勒让德

```python
import sympy as sym

# sympy 符号计算

x=sym.Symbol("x")    #声明变量，就是那个自变量x

f=sym.legendre(0,x)    #勒让德多项式 P0(x)  那个legendre就相当于把那一整串符号整合成函数，你输入输入值，就会得到输出值
print(f)               #显示输出值
sym.plot(f,(x,-1,1))   #作图
f=sym.legendre(1,x) #勒让德多项式 P1(x)
print(f)
sym.plot(f,(x,-1,1))

f=sym.legendre(2,x) #勒让德多项式 P2(x)
print(f)
sym.plot(f,(x,-1,1))

f=sym.legendre(3,x) #勒让德多项式 P3(x)
print(f)
sym.plot(f,(x,-1,1))

f=sym.legendre(4,x) #勒让德多项式 P4(x)
print(f)
sym.plot(f,(x,-1,1))

f=sym.legendre(5,x) #勒让德多项式 P5(x)
print(f)
sym.plot(f,(x,-1,1))

f=sym.legendre(6,x) #勒让德多项式 P6(x)
print(f)
sym.plot(f,(x,-1,1))
```

```python
# 求多项式函数f(x)=2*x**3+3*x+4 的勒让德多项式展开

ff=(2*x**3+3*x+4)*sym.legendre(0,x)    # f(x)*P0(x)
ff0=1/2*sym.Integral(ff, (x, -1,1))    # C0=f(x)*P0(x)在(-1,1)积分

ff=(2*x**3+3*x+4)*sym.legendre(1,x)    # f(x)*P1(x)
ff1=3/2*sym.Integral(ff, (x, -1,1))    # C1=f(x)*P1(x)在(-1,1)积分

ff=(2*x**3+3*x+4)*sym.legendre(2,x)    # f(x)*P2(x)
ff2=5/2*sym.Integral(ff, (x, -1,1))    # C2=f(x)*P2(x)在(-1,1)积分

ff=(2*x**3+3*x+4)*sym.legendre(3,x)    # f(x)*P3(x)
ff3=7/2*sym.Integral(ff, (x, -1,1))    # C3=f(x)*P3(x)在(-1,1)积分

C0=ff0.doit()
C1=ff1.doit()
C2=ff2.doit()
C3=ff3.doit()
```

$$C_n = \frac{2n+1}{2}\int_{-1}^{1} f(x)P_n(x)dx$$

用的是这个函数

3

```python
import scipy as scp       # 数值计算
import numpy as np        # 数据
from scipy import integrate
import matplotlib.pyplot as plt #画图
from numpy.polynomial import Polynomial, Legendre

scp.special.legendre(2)    #构造勒让德多项式P2(x)
scp.special.legendre(2)(np.linspace(-1,1,50)) #计算P2(x) (-1,1)

for i in range(6):
    plt.plot(scp.special.legendre(i)(np.linspace(-1,1,50)))

fx=scp.poly1d([2,0,3,1])                  #生成多项式函数f(x)=2*x**3+3*x+4
ffa0=fx*scp.special.legendre(0)           #f(x)*P0(x)
C0,err=scp.integrate.quad(ffa0,-1,1)      #C0=f(x)*P0(x)在(-1,1)积分,eer为误差
C0=1/2*C0                                 #C0

ffa1=fx*scp.special.legendre(1)           #f(x)*P1(x)
C1,err=scp.integrate.quad(ffa1,-1,1)      #C1=f(x)*P1(x)在(-1,1)积分,eer为误差
C1=3/2*C1                                 #C0

ffa2=fx*scp.special.legendre(2)           #f(x)*P2(x)
C2,err=scp.integrate.quad(ffa2,-1,1)      #C2=f(x)*P2(x)在(-1,1)积分,eer为误差
C2=5/2*C2                                 #C2


ffa3=fx*scp.special.legendre(3)           #f(x)*P3(x)
C3,err=scp.integrate.quad(ffa3,-1,1)      #C3=f(x)*P3(x)在(-1,1)积分,eer为误差
C3=7/2*C3                                 #C3
```

```python
for i in range(5):
    p = Legendre.basis(i).convert(kind=Polynomial)    #用numpy
    print(p.coef)
#不转换成多项式，直接数值计算也可以
Legendre.basis(2)(np.linspace(-1,1,10))

plt.plot(*Legendre.basis(0).linspace(),label = 'P(0)')
plt.plot(*Legendre.basis(1).linspace())
plt.plot(*Legendre.basis(2).linspace())
plt.plot(*Legendre.basis(3).linspace(),label = 'P(3)')
plt.plot(*Legendre.basis(4).linspace())
plt.plot(*Legendre.basis(5).linspace())
plt.plot(*Legendre.basis(6).linspace())
plt.plot(*Legendre.basis(7).linspace())
plt.plot(*Legendre.basis(8).linspace())

plt.show()
```

创建一些数据，种类为多项式

P的系数，即多项式的系数

算具体的值，这里是P(2)

贝塞

```
import sympy
from sympy import besselj,jn,bessely,besselk,besseli
from sympy.abc import x, n

# 第一类贝塞尔函数 besselj(0, x)，第一个参数为阶，第二个为变量x
sympy.plot(besselj(0, x), besselj(1, x),besselj(2, x),besselj(3, x),(x, 0, 10))
sympy.plot(besselj(2, x), besselj(-2, x),(x, 0, 10),xlabel='J_-1(x),J_1(x)')      #p为正、负整数
sympy.plot(besselj(1.5, x), (x, 0, 1),xlabel='J_1.5(x)',xlim=[0,0.5], ylim=[-1,0.5]) #p为非整数，正数
sympy.plot(besselj(-1.5, x), (x, 0, 1),xlabel='J_-1.5(x)',xlim=[0,2], ylim=[-50,10]) #p为非整数，负数

# 第二类贝塞尔函数 bessely(0, x)，第一个参数为阶，第二个为变量x
sympy.plot(bessely(0, x),(x, 0, 20),xlabel='N_0(x)',xlim=[0,10], ylim=[-2,2]) #N_0(x)，N_1(x)，N_3(x)曲线
sympy.plot(bessely(1, x),(x, 0, 20),xlabel='N_1(x)',xlim=[0,10], ylim=[-2,2])
sympy.plot(bessely(2, x),(x, 0, 20),xlabel='N_2(x)',xlim=[0,10], ylim=[-2,2])


for i in range(6):
    b = besselj(i, x)
    sympy.plot(b, (x, 0, 20))

#sym.plot(x**2, (x, -2, 2))
for i in range(6):
    b = bessely(i, x)
    sympy.plot(b, (x, 0, 2335.2))
```

变量x范围(0,10)

标签而已

0~0.5
0~2

-1~0.5
-50~10

分别代表x,与y轴的显示范围

同理，也是范围

2

```
import numpy as np
from scipy.integrate import odeint
from scipy.special import jn # bessel function
import matplotlib.pyplot as plt

def fbessel(Y, x):
    nu = 0.0
    y = Y[0]
    x = Y[1]

    dydx = x
    dxdx = 1.0 / x**2 * (-x * x - (x**2 - nu**2) * y)
    return [dydx, dxdx]

x0 = 1e-15
y0 = 1
x0 = 0
Y0 = [y0, x0]

xspan = np.linspace(1e-15, 10)
sol = odeint(fbessel, Y0, xspan)
plt.plot(xspan, sol[:,0], label='numerical soln')
plt.plot(xspan, jn(0, xspan), 'r--', label='Bessel')
plt.legend()

x0 = 1e-15
y0 = 1
x0 = 0
Y0 = [y0, x0]

xspan = np.linspace(1e-15, 10)
sol = odeint(fbessel, Y0, xspan)

plt.plot(xspan, sol[:,1], label='numerical soln')
plt.plot(xspan, jn(-1, xspan), 'r--', label='Bessel')
plt.legend()
```

创建贝塞尔函数

分10份

数值求解微分方程

微分方程函数

微分方程初值

微分的跨度

一般贝塞尔的参数范围：0～xspan

可以理解为图层0.1

同一个画图可以有很多图层

3

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import *

X = np.random.random(3)
v = 2.000000000

print("Bessel Function J")
print (jn(v,X))

print("Modified Bessel Function, Iv")
print ((iv(v,X)))

print("Modified Bessel Function of the second kind, Kv")
print(kv(v,X))

n=0
x=np.linspace(0,10,100)
fig,ax=plt.subplots()
for n in range(4):
    ax.plot(x,jn(n,x),label=r"$J_%d(x)$"% n)
ax.legend()

n=0
m=4
jn_zeros(n,m)

x=np.linspace(0,10,100)
fig,ax=plt.subplots()
for n in range(4):
    ax.plot(x,yn(n,x),label=r"$Y_%d(x)$"% n)
ax.legend()
```

自变量

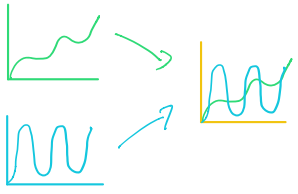
阶数

一般贝塞尔

第一类修正贝塞尔的两个特解

生成子图

这里应该是想把多张图画在一起



4
```
#解贝塞尔方程
x,p = sympy.symbols("x,p")
y = sympy.Function("y")
f=sympy.Eq( x*x* y(x).diff(x,2) + x* y(x).diff(x)+(x*x-p**2)*y(x),0)
print(f)
sympy.dsolve(f)


#解勒让德方程
x = sympy.symbols("x")
y = sympy.Function("y")
f=sympy.Eq((1-x*x)*y(x).diff(x,2)-2*x*y(x).diff(x)+3*4*y(x),0)
sympy.dsolve(f)

#双曲贝塞尔函数
x,p = sympy.symbols("x,p")
y = sympy.Function("y")
f=sympy.Eq( x*x* y(x).diff(x,2) + x* y(x).diff(x)-(x*x+p**2)*y(x),0)
sympy.dsolve(f)

#y'' +9xy=0
x = sympy.symbols("x")
y = sympy.Function("y")
f=sympy.Eq(y(x).diff(x,2)+9*x*y(x),0)
sympy.dsolve(f)
```