt.show()
```



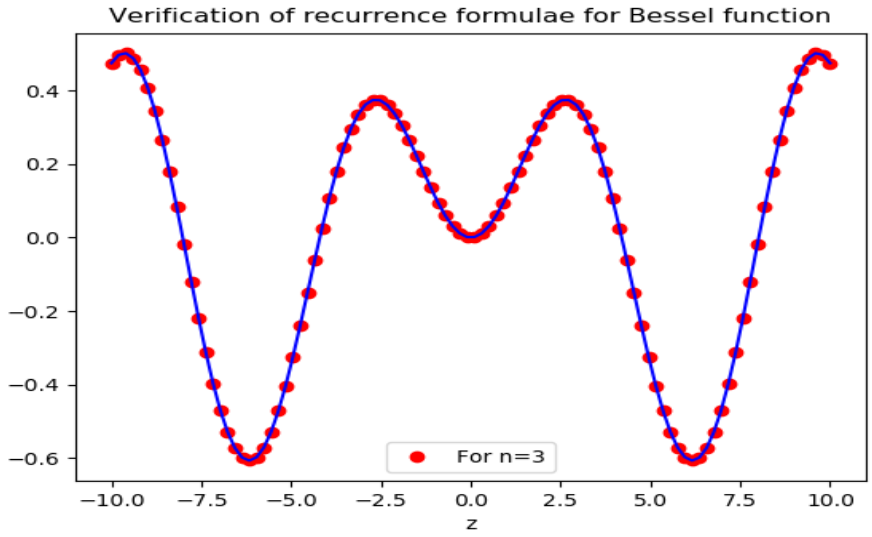
```
# Verifying recurrence formulae for Bessel functions
# zJ'_n(z)+nJ_n(z)=zJ_(n-1)(z)

import numpy as np
from scipy.special import jv, jvp
import matplotlib.pyplot as plt

z=np.linspace(-20,20,100) # range of z

n=3  # value of n
tol=1.e-4
lhs=z*jvp(n,z)+n*jv(n,z)
rhs=z*jv(n-1,z)

if abs(lhs-rhs).max()<tol:
        flag=1
else:
        flag=0
if flag==1:
        print "Given recurrence fromulae is verified"
else:
        print "Given formulae is NOT verified"
```

```
plt.plot(z,lhs,'ro',markevery=1,label='For n=%d'%(n))
plt.plot(z,rhs,'b-')
plt.title('Verification of recurrence formulae for Bessel function')
plt.xlabel('z')
plt.legend()
plt.show()
```



```
# Verifying recurrence formulae for Bessel functions
# J'n(x)=0.5*[J_(n-1)(x)-J_(n+1)(x)

import numpy as np
from scipy.special import jv, jvp
import matplotlib.pyplot as plt

z=np.linspace(-10,10,100) # range of z

n=3  # value of n
tol=1.e-4
lhs=2.*jvp(n,z)
rhs=(jv(n-1,z)-jv(n+1,z))

if abs(lhs-rhs).max()<tol:
        flag=1
else:
        flag=0
if flag==1:
        print "Given recurrence fromulae is verified"
else:
        print "Given formulae is NOT verified"


plt.plot(z,lhs,'ro',markevery=1,label='For n=%d'%(n))
```

```
plt.plot(z,rhs,'b-')
plt.title('Verification of recurrence formulae for Bessel function')
plt.xlabel('z')
plt.legend()
plt.show()
```



```
# Orthogonality of Bessel functions
# Integrate(xJn(ax)Jn(bx)dx, 0,1)=0 for a!=b, else 0.5*(J_n+1(a))**2

import numpy as np
from scipy.special import jv, jvp,jn_zeros
import matplotlib.pyplot as plt
from scipy.integrate import simps
import random


n=input('Enter the order of Bessel function\n')
nt=input('Enter the total number of zeros(root)\n')

a=jn_zeros(n,nt) # Calculation of zeros of the Jn(x)=0

x=np.linspace(0,1,100) # limit of integration

print '-------------------------------------------------------'
print '  a=\t\t\t','  b=\t\t','  \tI='
print '-------------------------------------------------------'
i=random.choice(a)
for j in a:
        fun=x*jv(n,i*x)*jv(n,j*x)
```

```
        rhs=0.5*(jv(n+1,i))**2
        I=simps(fun,x)/rhs
        print'%f\t\t %f\t\t %f'%(i,j,I)

print '------------------------------------------------------------'
```

OUTPUT

```
Enter the order of Bessel function
6
Enter the total number of zeros(root)
9
------------------------------------------------------------
  a=              b=            I=
------------------------------------------------------------
20.320789       9.936110       -0.000045
20.320789       13.589290           0.000056
20.320789       17.003820           -0.000064
20.320789       20.320789           1.000070
20.320789       23.586084           -0.000076
20.320789       26.820152           0.000081
20.320789       30.033722           -0.000085
20.320789       33.233042           0.000090
20.320789       36.422020           -0.000094
------------------------------------------------------------
```