Mathematical Physics - III (Practical)
Paper: PHS-A-CC-4-8-P
SPECIAL FUNCTIONS

'A special function is a function having a particular use in mathematical physics or some other branch of mathematics'. There are several special functions that recur in many branches in physics.

1. Legendre Polynomials:

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

Legendre Polynomials of different orders that satisfy the above different differential equation,

$$P_0(x) = 1,$$

$$P_1(x) = x = 1.x + 0,$$

$$P_2(x) = \frac{1}{2}(3x^2 - 1) = \frac{3}{2}x^2 + 0.x - \frac{1}{2},$$

$$P_3(x) = \frac{1}{2}(5x^3 - 3x) = \frac{5}{2}x^3 + 0.x^2 - \frac{3}{2}x + 0$$

…….
To evaluate the Legendre Polynomial (different order) for a range of values and plot we can run the following script:

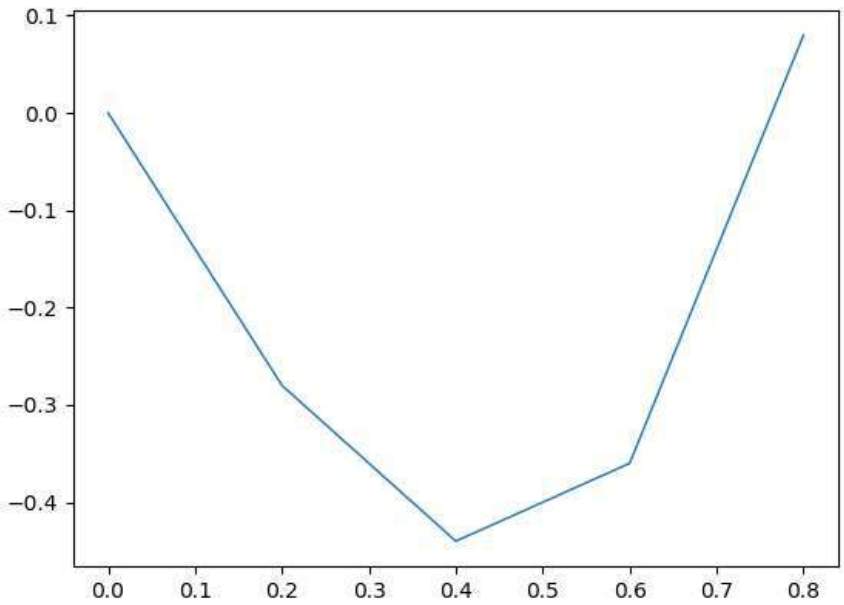
(a)

```
""" Special functions:LEGENGRE POLYNOMIALS"""

import numpy as np
from scipy.special import legendre, hermite, jn, yn
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

print  ('~~~ LEGENDRE POLYNOMIAL 3rd ORDER~~~')
print ('P0(x)=', legendre(0), '\nP1(x)=', legendre(1) ) # P0(x)=1, P1(x)=x
print ('P2(x)=', legendre(2), '\nP3(x)=', legendre(3))  # P2(x)=(3x^2-1)/2, P3(x)=(5x^3-3x)/2
p3 = legendre(3); [a3, a2, a1, a0] = p3
print ('Coefficients in decreasing power: ', p3[0],p3[1],p3[2],p3[3])
# construct polynomial at point x : p3(x) = a3*x**3 + a2*x**2 + a1*x + a0
print ('Polynomial at x = 0.9 is : ', legendre(3)(0.9), 'or ', p3(0.9))
```

```
x = np.arange(0,1,0.2) # construct various points x
plt.figure(1)
plt.plot(x, p3(x), lw=1)
plt.show()
```

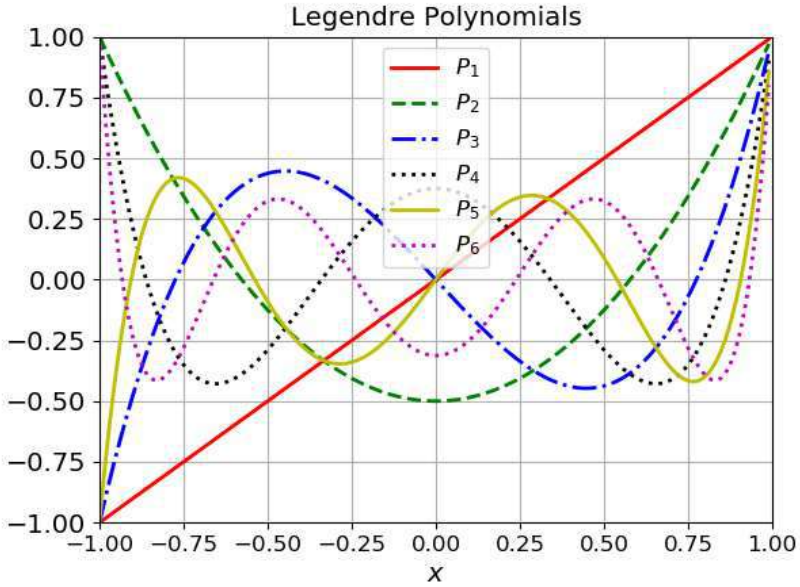
Python output as follows:

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Python3\semIV-CC8\sem4\poly-legengre-3rd.py ==========
~~~ LEGENDRE POLYNOMIAL 3rd ORDER~~~
P0(x)=
1
P1(x)=
1 x
P2(x)=      2
1.5 x - 0.5
P3(x)=      3
2.5 x - 1.5 x
Coefficients in decreasing power:  0.0 -1.5000000000000002 0.0 2.5
Polynomial at x = 0.9 is :  0.47250000000000014 or  0.47250000000000014
```

(b)

```
""" Special functions: Legendre polynomial (different order)"""
import numpy as np
from scipy.special import legendre, hermite, jn, yn
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

x = np.arange(-1,1,0.01)
p1 = legendre(1); p2 = legendre(2); p3 = legendre(3); p4 = legendre(4);
p5 = legendre(5); p6 = legendre(6);
plt.figure(1)
plt.plot(x, p1(x), lw=2, ls='-',  color='r', label=r'$P_1$')
plt.plot(x, p2(x), lw=2, ls='--', color='g', label=r'$P_2$')
plt.plot(x, p3(x), lw=2, ls='-.', color='b', label=r'$P_3$')
plt.plot(x, p4(x), lw=2, ls=':',  color='k', label=r'$P_4$')
plt.plot(x, p5(x), lw=2, ls='-',  color='y', label=r'$P_5$')
plt.plot(x, p6(x), lw=2, ls=':',  color='m', label=r'$P_6$')
plt.legend(loc='best',prop={'size':12})
plt.grid()
plt.axis([-1, 1, -1, 1])
plt.title('Legendre Polynomials', fontsize = 14)
plt.xlabel('$x$', fontsize = 14)
plt.xticks(fontsize = 12)
plt.ylabel(r'$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2-1)^n$',fontsize = 14)
plt.yticks(fontsize = 14)
plt.show()
#plt.savefig('FIG1.pdf')
```

(C)
Orthogonal Property:

$\int_{-1}^{1} P_n(x)\, P_m(x) = 0$ for $m \neq n$

$\int_{-1}^{1} [P_n(x)]^2\, dx = \frac{2}{2n+1}$ for $m = n$

Recursion relations:

(i) $nP_n(x) = (2n-1)x\, P_{n-1}(x) - (n-1)P_{n-2}(x)$

(ii) $(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x)$

(iii) $(1-x^2)P_n'(x) = n\big(P_{n-1}(x) - xP_n(x)\big)$

(iv) $nP_n(x) = xP_n'(x) - P_{n-1}'(x)$

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

""" Orthonormality and Recursion relation for Legendre functions"""


```
import numpy as np
from scipy.special import legendre
from scipy.misc import derivative
import scipy.integrate as sci
import warnings
warnings.filterwarnings("ignore")

# Feed the value of n, m, start, stop, Np from keyboard
n = 4; m = 5; start = -1; stop = 1; Np = 1e6;
x = np.linspace(start, stop, Np);

# Create Poly1D Legendre polynomial and derivatives
pn = legendre(n); pm = legendre(m); pnm1 = legendre(n-1);
pnm2 = legendre(n-2); pnp1 = legendre(n+1);
pnprime = derivative(pn, x, 1e-6)  # spacing=10^-6
pnm1prime = derivative(pnm1, x, 1e-6)

# Logical case switch for different recursions to choose from
recl1=1; recl2=1; recl3=1; recl4=1; recl5=1; recl6=1
print ('Compare maximum of |lhs-rhs| (L1 norm) to zero for n = ', n)

if(recl1): # \int Pn(x)Pm(x) = 2/(2n+1)*delta(nm)
  I = sci.simps(pn(x)*pm(x),x)*(2.0*n+1)/2.0
```

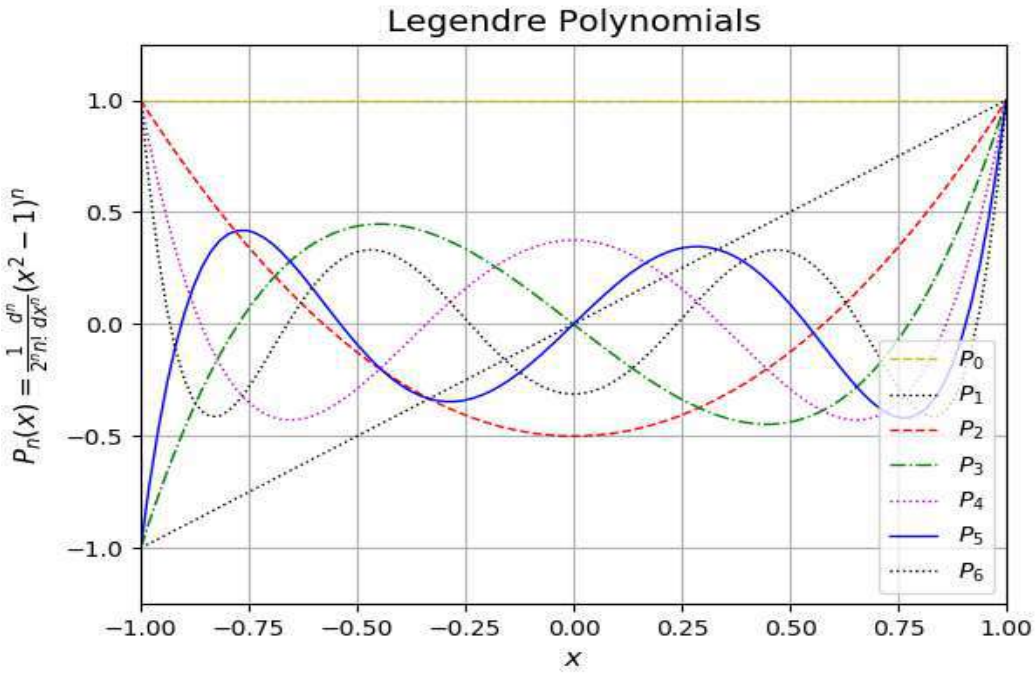
```
  print ('Orthonormality : \int P_',n,'(x)P_',m,'(x)dx = ', I)

if(recl2): #nPn(x) = (2n-1)xP(n-1)(x) - (n-1)P(n-2)(x)
  lhs = n*pn(x)
  rhs = (2*n-1)*x*pnm1(x)-(n-1)*pnm2(x)
  print ('Maximum of nPn(x)-(2n-1)xPn(x)+(n-1)P(n-2)(x) = ', abs(max(lhs-rhs)))

if(recl3): #(n+1)P(n+1)(x) = (2n+1)xPn(x) - nP(n-1)(x)
  lhs = (n+1)*pnp1(x)
  rhs = (2*n+1)*x*pn(x)-n*pnm1(x)
  print ('Maximum of (n+1)P(n+1)(x)-(2n+1)xPn(x)+nP(n-1)(x) = ', abs(max(lhs-rhs)))

if(recl4): #(1-x^2)Pn'(x) = n(P(n-1)(x) - xPn(x))
  lhs = (1-x**2)*pnprime
  rhs = n*(pnm1(x)-x*pn(x))
  print ('Maximum of (1-x^2)dPn(x)/dx-n[P(n-1)(x)-xPn(x)] = ', abs(max(lhs-rhs)))

if(recl5): #nPn(x) = xPn'(x) - P(n-1)'(x)
  lhs = n*pn(x)
  rhs = x*pnprime-pnm1prime
  print ('Maximum of nPn(x)-xdPn(x)/dx+dP(n-1)(x)/dx = ', abs(max(lhs-rhs)))

if(recl6): #Pn'(x) = xP(n-1)'(x) + nP(n-1)(x)
  lhs = pnprime
  rhs = x*pnm1prime + n*pnm1(x)
  print ('Maximum of dPn(x)/dx-xdP(n-1)(x)/dx-nP(n-1)(x) = ', abs(max(lhs-rhs)))
```


Plotting:

(d)

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.integrate import quad
from scipy.special import jv

def legendrerecur(n):
    if (n == 0):
        Pn = [1.]
    elif (n == 1):
        Pn = [1.,0.]
    else:
        Pn_2 = legendrerecur(0)
        Pn_1 = legendrerecur(1)
        for i in range(2,n+1):
            Pn =  ((2*i - 1)*np.append(Pn_1,0) - (i-1)*np.insert(Pn_2,0,[0,0]))/i
            Pn_2 = Pn_1
            Pn_1 = Pn
```

```
    P = np.poly1d(Pn) # Convert array into poly1D object to allow natural operations with polynomials
    return P

print (legendrerecur(0),legendrerecur(1), legendrerecur(2) )

lb=-1; ub=1; x = np.linspace(lb,ub,100)
plt.figure()
plt.plot(x, legendrerecur(0)(x), lw=1, ls='--', color='y',  label=r'$P_0$')
plt.plot(x, legendrerecur(1)(x), lw=1, ls=':',  color='k',  label=r'$P_1$')
plt.plot(x, legendrerecur(2)(x), lw=1, ls='--', color='r',  label=r'$P_2$')
plt.plot(x, legendrerecur(3)(x), lw=1, ls='-.', color='g',  label=r'$P_3$')
plt.plot(x, legendrerecur(4)(x), lw=1, ls=':',  color='m',  label=r'$P_4$')
plt.plot(x, legendrerecur(5)(x), lw=1, ls='-',  color='b',  label=r'$P_5$')
plt.plot(x, legendrerecur(6)(x), lw=1, ls=':',  color='k',  label=r'$P_6$')
plt.legend(loc='best',prop={'size':10})
plt.grid()
plt.axis([lb, ub, -1.25, 1.25])
plt.title('Legendre Polynomials', fontsize = 14)
plt.xlabel('$x$', fontsize = 12)
plt.ylabel(r'$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2-1)^n$',fontsize = 12)
plt.xticks(fontsize = 10)
plt.yticks(fontsize = 10)
plt.show()
```

(e)
BESSEL FUNCTIONS:
Bessel functions are the solutions of the following second order differential equation,

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

Bessel functions of the first kind,

$$J_n(x) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m!\ \Gamma(m+n+1)}\left(\frac{x}{2}\right)^{2m+n}$$

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

Bessel functions of the second kind,

$$Y_n(x) = \frac{J_n(x)\cos n\pi - J_{-n}(x)}{\sin n}$$

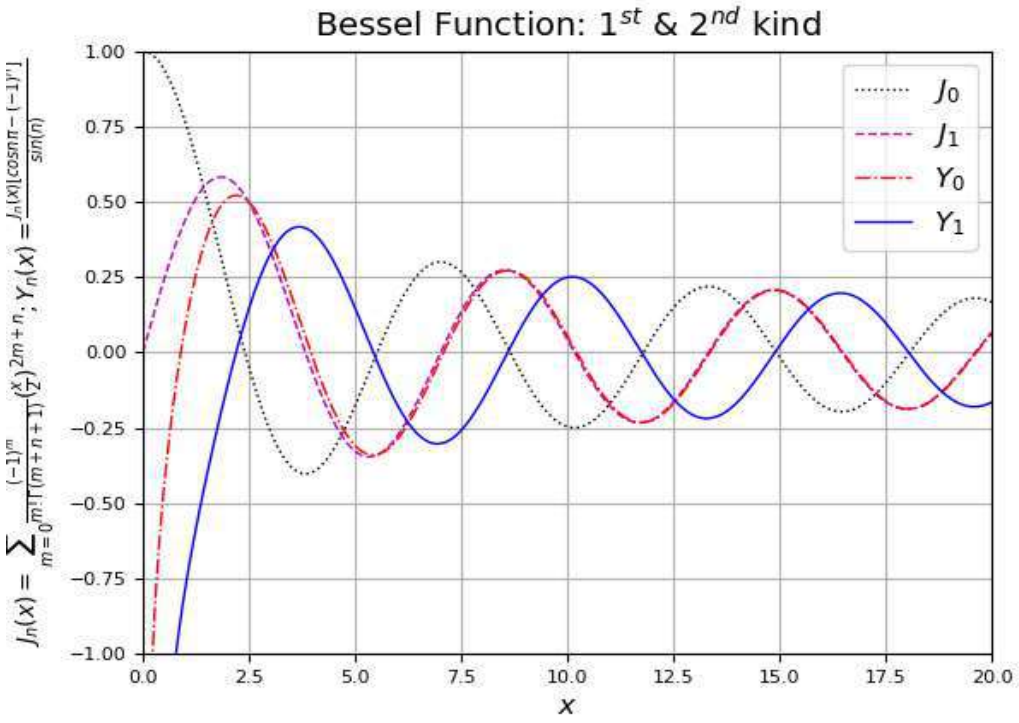
""" Special functions: BESSEL FUNCTIONS"""


```
import numpy as np
from scipy.special import legendre, hermite, jn, yn
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")


#=================================================================#
print     ('~~~ BESSEL FUNCTION ~~~')                    #
#=================================================================#
print ('J0(1)=', jn(0,1), '\nJ1(5 )=', jn(1,5))  # First Kind
print ('Y0(1)=', yn(0,1), '\nY0(30)=', yn(0,30)) # Second Kind
x = np.linspace(0,20,300)
j0 = jn(0,x); j1 = jn(1,x); y0 = yn(0,x); y1 = yn(1,x);
plt.figure()
plt.plot(x, j0, lw=1, ls=':',  color='k', label=r'$J_0$')
plt.plot(x, j1, lw=1, ls='--', color='m', label=r'$J_1$')
plt.plot(x, y0, lw=1, ls='-.', color='r', label=r'$Y_0$')
plt.plot(x, y1, lw=1, ls='-',  color='b', label=r'$Y_1$')
plt.legend(loc='best',prop={'size':12})
plt.grid()
plt.axis([0, 20, -1, 1])
plt.title('Bessel Function: $1^{st}$ & $2^{nd}$ kind', fontsize = 14)
```

```
plt.xlabel('$x$', fontsize = 12)
plt.xticks(fontsize = 8)
plt.ylabel(r'$J_n(x) = \sum_{m=0}^\infty \frac{(-
1)^m}{m!\Gamma(m+n+1)}(\frac{x}{2})^{2m+n}; Y_n(x) = \frac{J_n(x)[cos n\pi - (-
1)^n]}{sin(n)}$',fontsize = 10)
plt.yticks(fontsize = 8)
plt.savefig('FIG-bessel.pdf')
plt.show()
```

Output as follows:

```
============= RESTART: C:\Python3\semIV-CC8\sem4\poly-Bessel.py =============
~~~ BESSEL FUNCTION ~~~
J0(1)= 0.7651976865579666
J1(5 )= -0.3275791375914651
Y0(1)= 0.08825696421567697
Y0(30)= -0.11729573168666398
>>>
```

(f) Hermite Polynomial

Hermite differential equation:

$\frac{d^2y}{dx^2} - 2x\frac{dy}{dx} + 2\lambda y = 0,$ where λ is a constant.

Differntial form of Hermite Polynomials:

$H_n(x) = (-1)^n\, e^{x^2}\, \frac{d^n}{dx^n}\left(e^{-x^2}\right)$

Therefore,

$$H_0 = 1$$

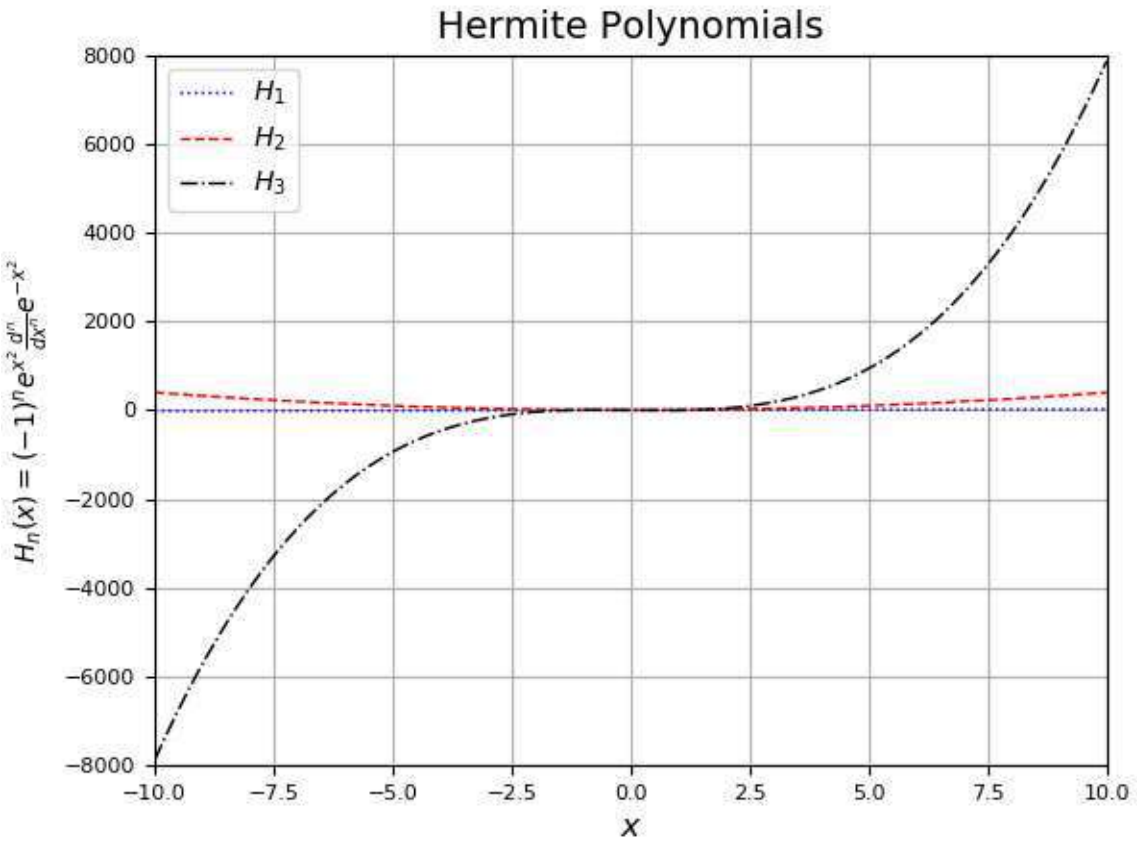
$$H_{1(x)} = 2x$$

$$H_2(x) = 4x^2 - 2$$

$$H_3(x) = 8x^3 - 12x$$


```
""" Special functions:HERMITE POLYNOMIAL"""

import numpy as np
from scipy.special import legendre, hermite, jn, yn
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

#================================================================#
print     ('~~~ HERMITE POLYNOMIAL ~~~'  )                      #
#================================================================#
print ('H0(x)=', hermite(0), '\nH1(x)=', hermite(1))  # H0(x)=1, H1(x)=2*x
print ('H2(x)=', hermite(2), '\nH3(x)=', hermite(3) ) # H2(x)=4x^2-2, H3(x)=8x^3-12x
x = np.arange(-10,10,0.01)
h1 = hermite(1); h2 = hermite(2); h3 = hermite(3);
plt.figure()
plt.plot(x, h1(x), lw=1, ls=':',  color='b', label=r'$H_1$')
plt.plot(x, h2(x), lw=1, ls='--', color='r', label=r'$H_2$')
plt.plot(x, h3(x), lw=1, ls='-.', color='k', label=r'$H_3$')
plt.legend(loc='best',prop={'size':10})
plt.grid()
```

```
plt.axis([-10, 10, -8000, 8000])
plt.title('Hermite Polynomials', fontsize = 14)
plt.xlabel('$x$', fontsize = 12)
plt.xticks(fontsize = 8)
plt.ylabel(r'$H_n(x) = (-1)^ne^{x^2}\frac{d^n}{dx^n}e^{-x^2}$',fontsize = 10)
plt.yticks(fontsize = 8)
plt.savefig('FIG.pdf')
plt.show()
```



(g) Recursion relations for Bessel functions

(i) $\frac{2n}{x} J_n(x) = J_{n+1}(x) + J_{n-1}(x)$

```
"""Recursion relations for Bessel functions
"""

import numpy as np
from scipy.special import jn, yn, jvp
from scipy.misc import derivative
```

```
import warnings
warnings.filterwarnings("ignore")

n = 5; start = 1e-2; stop = 10; Np = 1000;
x = np.linspace(start, stop, Np);

# (2*n/x)*J(n)(x) = J(n+1)(x)+J(n-1)(x)
lhs = np.divide(2*n*jn(n,x), x)
rhs = jn(n+1,x) + jn(n-1,x)
print ('Maximum of (2*n/x)*J(n)(x)-J(n+1)(x)+J(n-1)(x) = ', abs(max(lhs-rhs)))
```

Assignments: Show that (I). $P_n(-x) = (-1)^n P_n(x)$

(II). $P_n(1) = 1$

(III) $xJ_n'(x) + nJ_n(x) = xJ_{n-1}(x)$