第三次作业

缔合拉盖尔多项式

$L_k^p(t) = (-1)^p \frac{d^p}{dt^p} L_{k+p}(t)$

已知

$L_n(x) = \frac{e^x}{n!} \frac{d}{dx^n}(x^n e^{-x})$

容易推出：$L_{n+1}(x) = (2n+1-x)L_n(x) - n^2 L_{n-1}(x)$

$L_0(x) = 1$

$L_1(x) = -x + 1$

$L_2(x) = x^2 - 4x + 2$

进行扩展

$L_k^p(t) = (-1)^p \frac{d^p}{dt^p} L_{k+p}(t)$

类比p=0,将L表达式代入，容易推出：

$L_{k+1}^p(x) = \frac{2k+1+p-x}{k+1} L_k^p(x) - \frac{p+k}{k+1} L_{k-1}^p(x)$

$L_0^p(x) = 1$

$L_1^p(x) = -x + p + 1$

$L_2^p(x) = \frac{1}{2}[x^2 - 2(k+2)x + (k+1)(k+2)]$

广义拉盖尔函数

```
import pandas as pd
import sympy as sy
def Laguerre(k,p):
    L0p=1
    L1p=-x+p+1
    L2p=1/2*(x**2-2*(p+2)*x+(p+1)*(p+2))
    if k==0:
        return L0p
    elif k==1:
        return L1p
    elif k==2:
        return L2p
    elif k>2:
        temp=(2*k-1-x)/k*Laguerre(k-1,p)-(k-1)/k*Laguerre(k-2,p)
        return temp
```

```
import pandas as pd
import sympy as sy
x= sy.symbols("x")
data_df = pd.DataFrame()
data_df['n'] = range(0,6)
data_df['l1'] = [0,0,0,1,0,1]
data_df['L(l+1/2,(n-l)/2)(x)0']=
[Laguerre(0,1/2),Laguerre(0,3/2),Laguerre(1,1/2),Laguerre(1,3/2),La
guerre(2,1/2),Laguerre(2,3/2)]
data_df['l2']=[None,None,2,3,2,3]
data_df['L(l+1/2,(n-l)/2)(x)1']=
[None,None,Laguerre(0,3/2),Laguerre(0,5/2),Laguerre(1,5/2),Laguerre
(1,7/2)]
data_df['l3']=[None,None,None,None,4,5]
data_df['L(l+1/2,(n-l)/2)(x)2']=
[None,None,None,None,Laguerre(0,9/2),Laguerre(0,11/2)]
data_df
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	N	L1	L(L+1/2,(N-L)/2)(X)0	L2	L(L+1/2,(N-L)/2)(X)1	L3	L(L+1/2,(N-L)/2)(X)2
0	0	0	1	NaN	None	NaN	NaN
1	1	0	1	NaN	None	NaN	NaN
2	2	0	1.5 - x	2.0	1	NaN	NaN
3	3	1	2.5 - x	3.0	1	NaN	NaN
4	4	0	0.5*x**2 - 2.5*x + 1.875	2.0	3.5 - x	4.0	1.0
5	5	1	0.5*x**2 - 3.5*x + 4.375	3.0	4.5 - x	5.0	1.0

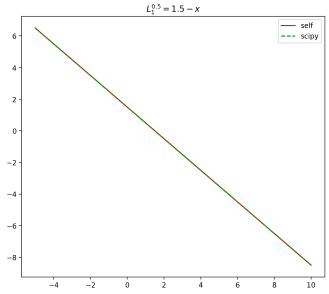
与table4-5进行对比，结果一模一样，没有问题
下面进行画图分析

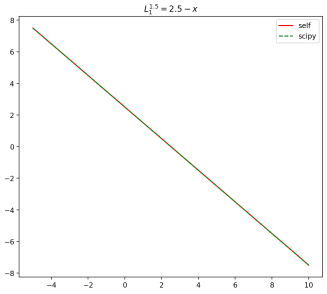
```
import sympy as sy
x=sy.symbols('x')
def Laguerre_shuzhi(x,k,p):#数值运算
    L0p=1
    L1p=-x+p+1
    L2p=1/2*(x**2-2*(p+2)*x+(p+1)*(p+2))
    if k==0:
        return L0p
    elif k==1:
        return L1p
    elif k==2:
        return L2p
    elif k>2:
        temp=(2*k-1+p-x)/k*Laguerre_shuzhi(x,k-1,p)-(k-1+p)/k*Laguerre_shuzhi(x,k-2,p)
        return temp
```

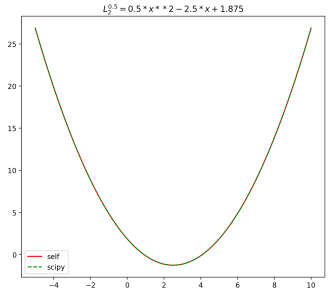
画图分析 与**scipy**进行比较分析

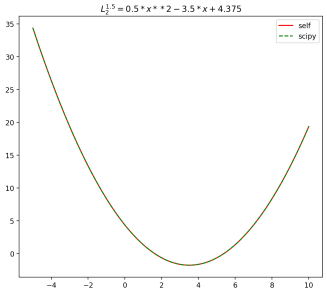
```
import matplotlib.pyplot as plt
import numpy as np
import scipy.special as special

plt.figure(figsize=(25,40),dpi=200)
klist=[1,1,2,2,1,1,3,4,4,4]
plist=[1/2,3/2,1/2,3/2,5/2,7/2,1/2,3/2,5/2,9/2]
for i in range(1,11):
    plt.subplot(5,2,i)
    k=klist[i-1]
    p=plist[i-1]
    title=Laguerre(k,p)
    plt.title('$L_{}^{}={}$'.format(k,{p},title))
    plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=1.2,
hspace=None)
    ff=np.linspace(-5,10,200)
```

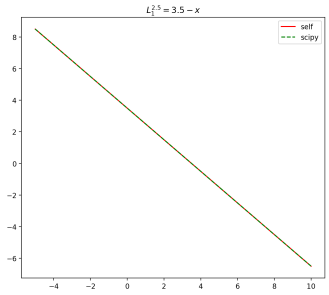
```
    ylist=[]
    ytrue=[]
    for j in range(0,len(ff)):
        temp1=ff[j]
        yy1=Laguerre_shuzhi(temp1,k,p)
        yy2=special.assoc_laguerre(temp1,k,p)
        ylist.append(yy1)
        ytrue.append(yy2)
    plt.plot(ff,ylist,'r',label='self')
    plt.plot(ff,ytrue,'g',label='scipy',linestyle='dashed')
    plt.legend()
```

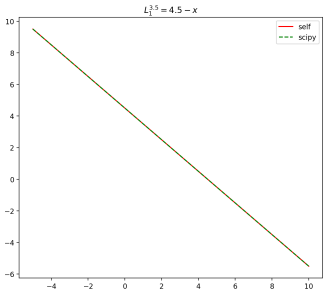


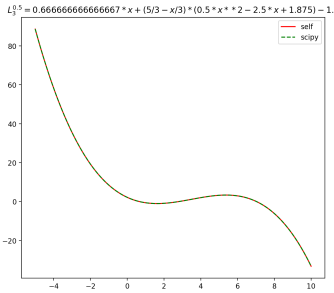


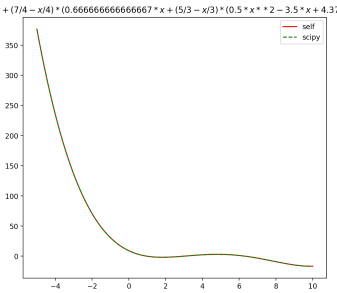


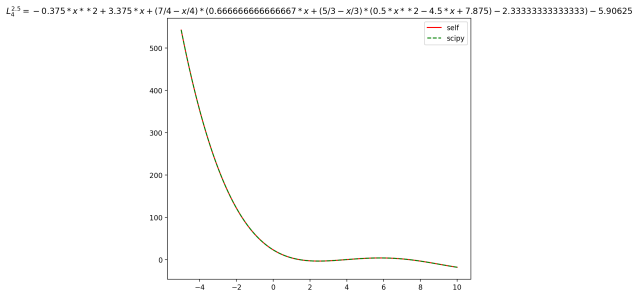


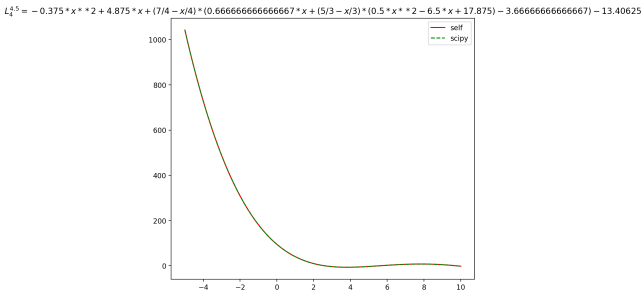












由绘制的图像可知，自己编写的拉盖尔多项式与sci库几乎完全重合

氢原子相关

预备工作，球谐函数与关联勒让德函数的编写

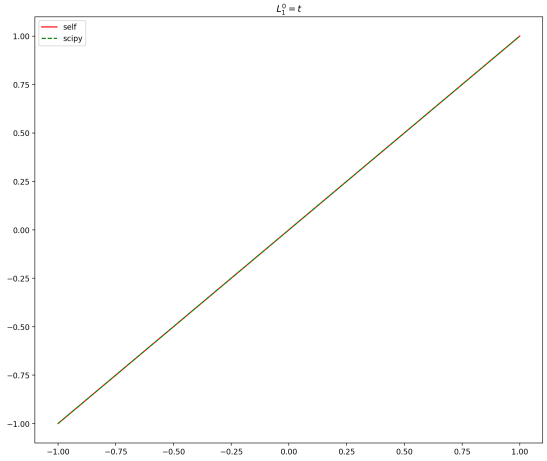
关联勒让德函数

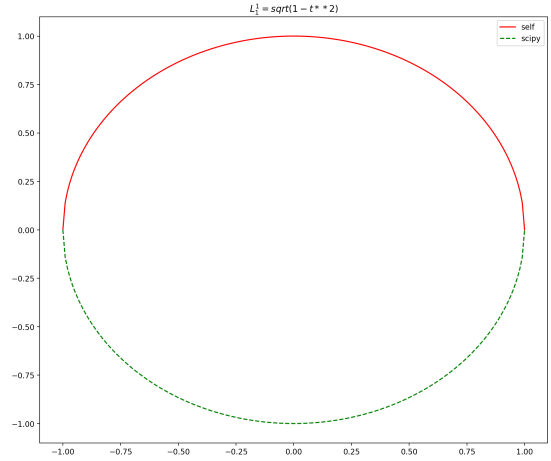
```
import pandas as pd
import sympy as sy
t=sy.symbols("t")
def Legendre(t,l,m):
    if m==0:
        L0m=1
        L1m=t
        L2m=1/2*(3*t**2-1)
    elif abs(m)==1:
        L0m=0
        L1m=sy.sqrt(1-t**2)
        L2m=3*sy.sqrt(1-t**2)*t
    elif abs(m)==2:
        L0m=0
        L1m=0
        L2m=3*(1-t**2)
    else:
        L0m=0
        L1m=0
        L2m=0
    if l==0:
        return L0m
    elif l==1:
        return L1m
    elif l==2:
        return L2m
    elif l>2 and (l-m)!=0:
        temp=(2*l-1)*t/(l-m)*Legendre(t,l-1,m)-(l-1+m)/(l-m)*Legendre(t,l-2,m)
        return temp
    elif l>2 and (l-m)==0:
        temp=Legendre(t,l-2,m)+(2*l-1)*sy.sqrt(1-t**2)*Legendre(t,l-1,m-1)
        return temp
```

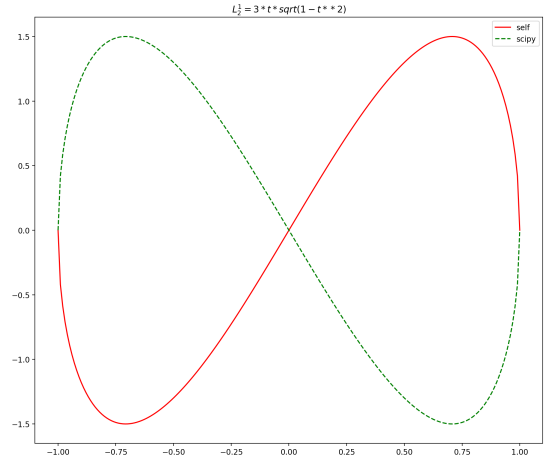
```
def Legendre_shuzhi(t,l,m):
    if m==0:
        L0m=1
        L1m=t
        L2m=1/2*(3*t**2-1)
    elif abs(m)==1:
        L0m=0
        L1m=sy.sqrt(1-t**2)
        L2m=3*sy.sqrt(1-t**2)*t
    elif abs(m)==2:
        L0m=0
        L1m=0
        L2m=3*(1-t**2)
    else:
```

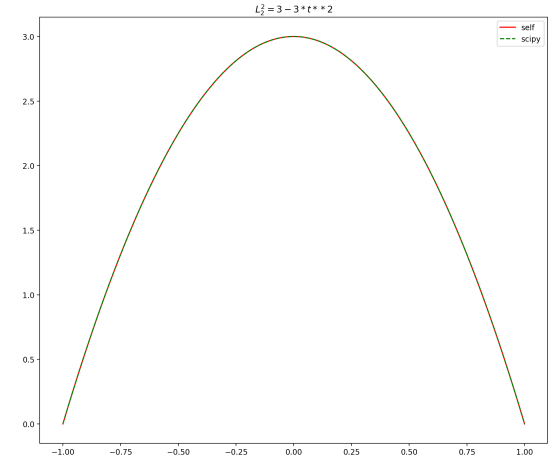
```
        L0m=0
        L1m=0
        L2m=0
    if l==0:
        return L0m
    elif l==1:
        return L1m
    elif l==2:
        return L2m
    elif l>2 and (l-m)!=0:
        temp=(2*l-1)*t/(l-m)*Legendre_shuzhi(t,l-1,m)-(l-1+m)/(l-
m)*Legendre_shuzhi(t,l-2,m)
        return temp
    elif l>2 and (l-m)==0:
        temp=Legendre_shuzhi(t,l-2,m)+(2*l-1)*sy.sqrt(1-t**2)*Legendre_shuzhi(t,l-1,m-
1)
        return temp
```

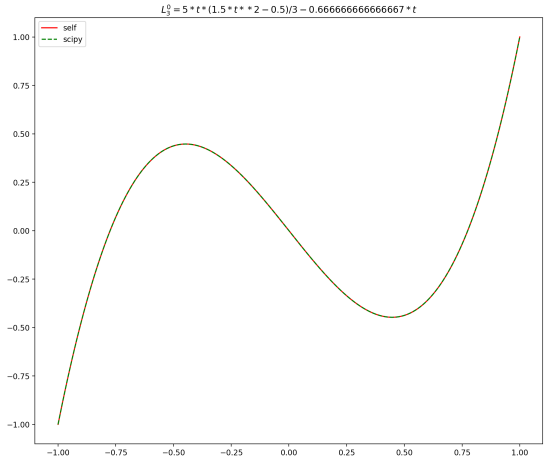
```
import matplotlib.pyplot as plt
import numpy as np
import scipy.special as special

plt.figure(figsize=(30,60),dpi=200)
klist=[1,1,2,2,3,3,3,4,4,4]
plist=[0,1,1,2,0,1,2,2,3,4]
for i in range(1,11):
    plt.subplot(5,2,i)
    l=klist[i-1]
    m=plist[i-1]
    t=sy.symbols('t')
    title=Legendre(t,l,m)
    plt.title('$L_{}^{}={}$'.format(l,{m},title))
    plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=0.5,
hspace=None)
    ff=np.linspace(-1,1,200)
    ylist=[]
    ytrue=[]
    for j in range(0,len(ff)):
        temp1=ff[j]
        yy1=Legendre_shuzhi(temp1,l,m)
        yy2=special.lpmv(m,l,temp1)
        ylist.append(yy1)
        ytrue.append(yy2)
    plt.plot(ff,ylist,'r',label='self')
    plt.plot(ff,ytrue,'g',label='scipy',linestyle='dashed')
    plt.legend()
```

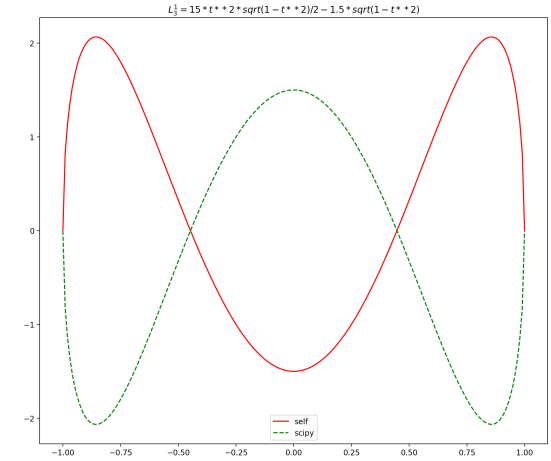

$L_1^0 = t$
self
scipy

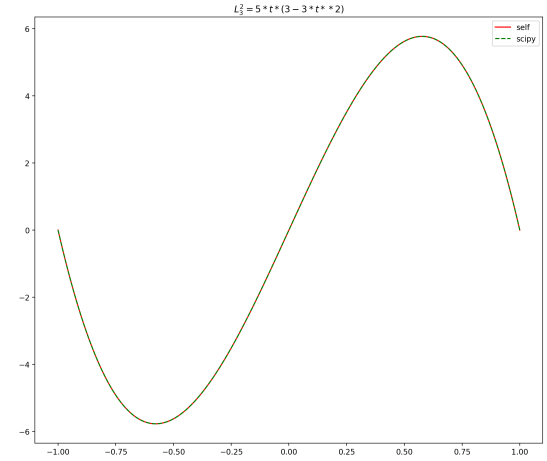

$L_1^1 = sqrt(1 - t**2)$
self
scipy

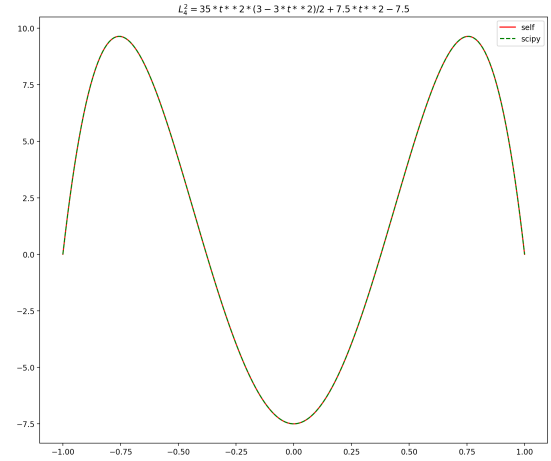

$L_2^1 = 3*t*sqrt(1 - t**2)$
self
scipy

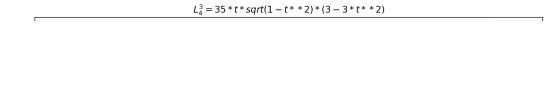

$L_2^2 = 3 - 3*t**2$
self
scipy

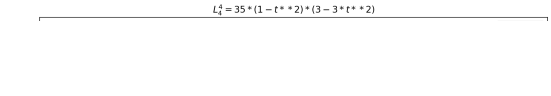

$L_3^0 = 5*t*(1.5*t**2 - 0.5)/3 - 0.6666666666666667*t$
self
scipy

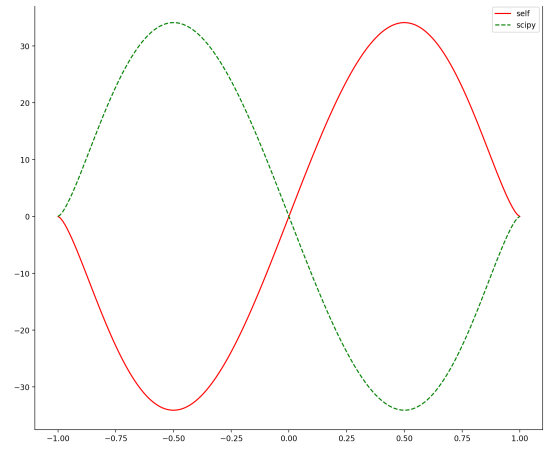

$L_3^1 = 15*t**2*sqrt(1 - t**2)/2 - 1.5*sqrt(1 - t**2)$
self
scipy

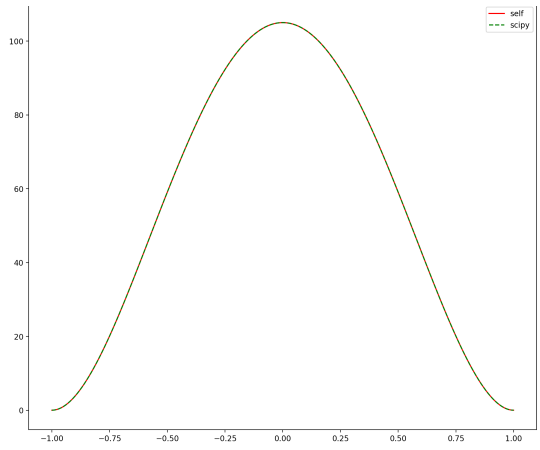

$L_3^2 = 5*t*(3 - 3*t**2)$
self
scipy


$L_4^2 = 35*t**2*(3 - 3*t**2)/2 + 7.5*t**2 - 7.5$
self
scipy


$L_4^3 = 35*t*sqrt(1 - t**2)*(3 - 3*t**2)$


$L_4^4 = 35*(1 - t**2)*(3 - 3*t**2)$





可以看到，部分值由于初始设置的正负而产生了正反差异

球谐函数

```
def Ball(l,m):
    j1=sy.symbols('j1')
    j2=sy.symbols('j2')
    if m>=0:
        epsi=(-1)**m
    else:
        epsi=1
    YY=epsi*np.sqrt((2*l+1)*np.math.factorial(l-
abs(m))/4/np.pi/np.math.factorial(l+abs(m)))*sy.exp(1j*m*j1)*Legendre(sy.cos(j2),l,m)
    return YY
```

```
Ball(3,2)
```

$$0.340661825477761 \left(3 - 3 \cos^{2}\left(j_{2}\right)\right) e^{2.0 i j_{1}} \cos\left(j_{2}\right)$$

与量子力学课本形式一致

推导

氢原子的势能$V（r）= -\frac{e^2}{4\pi\varepsilon_0}\frac{1}{r}$

径向方程：$-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2u}{\mathrm{d}r^2}+\left[-\frac{e^2}{4\pi\varepsilon_0}\frac{1}{r}+\frac{\hbar^2}{2m}\frac{l(l+1)}{r^2}\right]u=uE$

首先需要对形式进行简化

$\kappa \equiv \frac{\sqrt{-2mE}}{\hbar}$

引入$\rho \equiv \kappa r$和$\rho_0 \equiv \frac{me^2}{2\pi\varepsilon_0\hbar^2\kappa}$

这样经过化简可以得到

$\frac{\mathrm{d}^2u}{\mathrm{d}\rho^2}=[1-\frac{\rho_0}{\rho}+\frac{l(l+1)}{\rho^2}]u$

分别在$\rho\to\infty$和$\rho\to0$下进行考察

对ρ较大时，有$u(\rho)\approx A\exp(-\rho)$

对ρ较小时，有$u(\rho)\approx C\rho^{l+1}$

引入新的函数$v(\rho)$

$u(\rho)=\rho^{l+1}e^{-\rho}v(\rho)$

$v(\rho)$可以展开为幂级数，确定系数，逐项求导

化简可以由递推关系得到

$v(\rho)=c_0e^{2\rho}$

为了级数截断 $c_{(jmax+1)}=0$

主量子数$n\equiv jmax+l+1$

$\rho_0=2n$

$E=\frac{-\hbar^2\kappa^2}{2m}$

径向波函数：$R_{nl}(r)=\frac{1}{r}\rho^{l+1}e^{-\rho}v(\rho)$

$v(\rho)=L_{n-l-1}^{2l+1}(2\rho)$

最终可以得到氢原子的归一化波函数为

$\psi_{nlm}=\sqrt{(\frac{2}{na})^3\frac{(n-l-1)!}{2n[(n+l)!]^3}}e^{-r/na}(\frac{2r}{na})^l[L_{n-l-1}^{2l+1}(2r/na)]Y_l^m(\theta,\phi)$

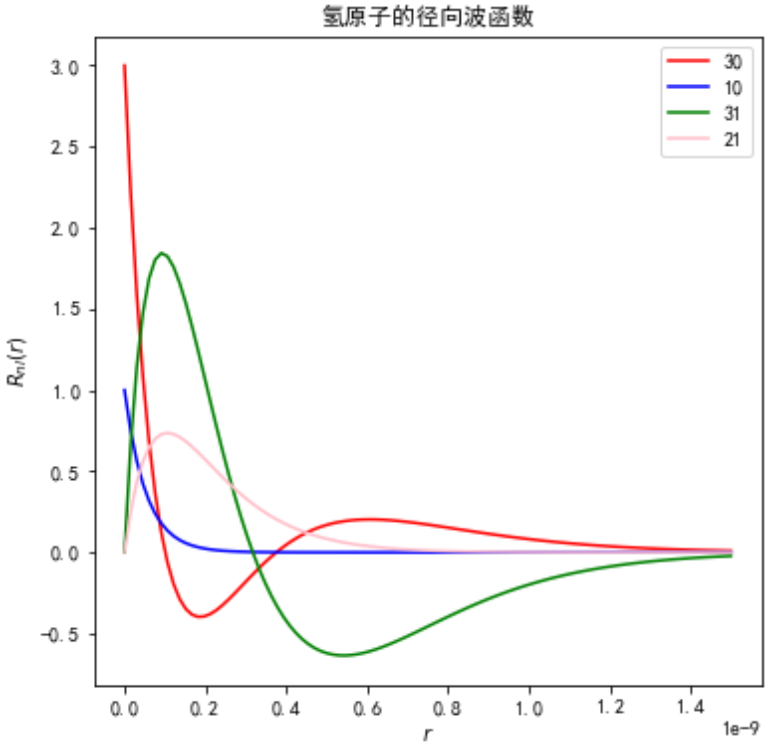
径向波函数绘制

```
def Rj(r,n,l):
    a0=0.529*10**(-10)
    p=r/a0/n
    v=Laguerre_shuzhi(2*p,n-l-1,2*l+1)
    R=(2*p)**(l)*np.exp(-p)*v
    return R
```

```
fig=plt.figure(figsize=(6,6))
Rlist1=np.linspace(0.0000001,1.5,100)
Rlist=[i*10**(-9) for i in Rlist1]
ylist1=[]
ylist2=[]
ylist3=[]
```

```
ylist4=[]
for i in range(0,len(Rlist)):
    temp=Rj(Rlist[i],3,0)
    ylist1.append(temp)
    temp1=Rj(Rlist[i],1,0)
    ylist2.append(temp1)
    temp2=Rj(Rlist[i],3,1)
    ylist3.append(temp2)
    temp3=Rj(Rlist[i],2,1)
    ylist4.append(temp3)
plt.plot(Rlist,ylist1,c='r',label='30')
plt.plot(Rlist,ylist2,c='b',label='10')
plt.plot(Rlist,ylist3,c='g',label='31')
plt.plot(Rlist,ylist4,c='pink',label='21')
plt.legend()
plt.rcParams['font.sans-serif']=['Simhei']
plt.rcParams['axes.unicode_minus']=False
plt.title("氢原子的径向波函数")
plt.ylabel('$R_{nl}(r)$')
plt.xlabel('$r$')
```

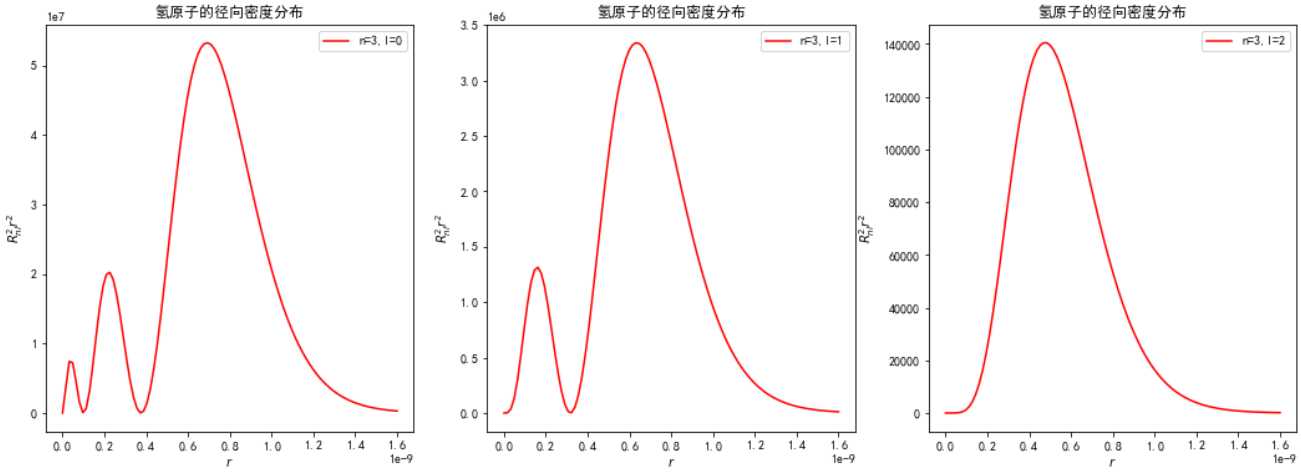
```
Text(0.5, 0, '$r$')
```



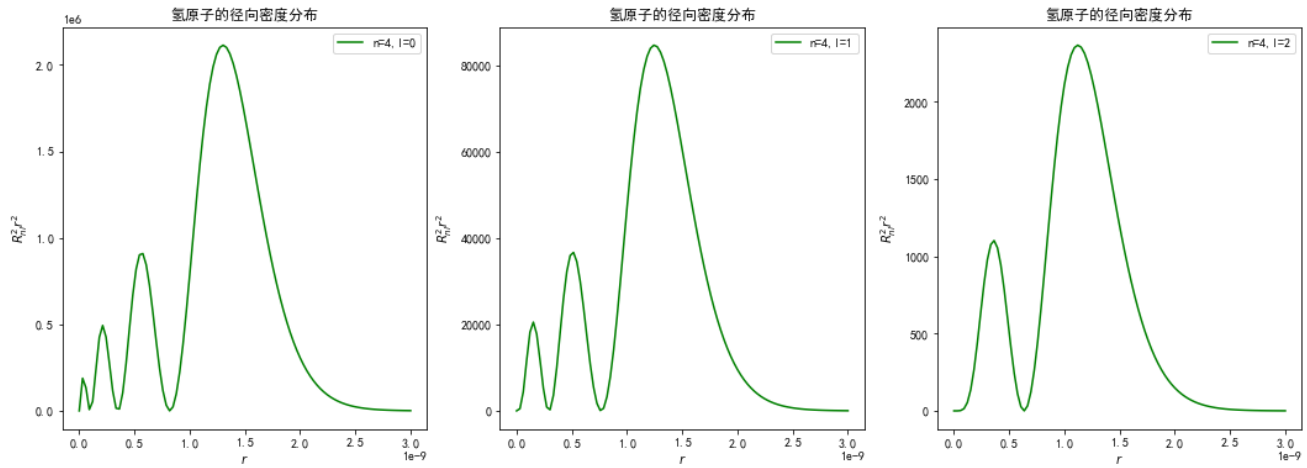
径向密度分布

```
import math
def Rou(r,n,l):
    a=0.529*10**(-10)
    A=np.sqrt((2/n/a)**3*np.math.factorial(n-l-1)/2/n/(np.math.factorial(n+l))**3)
    #print('A',A)
    re=(Rj(r,n,l)*A*r)**2
    #print(Rj(r,n,l))
    return re
```

```
fig=plt.figure(figsize=(18,6))
Rlist1=np.linspace(0.0000001,1.6,100)
Rlist=[i*10**(-9) for i in Rlist1]
ylist1=[]
ylist2=[]
ylist3=[]
ylist4=[]
for i in range(0,len(Rlist)):
    temp=Rou(Rlist[i],3,0)
    ylist1.append(temp)
    temp2=Rou(Rlist[i],3,1)
    ylist3.append(temp2)
    temp3=Rou(Rlist[i],3,2)
    ylist4.append(temp3)
yresu=[ylist1,ylist3,ylist4]
nlist=[3,3,3]
llist=[0,1,2]
for j in range(1,4):
    plt.subplot(1,3,j)
    plt.plot(Rlist,yresu[j-1],c='r',label='n={},l={}'.format(nlist[j-1],llist[j-1]))

    plt.legend()
    plt.rcParams['font.sans-serif']=['Simhei']
    plt.rcParams['axes.unicode_minus']=False
    plt.title("氢原子的径向密度分布")
    plt.ylabel('$R_{nl}^2r^2$')
    plt.xlabel('$r$')
```




```
fig=plt.figure(figsize=(18,6))
Rlist1=np.linspace(0.0000001,3,100)
Rlist=[i*10**(-9) for i in Rlist1]
ylist1=[]
ylist2=[]
ylist3=[]
ylist4=[]
for i in range(0,len(Rlist)):
    temp=Rou(Rlist[i],4,0)
    ylist1.append(temp)
    temp2=Rou(Rlist[i],4,1)
    ylist3.append(temp2)
    temp3=Rou(Rlist[i],4,2)
    ylist4.append(temp3)
yresu=[ylist1,ylist3,ylist4]
nlist=[4,4,4]
llist=[0,1,2]
for j in range(1,4):
    plt.subplot(1,3,j)
    plt.plot(Rlist,yresu[j-1],c='g',label='n={},l={}'.format(nlist[j-1],llist[j-1]))

    plt.legend()
    plt.rcParams['font.sans-serif']=['Simhei']
    plt.rcParams['axes.unicode_minus']=False
    plt.title("氢原子的径向密度分布")
    plt.ylabel('$R_{nl}^2r^2$')
    plt.xlabel('$r$')
```



角向分布

```
def Ball1(l,m,j1,j2):
    if m>=0:
        epsi=(-1)**m
    else:
        epsi=1
    YY=epsi*np.sqrt((2*l+1)*np.math.factorial(l-
abs(m))/4/np.pi/np.math.factorial(l+abs(m)))*
(np.cos(m*j1)+1j*np.sin(m*j1))*Legendre_shuzhi(np.cos(j2),l,m)
    return YY
```

```
test=Ball1(2,1,np.pi/2,np.pi/2)
print(test)
special.sph_harm(2, 1, np.pi/2, np.pi/2)
```

```
-2.89659256637091e-33 - 4.73049465100897e-17*I
```

```
(nan+0j)
```

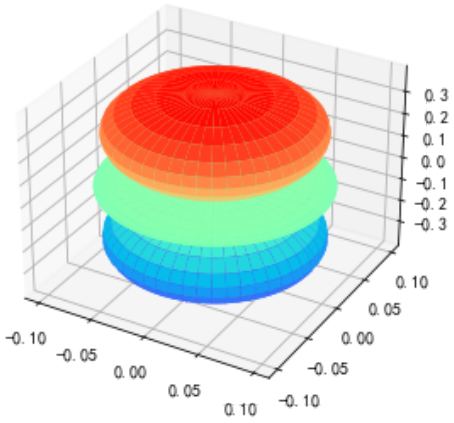
```
import numpy as np
import scipy.special
import matplotlib.pyplot as plt
def sph(r, theta, phi):
    x = r * np.sin(theta) * np.cos(phi)
    y = r * np.sin(theta) * np.sin(phi)
    z = r * np.cos(theta)
    return np.array([x, y, z])
```

```
l = 2
m = 0
Ylm=[]
num=100
theta = np.linspace(0, np.pi, num)
phi = np.linspace(0, 2 * np.pi, num)
Theta, Phi = np.meshgrid(theta, phi)
Ylm1 = Ball1(l, m, Phi, Theta)
for i in range(0,num):
    for j in range(0,num):
        Ylm1[i][j]=complex(Ylm1[i,j])
Ylm2 = np.abs(Ylm1)**2
X, Y, Z = sph(Ylm2, Theta, Phi)
plt.rcParams['font.sans-serif']=['Simhei']
plt.rcParams['axes.unicode_minus']=False
fig = plt.figure(figsize=(16,16))
ax = fig.add_subplot(3,2,1,projection = "3d")
ax.set_title("Y%d%d自己编写的特殊函数" % (l, m))
ax.plot_surface(X, Y, Z, cmap="rainbow")


l = 2
m = 0
Ylm=[]
num=100
theta = np.linspace(0, np.pi, num)
phi = np.linspace(0, 2 * np.pi, num)
Theta, Phi = np.meshgrid(theta, phi)
Ylm = special.sph_harm(m, l, Phi, Theta)
Ylm2 = np.abs(Ylm)**2
X, Y, Z = sph(Ylm2, Theta, Phi)
plt.rcParams['font.sans-serif']=['Simhei']
plt.rcParams['axes.unicode_minus']=False
ax = fig.add_subplot(3,2,2,projection = "3d")
ax.set_title("Y%d%dscipy特殊函数" % (l, m))
ax.plot_surface(X, Y, Z, cmap="rainbow")

l = 2
m = 2
Ylm=[]
num=100
theta = np.linspace(0, np.pi, num)
phi = np.linspace(0, 2 * np.pi, num)
Theta, Phi = np.meshgrid(theta, phi)
Ylm1 = Ball1(l, m, Phi, Theta)
for i in range(0,num):
    for j in range(0,num):
        Ylm1[i][j]=complex(Ylm1[i,j])
Ylm2 = np.abs(Ylm1)**2
X, Y, Z = sph(Ylm2, Theta, Phi)
plt.rcParams['font.sans-serif']=['Simhei']
```

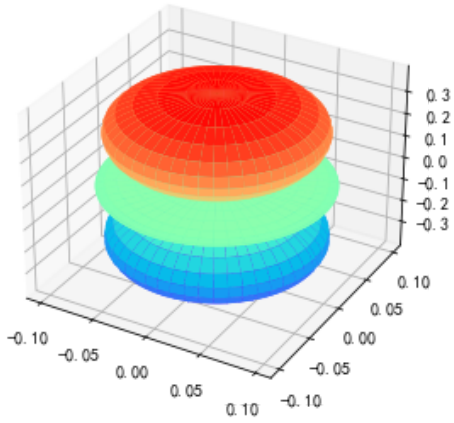
```
plt.rcParams['axes.unicode_minus']=False
fig = plt.figure(figsize=(16,16))
ax = fig.add_subplot(3,2,3,projection = "3d")
ax.set_title("Y%d%d自己编写的特殊函数" % (l, m))
ax.plot_surface(X, Y, Z, cmap="rainbow")


l = 2
m = 2
Ylm=[]
num=100
theta = np.linspace(0, np.pi, num)
phi = np.linspace(0, 2 * np.pi, num)
Theta, Phi = np.meshgrid(theta, phi)
Ylm = special.sph_harm(m, l, Phi, Theta)
Ylm2 = np.abs(Ylm)**2
X, Y, Z = sph(Ylm2, Theta, Phi)
plt.rcParams['font.sans-serif']=['Simhei']
plt.rcParams['axes.unicode_minus']=False
ax = fig.add_subplot(3,2,4,projection = "3d")
ax.set_title("Y%d%dscipy特殊函数" % (l, m))
ax.plot_surface(X, Y, Z, cmap="rainbow")

l = 4
m = 2
Ylm=[]
num=100
theta = np.linspace(0, np.pi, num)
phi = np.linspace(0, 2 * np.pi, num)
Theta, Phi = np.meshgrid(theta, phi)
Ylm1 = Ball1(l, m, Phi, Theta)
for i in range(0,num):
    for j in range(0,num):
        Ylm1[i][j]=complex(Ylm1[i,j])
Ylm2 = np.abs(Ylm1)**2
X, Y, Z = sph(Ylm2, Theta, Phi)
plt.rcParams['font.sans-serif']=['Simhei']
plt.rcParams['axes.unicode_minus']=False
fig = plt.figure(figsize=(16,16))
ax = fig.add_subplot(3,2,5,projection = "3d")
ax.set_title("Y%d%d自己编写的特殊函数" % (l, m))
ax.plot_surface(X, Y, Z, cmap="rainbow")

l = 4
m = 2
Ylm=[]
num=100
theta = np.linspace(0, np.pi, num)
phi = np.linspace(0, 2 * np.pi, num)
Theta, Phi = np.meshgrid(theta, phi)
Ylm = special.sph_harm(m, l, Phi, Theta)
Ylm2 = np.abs(Ylm)**2
```

```
X, Y, Z = sph(Ylm2, Theta, Phi)
plt.rcParams['font.sans-serif']=['Simhei']
plt.rcParams['axes.unicode_minus']=False
ax = fig.add_subplot(3,2,6,projection = "3d")
ax.set_title("Y%d%dscipy特殊函数" % (l, m))
ax.plot_surface(X, Y, Z, cmap="rainbow")



plt.show()
```

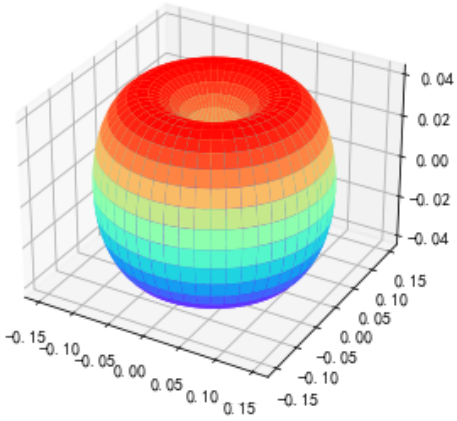
Y20自己编写的特殊函数



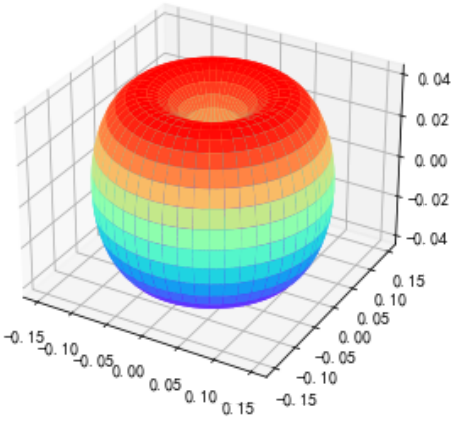
Y20scipy特殊函数



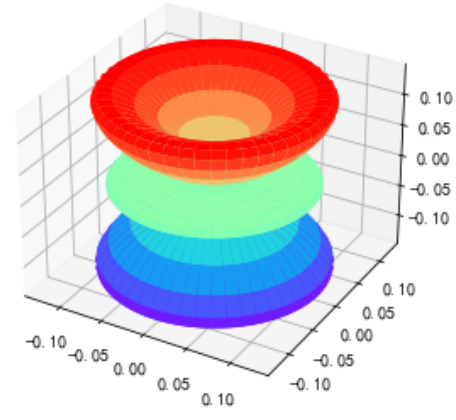
Y22自己编写的特殊函数



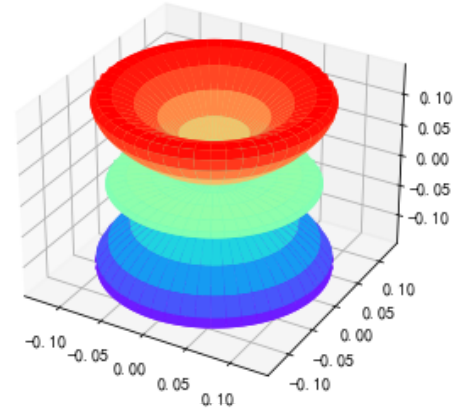
Y22scipy特殊函数



Y42自己编写的特殊函数



Y42scipy特殊函数



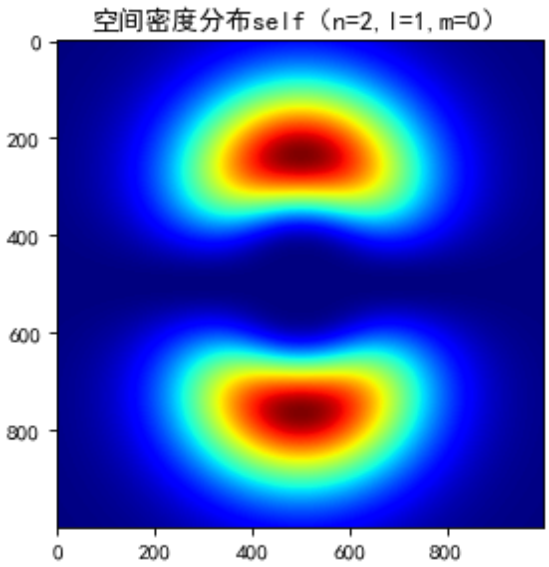
小结：几乎一模一样，但对于某些值，自己编的代码可能不能正常使用sympy求解

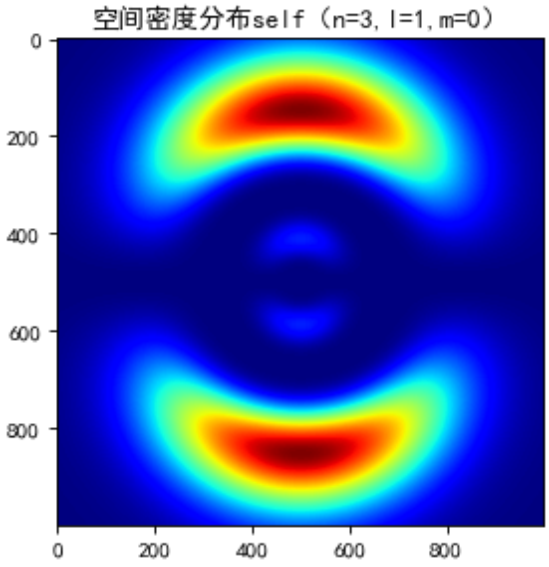
空间密度分布

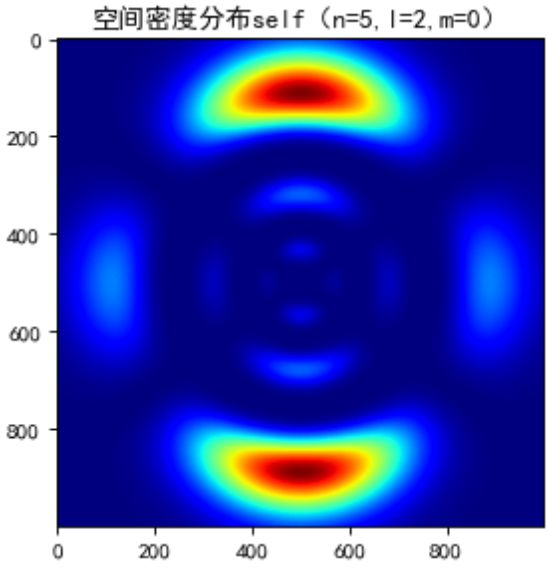
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.special as spe
import scipy.constants as const

def sp(x, y, z):
    r = np.sqrt(x**2+y**2 +z**2)
    theta = np.arccos(z/r)
    phi = np.arctan(y/x)
    return np.array([r, theta, phi])
def wave(r, theta, phi, n, l, m):
    return Rou(r, n, l) * Ball1(l,m, phi, theta)
def wave_true(r, theta, phi, n, l, m):
    return Rou(r, n, l) * special.sph_harm(m, l, phi, theta)
def width(n, l):
    if l < n // 2:
        a = -1e-10 * (n + l)
    else:
        a = -1e-10 * n**2
    return a, -a
n = 2
l = 1
m = 0
a, b = width(n, l)
x = np.linspace(a, b, 1000)
y = 0
z = np.linspace(a, b, 1000)
X, Z = np.meshgrid(x, z)
r, theta, phi = sp(X, y, Z)
psi = wave(r, theta, phi, n, l, m)
density = np.abs(psi)**2
fig = plt.figure(dpi = 81)
ax = fig.gca()
ax.axis('auto')
ax.imshow(density,cmap='jet')
ax.set_title("空间密度分布self（n={},l={},m={}）" .format (n,l,m))
plt.show()

n = 3
l = 1
m = 0
a, b = width(n, l)
x = np.linspace(a, b, 1000)
y = 0
z = np.linspace(a, b, 1000)
X, Z = np.meshgrid(x, z)
r, theta, phi = sp(X, y, Z)
psi = wave(r, theta, phi, n, l, m)
density = np.abs(psi)**2
fig = plt.figure(dpi = 81)
ax = fig.gca()
```

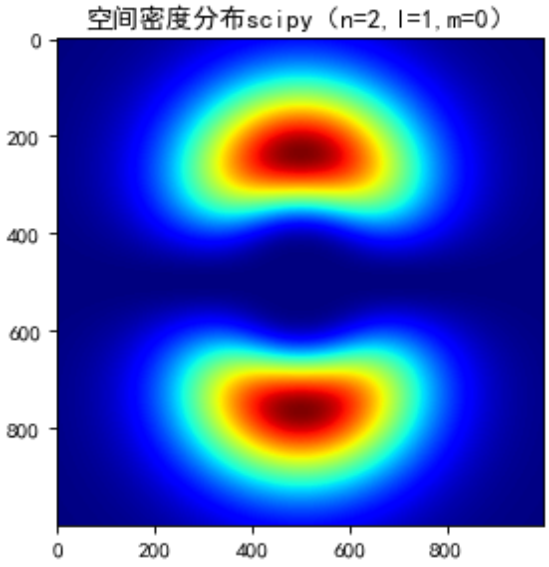
```
ax.axis('auto')
ax.imshow(density,cmap='jet')
ax.set_title("空间密度分布self（n={},l={},m={}）" .format (n,l,m))
plt.show()

n = 5
l = 2
m = 0
a, b = width(n, l)
x = np.linspace(a, b, 1000)
y = 0
z = np.linspace(a, b, 1000)
X, Z = np.meshgrid(x, z)
r, theta, phi = sp(X, y, Z)
psi = wave(r, theta, phi, n, l, m)
density = np.abs(psi)**2
fig = plt.figure(dpi = 81)
ax = fig.gca()
ax.axis('auto')
ax.imshow(density,cmap='jet')
ax.set_title("空间密度分布self（n={},l={},m={}）" .format (n,l,m))
plt.show()



n = 2
l = 1
m = 0
a, b = width(n, l)
x = np.linspace(a, b, 1000)
y = 0
z = np.linspace(a, b, 1000)
X, Z = np.meshgrid(x, z)
r, theta, phi = sp(X, y, Z)
psi = wave_true(r, theta, phi, n, l, m)
density = np.abs(psi)**2
fig = plt.figure(dpi = 81)
ax = fig.gca()
ax.axis('auto')
ax.imshow(density,cmap='jet')
ax.set_title("空间密度分布scipy（n={},l={},m={}）" .format (n,l,m))
plt.show()

n = 3
l = 1
m = 0
a, b = width(n, l)
x = np.linspace(a, b, 1000)
y = 0
z = np.linspace(a, b, 1000)
X, Z = np.meshgrid(x, z)
r, theta, phi = sp(X, y, Z)
```

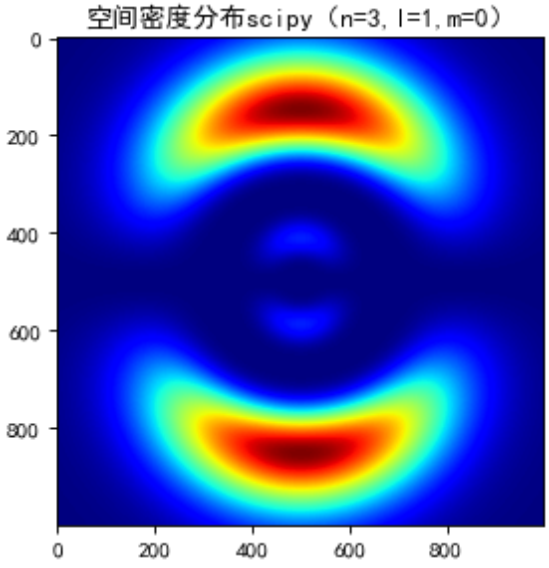
```
psi = wave_true(r, theta, phi, n, l, m)
density = np.abs(psi)**2
fig = plt.figure(dpi = 81)
ax = fig.gca()
ax.axis('auto')
ax.imshow(density,cmap='jet')
ax.set_title("空间密度分布scipy（n={},l={},m={}）" .format (n,l,m))
plt.show()

n = 5
l = 2
m = 0
a, b = width(n, l)
x = np.linspace(a, b, 1000)
y = 0
z = np.linspace(a, b, 1000)
X, Z = np.meshgrid(x, z)
r, theta, phi = sp(X, y, Z)
psi = wave_true(r, theta, phi, n, l, m)
density = np.abs(psi)**2
fig = plt.figure(dpi = 81)
ax = fig.gca()
ax.axis('auto')
ax.imshow(density,cmap='jet')
ax.set_title("空间密度分布scipy（n={},l={},m={}）" .format (n,l,m))
plt.show()
```

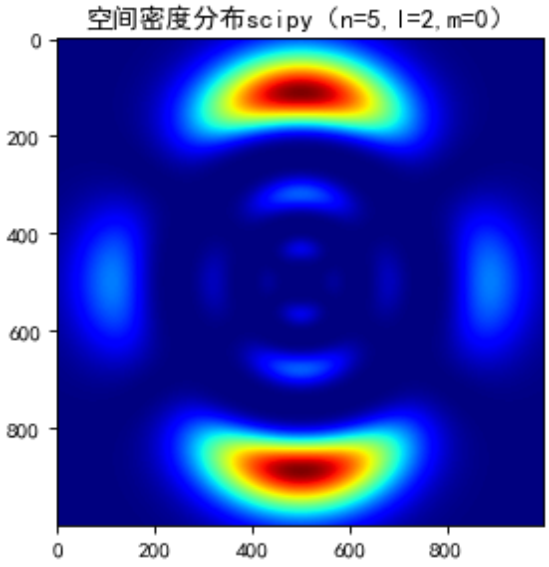



空间密度分布self（n=3, l=1, m=0）
0
200
400
600
800
0
200
400
600
800


空间密度分布self（n=5, l=2, m=0）
0
200
400
600
800
0
200
400
600
800


空间密度分布scipy（n=2, l=1, m=0）
0
200
400
600
800
0
200
400
600
800


空间密度分布scipy（n=3, l=1, m=0）
0
200
400
600
800
0
200
400
600
800


空间密度分布scipy（n=5, l=2, m=0）
0
200
400
600
800
0
200
400
600
800