UNIVERSIDAD DE LOS ANDES, DEPARTAMENTO DE FÍSICA
FÍSICA ATÓMICA

Átomo de hidrógeno

Juan Barbosa, 201325901
Marzo 07, 2017

$$\boxed{-\frac{\hbar^2}{2m}\nabla^2\Psi(x)+V(x)\Psi(x)=E\Psi(x)}$$

Para el átomo de hidrógeno la ecuación anterior se escribe usando coordenedas esféricas, cuyo potencial depende de la parte radial:

$$V(r)=\frac{ze^2}{4\pi\epsilon_0 r} \tag{1}$$

Usando separación de variable $\Psi(r,\theta,\phi)=R(r)\Theta(\theta)\Phi(\phi)$, la ecuación diferencial para R se escribe:

$$\frac{d^2R}{dr^2}+\frac{2}{r}\frac{dR}{dr}+\frac{2\mu}{\hbar^2}\left(E-\frac{ze^2}{4\pi\epsilon_0 r}\right)R-\frac{l(l+1)}{r^2}R=0 \qquad \text{donde} \qquad l\in\mathbb{Z} \tag{2}$$

Haciendo un cambio de variable $r=\frac{a_0}{z}u$ se puede reescribir la ecuación anterior como:

$$\frac{d^2R}{du^2}\left(\frac{z}{a_0}\right)^2+\frac{2}{u}\frac{dR}{du}\left(\frac{z}{a_0}\right)^2+\frac{2\mu}{\hbar^2}\left(E-\frac{ze^2}{4\pi\epsilon_0 u}\frac{z}{a_0}\right)R-\frac{l(l+1)}{u^2}R\left(\frac{z}{a_0}\right)^2=0 \tag{3}$$

Usando como factor común z^2/a_0^2:

$$\frac{d^2R}{du^2}+\frac{2}{u}\frac{dR}{du}+\frac{2\mu a_0^2}{\hbar^2 z^2}\left(E-\frac{z^2e^2}{4\pi\epsilon_0 u a_0}\right)R-\frac{l(l+1)}{u^2}R=0 \tag{4}$$

Asumiendo $2\mu a_0^2/(\hbar^2 z^2)=1/E_0$

$$\frac{d^2R}{du^2}+\frac{2}{u}\frac{dR}{du}+\left(\frac{E}{E_0}-\frac{z^2e^2}{4\pi\epsilon_0 u a_0 E_0}\right)R-\frac{l(l+1)}{u^2}R=\frac{d^2R}{du^2}+\frac{2}{u}\frac{dR}{du}+\left(\epsilon-\frac{2}{u}\right)R-\frac{l(l+1)}{u^2}R=0 \tag{5}$$

$$\frac{d^2R}{du^2}+\frac{2}{u}\frac{dR}{du}+\left(\epsilon-\frac{2}{u}-\frac{l(l+1)}{u^2}\right)R=0 \tag{6}$$

Usando el método de Euler para resolver numéricamente se obtiene:

$$\dot{R}_n = \dot{R}_{n-1} + \ddot{R}_{n-1}\Delta u \qquad \text{donde} \qquad \ddot{R}_{n-1} = -\frac{2}{u}\dot{R}_{n-1} - \left(\epsilon - \frac{2}{u} - \frac{l(l+1)}{u^2}\right)R_{n-1}$$
$$R_n = R_{n-1} + \dot{R}_n \Delta u \tag{7}$$

El sistema de ecuaciones diferenciales es resuelto con $N = 1000$ puntos, $dx = 0{,}1$, para $l = 0, 1, 2, 3$. Para $l = 0$ se usa como condiciones iniciales $R'(0) = -1$ y $R(0) = 1$, para todas las demas se usa $R'(0) = 1$ y $R(0) = 0$. El algoritmo funciona de forma analoga a el método de bisección para encontrar raíces.

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

def solver(y, u, l, epsilon, du):
    L = l*(l+1)
    prime = np.zeros_like(u)
    x = np.ones_like(u)
    prime[0], x[0] = y
    for i in range(N-1):
        derivative = - 2*prime[i]/u[i] - (epsilon + 2/u[i] - L/u[i]**2)*x[i]
        prime[i+1] = prime[i] + derivative*du
        x[i+1] = x[i] + prime[i+1]*du
    return x

N = 1000
du = 100/N
U = np.linspace(du, N*du, N)
n = 6
nl = 4

energies = np.zeros((nl, n))
functions = np.zeros((nl, n, N))

for l in range(nl):
    if l == 0:
        e = -1.0
        y = [-1, 1]
    else:
        e = E[0]*1.1
        y = [10**(-l+1), 0]
    E = np.zeros(n)
    R = np.zeros((n, N))
    for i in range(n):
        de = abs(e)*0.01
        last = 0
        current = 10
        thresh = 0.09
        while abs(current) > thresh and e < 0 and de > 1e-9:
            e += de
            r = solver(y, U, l, e, du)
            current = r[-1]
            if current*last < 0:
                e += -de
                de *= 0.1
            last = current
        E[i] = e
        R[i] = r
        e += 20*de
    energies[l] = E
    functions[l] = R

fix, axes = plt.subplots(2, 2, sharex=True, figsize=(12, 6))
axes = axes.reshape(4)
for l in range(nl):
    y = functions[l]
    for i in range(n):
```

```
        axes[l].plot(U, y[i], label="$\epsilon=%f$"%energies[l, i])
    if l == 0 or l == 2:
        axes[l].set_ylabel("$R(U)$")
    if l > 1:
        axes[l].set_xlabel("$U$")
    axes[l].set_title("$l=%d$"%l)
    axes[l].set_ylim(min(y[0]), max(y[1]))
    axes[l].legend(fontsize=8)
plt.savefig("R.pdf")

n_n = np.arange(2, n+2)
E_T = -1/n**2

L = np.zeros((nl, n+2))
for l in range(1, 3):
    L[l+1, l:n+l] = energies[l+l]

L[0, :n] = energies[0]
L[1, :n] = energies[1]
L[2, 1:n+1] = energies[2]
L[3, 2:n+2] = energies[3]
for i in range(n+2):
    values = L[:, i]
    values = [str(item) if item != 0 else "" for item in values]
    text = " & ".join(values)
    print(r"%d & %f & %s \\"%(i+2, -1/(i+2)**2,text))

temps = np.arange(0, 400)
thresh = [0.01, 0.01, 0.25, 5]
fix, axes = plt.subplots(2, 2, sharex=True, figsize=(12, 6))
axes = axes.reshape(4)
for l in range(nl):
    rl = functions[l]
    th = thresh[l]
    for i in range(n):
        r = rl[i]
        pos = np.where(abs(r[400:]) < th)[0] + 400
        pos = np.concatenate((temps, pos))
        u = U[pos]
        r = r[pos]
        P = 4*np.pi*(u*r)**2
        P *= 1/np.trapz(P)
        axes[l].plot(u, P, label="$\epsilon=%f$"%energies[l, i])
    if l == 0 or l == 2:
        axes[l].set_ylabel("$P(U)/\int P(u)dU$")
    if l > 1:
        axes[l].set_xlabel("$U$")
    axes[l].set_title("$l=%d$"%l)
    axes[l].legend(fontsize=8)
plt.savefig("P.pdf")
```

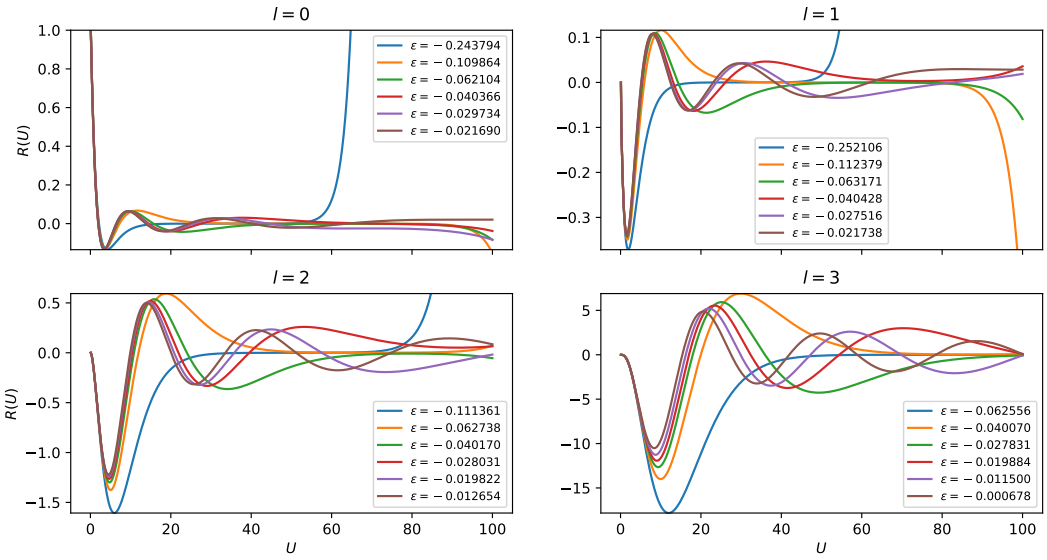

Figura 1: Primeros 6 valores de energía para $l = 0, 1, 2, 3$.

Cuadro 1: Energías obtenidas para los distintos valores de l ordenadas en niveles n, el valor teórico esperado corresponde con ϵ_T.

n	ϵ_T	$\epsilon_{l=0}$	$\epsilon_{l=1}$	$\epsilon_{l=2}$	$\epsilon_{l=3}$
2	-0.250000	-0.243793716	-0.252106231851		
3	-0.111111	-0.109863584471	-0.112379400942	-0.111360953653	
4	-0.062500	-0.0621041237279	-0.0631713802996	-0.0627381968616	-0.0625562074569
5	-0.040000	-0.0403662521966	-0.0404282449355	-0.0401698413622	-0.0400696123192
6	-0.027778	-0.0297344023873	-0.0275164742092	-0.028031058596	-0.0278314836864
7	-0.020408	-0.0216904640572	-0.0217380146253	-0.0198219628643	-0.0198838400997
8	-0.015625			-0.0126540207376	-0.0115004356954
9	-0.012346				-0.000678129139279

En la Figura 1 se muestra las funciones R para las cuales se considera convergencia existe un cambio en el signo de la pendiente entre ellas y un $d\epsilon$. Las funciones son simuladas de $u = 0$ hasta $u = 100$. En el Tabla 1 se muestran los primeros seis valores de energía obtenidos para distintos valores de l. Se observa que los valores de l son estrictamente menores a n. Adicionalmente se obtiene aproximadamente la misma energía para un mismo n y distinto l.

En la Figura 2 se muestran las funciones de probabilidad para las distintas funciones $R_{l,n}(u)$. La probabilidad se encuentra normalizada, esto es que:

$$P = \frac{P}{\int P du} \tag{8}$$

En ellas se observa que la probabilidad de encontrar el electrón a distancias grandes aumenta con la energía del mismo. También se puede ver que el máximo existe para el valor mínimo de n que puede soportar l, es decir $n = l - 1$.

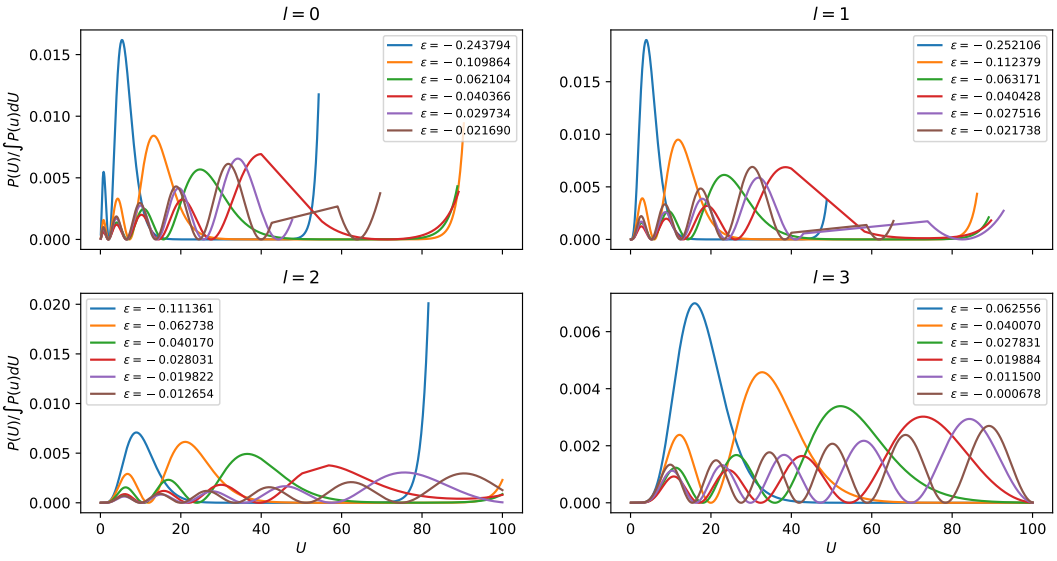

Figura 2: Función normalizada de probabilidad. Para distintos valores de ϵ y l.