Υπολογιστική Κβαντομηχανική

Μαρία Λεκού

Ιούνιος 2022

Associated Legendre Polynomials

Διαφορική εξίσωση:

$$\frac{d}{dx}\left[(1-x^2)\frac{P_l^m(x)}{dx}\right] + \left[l(l+1) - \frac{m^2}{1-x^2}\right]P_l^m(x) = 0, \quad -l \leq m \leq l$$

Ή αλλιώς:

$$y(x) = P'(x)$$

$$y'(x) = 2\frac{x \cdot y(x)}{1-x^2} - \left(l(l+1) - \frac{m^2}{1-x^2}\right)\frac{P(x)}{1-x^2}$$

Runge Kutta 4th order Algorithm

```
import numpy as np
def rk4Algor (t,h,N,y,f):
    k1=np.zeros(N)
    k2=np.zeros(N)
    k3=np.zeros(N)
    k4=np.zeros(N)
    k1 = h*f(t,y)
    k2 = h*f(t+h/2.,y+k1/2.)
    k3 = h*f(t+h/2.,y+k2/2.)
    k4 = h*f(t+h,y+k3)
    y = y+(k1+ 2*(k2+k3)+k4)/6.
    return y
```

όπου t είναι η ανεξάρτητη μεταβλητή, h το βήμα, N ο αριθμός εξισώσεων το συστήματος, y οι τιμές της συνάρτησης στο t και f η συνάρτηση.

Listing 6.7

Αρχικοποίηση μεταβλητών
Στον πίνακα y θα αποθηκεύονται οι τιμές P(x),P'(x)

```
N=20000
x = np.zeros((N-2),float)
Plm = np.zeros((N-2),float)
y =np.zeros(2) #Στον y αποθηκεύουμε τις αρχικές τιμές P, dP/dx
step=0.0001
```

```
def normPm0(el,m):
    if m==0:
        if el%2==0:
            y[0]=1
        elif el%2==1:
            y[0]=-1
    return y[0]
```

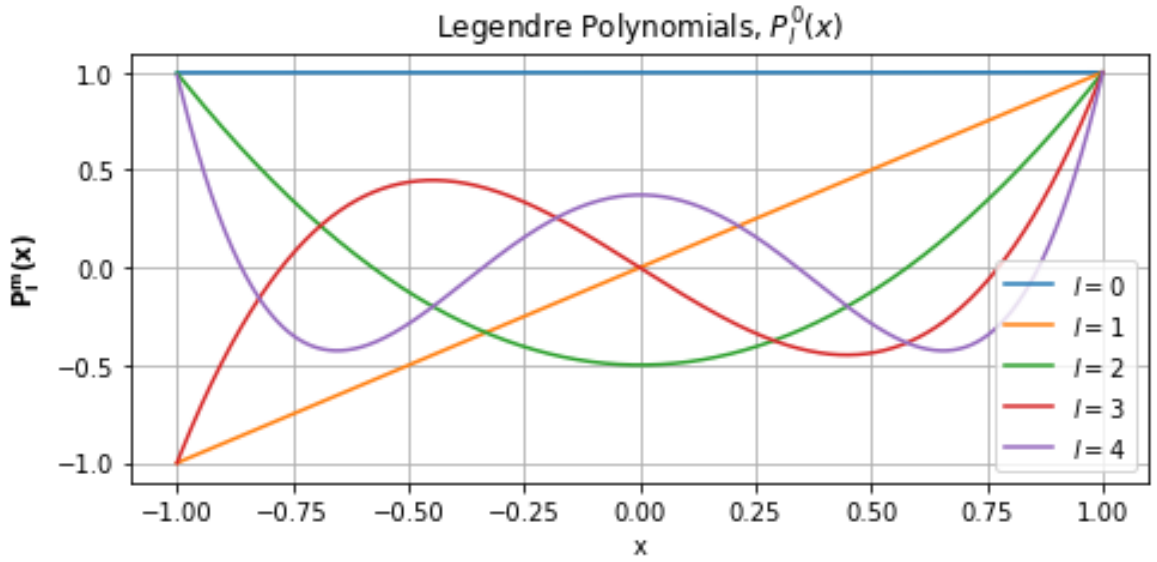
```
def normPm1(el,m):
    if m>0:
        if el>2 and el%2==0:
            y[0]=1
        elif el>0 and el%2==1:
            y[0]=-1
    return y[0]
```

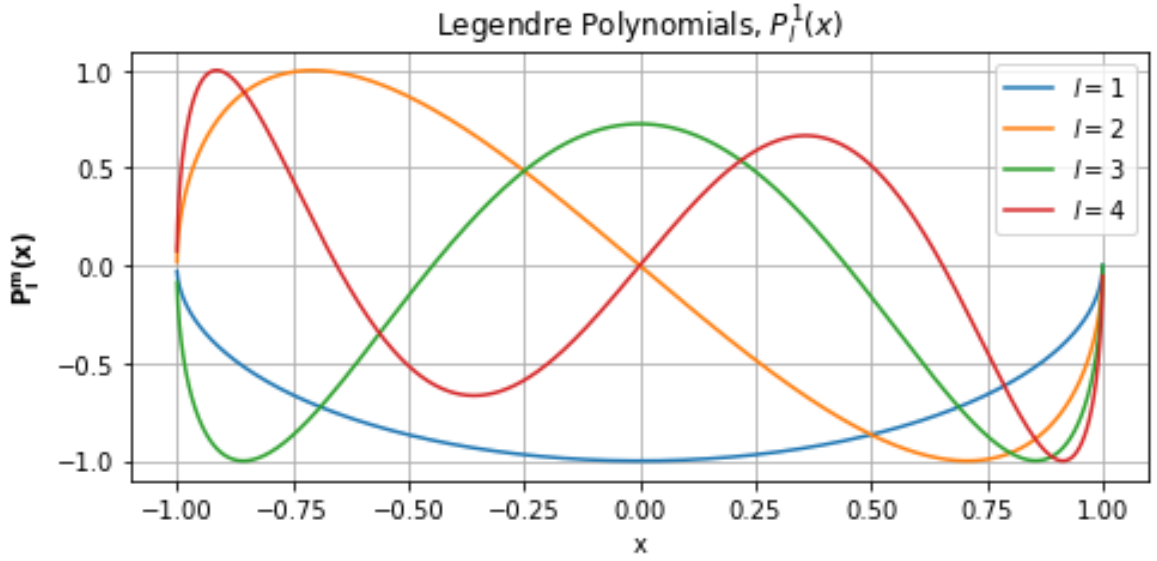
Listing 6.7: Υπολογισμός Plm

Υπολογίζουμε το P_l^m για m=0,1 και l από m έως 5.
Στην f αποθηκεύουμε τη διαφορική που θέλουμε να λύσουμε.

```
for m in range(0,2):
    plt.figure()
    for l in range(m,5):
        i = -1
        if m==0:
            y[0]=normPm0(l,m)
        if m==1:
            y[0]=normPm1(l,m)
        el=l
        def f(xi,yi):
            rhs=np.zeros(2) # Declare array dimension
            rhs[0]=yi[1]
            rhs[1]=2*xi*yi[1]/(1-xi**2)-(el*(el+1)-
                        m**2/(1-xi**2))*yi[0]/(1-xi**2)
            return rhs
        for xi in np.arange (-1+step , 1-step , step ):
            i=i+1
            x[i]=xi
            y=rk4Algor(xi,step,2,y,f)
            Plm[i]=y[0]
        if max(abs(Plm))>1:
            Plm=Plm/max(abs(Plm))
        plt.plot(x,Plm,label=r'$l=$'+str(l))
        plt.grid(True)
        plt.title(r'Legendre Polynomials, $P_l^%i(x)$'%m)
        plt.xlabel('x')
        plt.ylabel('$\mathbf{P_l^m(x)}$')
        plt.legend(loc='best')
plt.show()
```

Listing 6.7: Αποτελέσματα





Listing 6.23

SU(3) generators:

```
L1 = np.array ( [ [ 0 , 1 , 0 ] , [ 1 , 0 , 0 ] , [ 0 , 0 , 0 ] ] )
L2 = np.array ( [ [0 , -1j , 0 ] , [ 1j , 0 , 0 ] , [ 0 , 0 , 0 ] ] )
L3 = np.array ( [ [ 1 , 0 , 0 ] , [ 0 , -1 , 0 ] , [ 0 , 0 , 0 ] ] )
L4 = np.array ( [ [ 0 , 0 , 1 ] , [ 0 , 0 , 0 ] , [ 1 , 0 , 0 ] ] )
L5 = np.array ( [ [0 ,0 , -1j ] , [ 0 , 0 , 0 ] , [ 1j , 0 , 0 ] ] )
L6 = np.array ( [ [ 0 , 0 , 0 ] , [ 0 , 0 , 1 ] , [ 0 , 1 , 0 ] ] )
L7 = np.array ( [ [ 0 , 0 , 0 ] , [ 0 , 0 , -1j ] , [ 0 , 1j , 0 ] ] )
L8 = np.array ( [ [ 1 , 0 , 0 ] , [ 0 , 1 , 0 ] , [ 0 , 0 , -2 ] ] ) *1/np.sqrt( 3 )
```

Quarks:

```
u = np.array([1,0,0]) # Up
d = np.array([0,1,0]) # Down
s = np.array([0,0,1]) # Strange
```

Listing 6.23

Τελεστές αναβίβασης:

```
Ip = 0.5*(L1+1j*L2)
Vp = 0.5*(L4+1j*L5)
Up = 0.5*(L6+1j*L7)
```

Τελεστές καταβίβασης:

```
Im = 0.5*(L1-1j*L2)
Vm = 0.5*(L4-1j*L5)
Um = 0.5*(L6-1j*L7)
```

Αν το φανταστικό μέρος είναι 0 δεν το εμφανίζει

```
def conv(vector):
    flag=0
    for i in np.imag(vector):
        if i!=0:
            flag=1
    if flag==0:
        vector=np.real(vector)
    return vector
```

Listing 6.23: Αποτελέσματα

Επίδραση των τελεστών στα quarks:

```
Ipxd=np.dot(Ip,d) # Raise d to u
print ("I\u208Ad=", conv(Ipxd),"= u")
```

output: $I_+d = [1.\ 0.\ 0.] = u$

```
Vpxs=np.dot(Vp,s) # Raise s to u
print ("V\u208As=",conv(Vpxs),"= u")
```

output: $V_+s = [1.\ 0.\ 0.] = u$

```
Upxs=np.dot(Up,s) # Raise s to d
print ("U\u208As=",conv(Upxs),"= d")
```

output: $U_+s = [0.\ 1.\ 0.] = d$

```
Imxu=np.dot(Im,u) #Lower u to d
print("I\u208Bu=",conv(Imxu),"= d")
```

output: $I_-u = [0.\ 1.\ 0.] = d$

```
Vmxu=np.dot(Vm,u) #Lower u to s
print("V\u208Bu=",conv(Vmxu),"= s")
```

output: $V_-u = [0.\ 0.\ 1.] = s$

```
Umxd=np.dot(Um,d) #Lower d to s
print("U\u208Bd=",conv(Umxd),"= s")
```

output: $U_-d = [0.\ 0.\ 1.] = s$

```
Ipxu=np.dot(Ip,u) #What happens if I+ to u
print("I\u208Au=",conv(Ipxu))
```

output: $I_+u = [0.\ 0.\ 0.]$

```
Upxs=np.dot(Um,s) #What happens if U- to s
print("U\u208Bs=",conv(Upxs))
```

output: $U_-s = [0.\ 0.\ 0.]$

Τέλος παρουσίασης