Równania całkowe w mechanice kwantowej

Jakub Jurczak, Piotr Głownia, Szymon Kuliński, Janusz Utrata
Bartek Sroczyński

Contents

1 Stany związane nielokalnych potencjałów

Zagadnienie: Cząsteczka doświadcza wielokrotnych oddziaływań z ośrodkiem co skutkuje odczuwaniem przez cząsteczke efektywnego potencjału w **r**. Potencjał ten zależy od funkcji falowej w punktach **r'** innych cząsteczek:

$$V(r)\psi(r) \to \int dr' V(r,r')\psi(r') \tag{1}$$

Ten rodzaj interakcji zwany jest *nielokalnym* i prowadzi do równania Schrödingera, które jest równaniem całkowo-różniczkowym:

$$-\frac{1}{2\mu}\frac{d^2\psi(r)}{dr^2} + \int dr' V(r,r')\psi(r') = E\psi(r) \tag{2}$$

Należy rozwiązać następujące zagadnienie: Znaleźć energi E stanów związanych i funkcje falowe ψ dla równania całkowego (2).Przyjmujemy notację jednostek naturalnych gdzie $\hbar = 1$.

2 Opis problemu

2.1 Równanie Schrödingera w reprezentacji przestrzeni pędu

Jedna możliwa technika rozwiązania równania (2) polega na przejściu do przestrzeni pędu, wtedy równanie staje się równaniem całkowym:

$$\frac{k^2}{2\mu}\psi(k) + \frac{2}{\pi}\int_0^\infty dp p^2 V(k,p)\psi(p) = E\psi(k) \tag{3}$$

gdzie zastrzegamy nasze rozwiązania do $l = 0$ cząstkowych fal. W równaniu (3) $V(k,p)$ jest podwójną transformatą Fourier'a potencjału:

$$V(k,p) = \frac{1}{kp}\int_0^\infty dr \sin(kr) V(r) \sin(pr) \tag{4}$$

a $\psi(k)$ jest nieunormowaną funkcją falową przestrzeni pędu (amplitudą prawdopodobieństwa dla znalezienia cząstki z pędem k).

$$\psi(k) = \int_0^\infty dr kr\psi(r) \sin(kr) \tag{5}$$

Równanie (3) jest równaniem całkowym dla $\psi(k)$ w przeciwieństwie do całkowej reprezentacji $\psi(k)$, gdyż wtedy całka ta nie może być policzona dopóki $\psi(p)$ jest znane. Pokażemy niżej, że równanie to może być przekształcone w równanie macierzowe rozwiązywalne przez operacje na macierzach.

2.2 Transformacja z całki w równania liniowe

Aproksymujemy całkę w stosunku do potencjału jako ważona suma N punktów całkowań dla $p = k_j$ i $j = 1, ..., N$:

$$\int_0^\infty dp p^2 V(k,p)\psi(p) \approx \sum_{j=1}^N w_j k_j^2 V(k,k_j)\psi(k_j) \tag{6}$$

Przekształcenie to sprawia, że (3) staje się równaniem algebraicznym:

$$\frac{k^2}{2\mu}\psi(k) + \frac{2}{\pi}\sum_{j=1}^N w_j k_j^2 V(k,k_j)\psi(k_j) = E \tag{7}$$

Równanie (7) zawiera N niewiadomych $\psi(k_j)$, pojedynczą niewiadomą E, i niewiadomą funkcją $\psi(k)$. Eliminujemy z potrzeby poznania całość funkcji $\psi(k)$ poprzez restrykcję rozwiązania do tych samych wartości k_i użytych przy aproksymacji całki. To prowadzi do układu N równań liniowych z $(N+1)$ niewiadomych:

$$\frac{k_i^2}{2\mu}\psi(k_i) + \frac{2}{\pi}\sum_{j=1}^{N} w_j k_j^2 V(k_i,k_j)\psi(k_j) = E\psi(k_i), i = 1, ..., N \tag{8}$$

Zapisujemy nasz układ równań dynamicznych jako :

$$[H][\psi] = E[\psi] \tag{9}$$

albo jako macierze:

$$\begin{pmatrix} \frac{k_1^2}{2\mu} + \frac{2}{\pi}V(k_1,k_1)k_1^2 w_1 & \frac{2}{\pi}V(k_1,k_2)k_2^2 w_2 & ... & \frac{2}{\pi}V(k_1,k_N)k_N^2 w_N \\ \frac{2}{\pi}V(k_2,k_1)k_1^2 w_1 & \frac{2}{\pi}V(k_2,k_2)k_2^2 w_2 + \frac{k_2^2}{2\mu} & ... & \\ & \ddots & & \\ ... & ... & ... & \frac{k_N^2}{2\mu} + \frac{2}{\pi}V(k_N,k_N)k_N^2 w_N \end{pmatrix}$$
$$\times \begin{pmatrix} \psi(k_1) \\ \psi(k_2) \\ \ddots \\ \psi(k_N) \end{pmatrix} = E \begin{pmatrix} \psi(k_1) \\ \psi(k_2) \\ \ddots \\ \psi(k_N) \end{pmatrix}$$

Równanie (9) jest macierzową reprezentacją równania Schrodingera (3). Funkcja falowa $\psi(k)$ na siatce wyrażeń to wektor $N \times 1$

$$[\psi(k_i)] = \begin{pmatrix} \psi(k_1) \\ \psi(k_2) \\ \ddots \\ \psi(k_N) \end{pmatrix} \tag{10}$$

Tylko czasami i dla pewnych wartości E komputer będzie w stanie znaleźć rozwiązania. Żeby zobaczyć dlaczego, przekształćmy równanie(9):

$$[H - EI][\psi] = [0] \tag{11}$$

$$[\psi] = [H - EI]^{-1}[0] \tag{12}$$

Równanie to mówi nam: (1) Jeśli odwrotność istnieje, mamy rozwiązanie trywialne $\psi := 0$ (2) Dla nietrywialnego rozwiązania by istniało, założenie o istnieniu odwrotności musi być niewłaściwe. Wiemy natomiast z teorii równań liniowych, że odwrotność macierzy nie występuje przy znikaniu wyznacznika:

$$det[H - EI] = 0 \tag{13}$$

Równanie powyżej to warunek stanu związanego. Jest dodatkowym równaniem potrzebnym do znalezienia unikalnych rozwiązań do problemu wartości własnych.Niezależnie, nie ma gwarancji na to, że rozwiązania do (14) mogą być

zawsze znalezione, ale jeśli są znalezione to są poszukiwanymi wartościami własnymi (9).

2.3 Potencjał delta-powłoka(Model)

Rozważmy lokalny potencjał delta-powłoka aby mieć analityczne rozwiązanie do porównania:

$$V(r) = \frac{\lambda}{2\mu}\delta(r-b) \tag{14}$$

Używamy równania (4) aby określić reprezentację przestrzeni pędu:

$$V(k',k) = \int_0^\infty \frac{\sin(k'r')}{k'k}\frac{\lambda}{2\mu}\delta(r-b)\sin(kr)dr = \frac{\lambda}{2\mu}\frac{\sin(k'b)\sin(kb)}{k'k} \tag{15}$$

Jeżeli energia jest sparametryzowana w stosunku do wektora κ przez $E = \frac{-\kappa^2}{2\mu}$, wtedy dla tego potencjału jest, co najwyżej, jeden stan związany i spełnia on równanie poniżej:

$$e^{-2\kappa b} - 1 = \frac{2\kappa}{\lambda} \tag{16}$$

Stany związane występują tylko dla przyciągających się potencjałów i tylko wtedy kiedy przyciąganie jest silne. Dla obecnego przypadku oznacza to, że musi zachodzić $\lambda < 0$.

3 Implementacja energii związanych

3.1 Algorytm w Python'ie

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

min1 = 0.
max1 = 200.
u = 0.5
b = 10.


def gauss(npts, a, b, x, w):
    pp = 0.
    m = (npts + 1) // 2
    eps = 3. * e - 10
    for i in range(1, m + 1):
        t = cos(math.pi * (float(i) - 0.25) / (float(npts) + 0.5))
```

```
            t1 = 1
            while (abs(t - t1)) >= eps:
                p1 = 1.
                p2 = 0.
                for j in range(1, npts + 1):
                    p3 = p2
                    p2 = p1
                    p1 = ((2 * j - 1) * t * p2 - (j - 1) * p3) / j
                pp = npts * (t * p1 - p2) / (t * t - 1)
                t1 = t
                t = t1 - p1 / pp
            x[i - 1] = -t
            x[npts - i] = t
            w[i - 1] = 2. / ((1. - t * t) * pp * pp)
            w[npts - i] = w[i - 1]
            for i in range(0, npts):
                x[i] = x[i] * (b - a) / 2. + (b + a) / 2.
                w[i] = w[i] * (b - a) / 2.


    for M in range(16, 32, 8):
        z = [-1024, -512, -256, -128, -64, -32, -16, -8, -4, -2]
        for lmbda in z:
            A = zeros((M, M), float)
            WR = zeros(M, float)
            k = zeros(M, float)
            w = zeros(M, float)
            gauss(M, min1, max1, k, w)
            for i in range(0, M):
                for j in range(0, M):
                    VR = lmbda/2/u*sin(k[i]*b)/k[i]*sin(k[j]*b)/k[j]
                    A[i, j] = 2. / math.pi * VR * k[j] * k[j]*w[j]
                    if i == j:
                        A[i, j] += k[i] * k[i] / 2 / u
            Es, evectors = eig(A)
            realev = Es.real
            for j in range(0, M):
                if realev[j] < 0:
                    print("M (size), lmbda, ReE=", M, " ",lmbda," ", realev[j])
                    break
    print("Enter and return any character to quit")
    s = input()
```

4 Nielokalne rozpraszanie potencjału

W tym przypadku cząsteczka ma wystarczająco dużo energii na rozpraszanie. Zagadnienie jakie należy rozwiązać to wyznaczenie przekroju poprzecznego rozpraszania dla rozpraszania w nielokalnym potencjale.

4.1 Równanie Lippmana-Schwingera

Równanie całkowe dla równania Schrodingera związane z amplitudą reakcji lub macierzą R zwane jest równaniem Lippmana-Schwingera:

$$R(k',k) = V(k',k) + \frac{2}{\pi} P \int_0^\infty dp \frac{p^2 V(k',p) R(p,k)}{(k_0^2 - p^2)/2\mu} \tag{17}$$

Równanie to jest falą cząstkową $l = 0$ oraz $\hbar = 1$. W (17) pęd k_0 jest powiązany z energią E i zredukowaną masą μ przez:

$$E = \frac{k_0^2}{2\mu}, \tag{18}$$

$$\mu = \frac{m_1 m_2}{m_1 + m_2} \tag{19}$$

a początkowe i końcowe pędy środka masy k i k' są zmiennymi przestrzeni pędu. Eksperymentalną obserwablą, którą wyciąga się z rozwiązania (17) jest element macierzy diagonalnej $R(k_0, k_0)$, który jest powiązany z przesunięciem fazowym δ_0 rozproszenia i wtedy mamy przekrój poprzeczny:

$$R(k_0, k_0) = -\frac{\tan \delta_l}{\rho} \tag{20}$$

$$\rho = 2\mu k_0 \tag{21}$$

W (17) występuje symbol P oznaczający wartość główną Cauchy'ego co pozwala uniknąć osobliwości przy zerze w mianowniku.

4.2 Redukcja równania całkowego do równań macierzowych

Zapiszmy wartość-główną równania (17) jako całkę oznaczoną:

$$R(k',k) = V(k',k) + \frac{2}{\pi} \int_0^\infty dp \frac{p^2 V(k',p) R(p,k) - k_0^2 V(k',k_0) R(k_0,k)}{(k_0^2 - p^2)/2\mu} \tag{22}$$

Przekształcamy te równanie całkowe w równania liniowe aproksymując całkę jako sumę N punktów całkowań k_j z wagami w_j:

$$R(k,k_0) \approx V(k,k_0) + \tfrac{2}{\pi} \textstyle\sum_{j=1}^N \frac{k_j^2 V(k,k_j) R(k_j,k_0) w_j}{(k_0^2 - k_j^2)/2\mu}$$

$-\frac{2}{\pi}k_0^2 V(k,k_0)R(k_0,k_0)\sum_{m=1}^{N}\frac{w_m}{(k_0^2-k_m^2)/2\mu}$
Zauważamy że ostatni wyraz powyższego równania implementuje kompozycję wartości głównej i eliminuje zachowanie osobliwe poprzedniedniego wyrazu. Równanie zawiera (N + 1) niewiadomych $R(k_j,k_0)$ dla $j=0,...,N$. Zamieniamy je na (N + 1) równoległych równań poprzez ewaluację ich na (N + 1)k siatce zmiennych zawierającej k_0 i punkty całkowań:

$$k=k_i=\begin{cases}k_j, & j=1,...,N\\ k_0, & i=0\end{cases} \tag{23}$$

Mamy teraz (N + 1) równań liniowych dla (N + 1) niewiadomych $R_i \equiv R(k_i,k_0)$:

$$R_i = V_i + \frac{2}{\pi}\sum_{j=1}^{N}\frac{k_j^2 V_{ij}R_j w_j}{(k_0^2-k_j^2)/2\mu} - \frac{2}{\pi}k_0^2 V_{i0}R_0\sum_{m=1}^{N}\frac{w_m}{(k_0^2-k_m^2)/2\mu} \tag{24}$$

Przenosimy równania do postaci macierzowej poprzez wektor mianownikowy D:

$$D_i=\begin{cases}+\frac{2}{\pi}\frac{w_i k_i^2}{(k_0^2-k_i^2)/2\mu} & i=1,...,N\\ -\frac{2}{\pi}\sum_{j=1}^{N}\frac{w_j k_0^2}{(k_0^2-k_j^2)/2\mu} & i=0\end{cases} \tag{25}$$

Teraz równania (24) można zgrabnie zapisać jako

$$R - DVR = [1-DV]R = V \tag{26}$$

gdzie R i V są wektorami kolumnowymi długości $N+1$:

$$[R]=\begin{pmatrix}R_{0,0}\\ R_{1,0}\\ \ddots\\ R_{N,0}\end{pmatrix}, [V]=\begin{pmatrix}V_{0,0}\\ V_{1,0}\\ \ddots\\ V_{N,0}\end{pmatrix} \tag{27}$$

Nazywamy macierz $[1-DV]$ w (26) macierz falową F i zapisujemy równanie całkowe jako równanie macierzowe:

$$[F][R]=[V], F_{ij}=\delta_{ij}-D_j V_{ij} \tag{28}$$

Gdzie R jest wektorem niewiadomym, równanie jest w zwykłej formie $AX=B$. Efektywne komputacyjne rozwiązanie używa eliminacji Gaussa do rozwiązania.

5 Rozwiązanie równania Lippmann'a–Schwinger'a metodą eliminacji Gaussa

5.1 Algorytm w Python'ie

```
import math
from visual.graph import *
from src.bound import gauss
import numpy.linalg as lina

graphscatt = gdisplay(x=0, y=0, xmin=0, xmax=6,ymin=0, ymax=1, width=600,
height=400, title='S wave cross section vs E', xtitle='kb',
ytitle='[ sin(delta) ]**2')
sin2plot = gcurve(color=color.yellow)
M = 27; b = 10.0; n = 26
k = zeros((M),float); x = zeros((M),float); w = zeros((M),float)
Finv = zeros((M,M),float); F = zeros((M,M), float); D = zeros((M),float)
V = zeros((M), float); Vvec = zeros((n+1,1),float)
scale = n/2; lambd = 1.5
gauss(n, 2, 0., scale, k, w)
ko = 0.02

    for m in range(1,901):
        k[n] = ko
        for i in range (0, n):
            D[i]=2/math.pi*w[i]*k[i]*k[i]/(k[i]*k[i]-ko*ko)
        D[n] = 0.
        for j in range(0,n):
            D[n]=D[n]+w[j]*ko*ko/(k[j]*k[j]-ko*ko)
        D[n] = D[n]*(-2./pi)
        for i in range(0,n+1):
            for j in range(0,n+1):
                pot = -b*b * lambd * sin(b*k[i])*sin(b*k[j])/(k[i]*b*k[j]*b)
                F[i][j] = pot*D[j]
                if i==j:
                    F[i][j] = F[i][j] + 1.
            V[i] = pot
        for i in range(0,n+1):
            Vvec[i][0]= V[i]
        Finv = lina.inv(F)
        R = dot(Finv, Vvec)
        RN1 = R[n][0]
        shift = atan(-RN1*ko)
        sin2 = (sin(shift))**2
        sin2plot.plot(pos = (ko*b,sin2))
        ko = ko + 0.2
print('Done')
```

6 Wynik programu

```
M (size), lmbda, ReE= 16    -1024    -49908.190988629794
M (size), lmbda, ReE= 16    -512    -22380.033477865014
M (size), lmbda, ReE= 16    -256    -9284.341736932616
M (size), lmbda, ReE= 16    -128    -3411.0031817533036
M (size), lmbda, ReE= 16    -64    -1075.9599198747562
M (size), lmbda, ReE= 16    -32    -319.43524496555364
M (size), lmbda, ReE= 16    -16    -99.29251195626598
M (size), lmbda, ReE= 16    -8    -30.860883828573765
M (size), lmbda, ReE= 16    -4    -9.294986176824679
M (size), lmbda, ReE= 16    -2    -2.798532882387297
M (size), lmbda, ReE= 24    -1024    -51290.69884814223
M (size), lmbda, ReE= 24    -512    -21071.009025313422
M (size), lmbda, ReE= 24    -256    -7609.784437060723
M (size), lmbda, ReE= 24    -128    -2425.1554494037678
M (size), lmbda, ReE= 24    -64    -717.9975556595776
M (size), lmbda, ReE= 24    -32    -202.00811243397402
M (size), lmbda, ReE= 24    -16    -53.92762488480991
M (size), lmbda, ReE= 24    -8    -14.041875630397607
M (size), lmbda, ReE= 24    -4    -4.413758333617814
M (size), lmbda, ReE= 24    -2    -1.6542753342180623
(base) jakub@Artificial777:~$
```

7 Podsumowanie i wnioski

Program miał za zadanie znalezienie wartości własnych równania Lippmanana-Schwingera. Używając kwadratury Gaussa program wylicza energie własne dla zadanych warunków początkowych.