Υπολογιστική Κβαντομηχανική

Τσιτλακίδης Θεόδωρος 4409

ΠΜΣ Υπολογιστικής Φυσικής

Ιούνιος 2021

Πίνακας Περιεχομένων

Περιεχόμενο

Bound States and Numerov Algorithm

Το απλούστερο πρόβλημα της Κβαντομηχανικής είναι το δέσμιο σωμάτιο, ενέργειας E, εντός μονοδιάστατου πηγαδιού δυναμικού.Η κυματοσυνάρτησή του θα ορίζεται από την χρονοανεξάρτητη εξίσωση Schrödinger:

$$\frac{d^2\psi(x)}{dx^2} - \frac{2m}{\hbar^2}V(x)\psi(x) = \kappa^2\psi(x)$$

με $\kappa^2 = -\frac{2m}{\hbar^2}E$.
Το δέσμιο σωμάτιο θα είναι είναι περιορισμένο σε μια περιοχή του χώρου, και η κυματοσυνάρτηση του θα πρέπει να είναι κανονικοποιημένη

$$\psi(x) \to \begin{cases} e^{-\kappa x}, & \text{for } x \to \infty \\ e^{\kappa x}, & \text{for } x \to -\infty \end{cases}$$

Για να υπολογίσουμε την κυματοσυνάρτηση πρέπει να εφαρμόσουμε συνοριακές συνθήκες, γεγονός που μετατρέπει το πρόβλημα σε πρόβλημα ιδιοτιμών.

Numerov Algorithm

Υπολογιστικά εφαρμόζουμε τον αλγόριθμο Numerov (πιο αποδοτικός από RK4 λόγω απουσίας όρων 1ης τάξης):

$$\psi(x+h) \simeq \frac{2(1-\frac{5}{12}h^2k^2(x))\psi(x) - (1+\frac{h^2}{12}k^2(x-h))\psi(x-h)}{1+h^2k^2(x+h)/12}$$

Παρατηρούμε ότι χρησιμοποιεί, τιμές της κυματοσυνάρτησης από δύο προηγούμενες θέσεις x και $x-h$ για τον υπολογισμό κάθε βήματος (εδώ $x+h$).Για να κινηθούμε αντίθετα στον άξονα του x αντιστρέφουμε το πρόσημο του βήματος h.

SU(3) Symmetry Group

Η ομάδα συμμετρίας SU(3) αποτελείται από 8 γεννήτορες $(\lambda_1, \lambda_2, ...\lambda_8)$ που δρούν σε τρισδιάστατες καταστάσεις.Οι καταστάσεις βάσεως της συγκεκριμένης ομάδας είναι:

$$|u\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad |d\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad |s\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Δηλαδή τα u,d και s quarks.
Οι γεννήτορες της ομάδας μπορούν να συνδυαστούν, παράγοντας τελεστές υποβίβασης και αναβίβασης:

$$I_{\pm} = \frac{1}{2}(\lambda_1 \pm i\lambda_2), \quad U_{\pm} = \frac{1}{2}(\lambda_6 \pm i\lambda_7), \quad V_{\pm} = \frac{1}{2}(\lambda_4 \pm i\lambda_5)$$

Listing 6.4:

```
#QuantumNumerov.py: quantum bound state via Numerov
                                    algorithm
#General method, but here for HO V(x)=9.4*x*x/2
# habrc*omega=hbarc*sqrt(k/m)=19.733, r mc**2=940 MeV,
                                    k=9.4
# E=(N+1/2)hbarc*omega = (n+1/2)19.733, N even, change
                                    N odd

from numpy import *
import numpy as np,matplotlib.pyplot as plt

n=1000; m=2; imax=100; Xleft0=-10; Xright0=10
h=0.02; amin=81.; amax=92.; e=amin; de=0.01
eps=1e-4; im=500; nl=im+2; nr=n-im+2
xmax=5.0
print("nl,nr",nl,nr)
print(h)

xLeft = arange(-10,0.02,0.02); xRight = arange(10,0.02
                                    ,-0.02)
xp = arange(-10,10,0.02)          #Bisection interval
uL = zeros([503],float);     uR = zeros([503],float)
```

```
k2L = zeros([1000],float);  k2R = zeros([1000],float)
uL[0]=0; uL[1]=0.00001; uR[0]=0; uR[1]=0.00001;


def V(x):          #Potential harmonic oscillator
    v=4.7*x*x
    return v

def setk2(e):          #Sets k2L=(sqrt(e-V))^2 and k2R
    for i in range(0,n):
        xLeft = Xleft0+i*h
        xr = Xright0-i*h
        fact=0.04829 # 2m*c**2/hbarc**2=2*940/(197.33)
                                          **2
        k2L[i]=fact*(e-V(xLeft))
        k2R[i]=fact*(e-V(xr))


def Numerov(n,h,k2,u,e):
    setk2(e)
    b=(h**2)/12.0          # L and R Psi
    for i in range(1,n):
        u[i+1]=(2*u[i]*(1-5.*b*k2[i])-(1+b*k2[i-1])*u[
```

```
                                        i-1]/(1+b*k2[i+1]))


def diff(e):
    Numerov(nl,h,k2L,uL,e)   #Left wf
    Numerov(nr,h,k2R,uR,e)   #Right wf
    f0 = (uR[nr-1]+uL[nl-1]-uR[nr-3]-uL[nl-3])/(h*uR[
                                    nr-2])
    return f0

istep=0
x1=arange(-10,0.02,0.02); x2=arange(10,-0.02,-0.02)
fig=plt.figure()
ax=fig.add_subplot(111)
ax.grid()

while abs(diff(e))>eps:      #Bisection algorithm
    e=(amin+amax)/2
    print(e,istep)
    if diff(e)*diff(amax)>0: amax=e
    else: amin=e
    ax.clear()
    plt.text(3,-200,'Energy=%10.4f'%(e),fontsize=14)
```

```
    plt.plot(x1,uL[:-2])
    plt.plot(x2,uR[:-2])
    plt.xlabel('x')
    plt.ylabel('$\psi(x)$',fontsize=18)
    plt.title('R&L Wavefunctions Matched at x=0')
    istep=istep+1
    plt.pause(0.8)  #Pause to delay figures
plt.show()
```

Listing 6.4

Το διάγραμμα που λαμβάνουμε τρέχοντας τον κώδικα είναι:

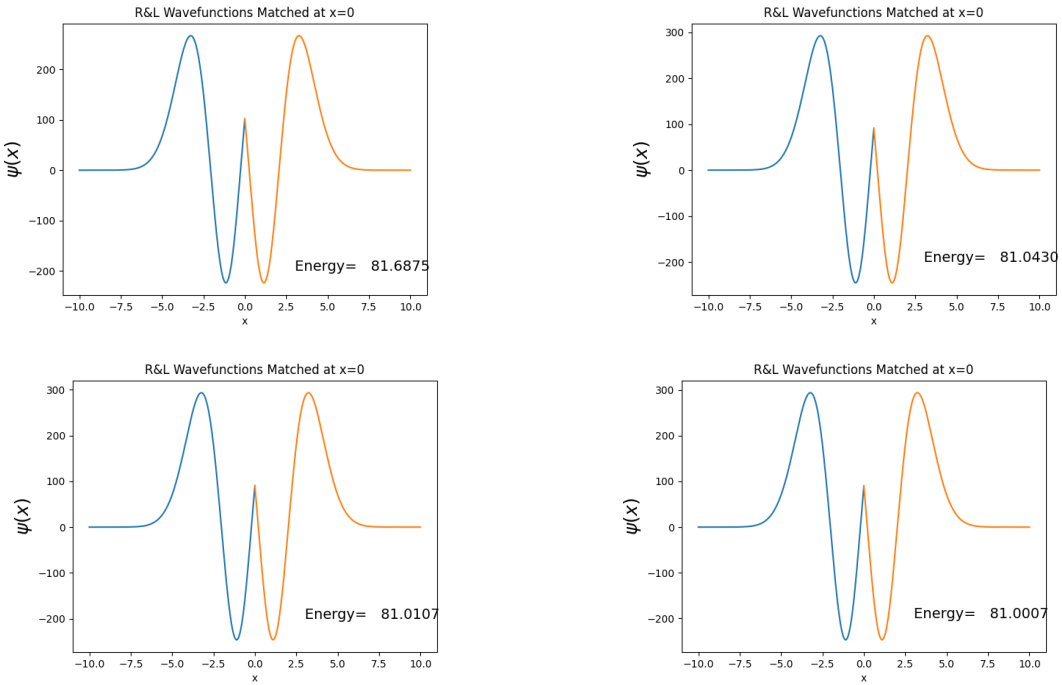

Figure: Η κυματοσυνάρτηση της δέσμιας κατάστασης εντός του πηγαδιού.

Listing 6.23

```
from numpy import *
from numpy.linalg import *

L1 = array([[0,1,0],[1,0,0],[0,0,0]])          #Eight
                                    generators
L2 = array([[0,-1j,0],[1j,0,0],[0,0,0]])
L3 = array([[1,0,0],[0,-1,0],[0,0,0]])
L4 = array([[0,0,1],[0,0,0],[1,0,0]])
L5 = array([[0,0,-1j],[0,0,0],[1j,0,0]])
L6 = array([[0,0,0],[0,0,1],[0,1,0]])
L7 = array([[0,0,0],[0,0,-1j],[0,1j,0]])
L8 = array([[1,0,0],[0,1,0],[0,0,-2]])*1/sqrt(3)
u = array([1,0,0])          #Up quark
d = array([0,1,0])          #Down quark
s = array([0,0,1])          #Strange quark
Ip = 0.5*(L1+1j*L2)         #Raising Operators
Up = 0.5*(L6+1j*L7)
Vp = 0.5*(L4+1j*L5)
Im = 0.5*(L1-1j*L2)         #Lowering Operators
Um = 0.5*(L6-1j*L7)
Vm = 0.5*(L4-1j*L5)
print("Operator matrices of the SU(3) symmetry group:"
```

```
                                    )
Ipxd = dot(Ip,d)            #Raise d to u
print("Ipxd:",Ipxd)
Vpxs = dot(Vp,s)            #Raise s to u
print("Vpxs:",Vpxs)
Upxs = dot(Up,s)            #Raise s to d
print("Upxs:", Upxs)
```

Τρέχοντας τον κώδικα, οι τελεστές αναβίβασης και υποβιβασμού προκύπτουν ίσοι με:

```
Operator matrices of the SU(3) symmetry group:
Ipxd: [1.+0.j 0.+0.j 0.+0.j]
Vpxs: [1.+0.j 0.+0.j 0.+0.j]
Upxs: [0.+0.j 1.+0.j 0.+0.j]
```