Física atómica: Tarea 3

Iván Mauricio Burbano Aldana

February 12, 2017

Para poder resolver el problema en cuestión se utilizó el siguiente código:

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

plt.rc("text", usetex = True)

x = np.linspace(0, 3, 300 + 1)
U0 = 100
res = 100
E_ligado = np.linspace (0, 100, res)
inicial = 1
criterio = 0.014
Epar = []
Eimpar = []

def U(x):
        if(x <= 1):
                return 0
        else:
                return U0

def psi(x, U0, E, paridad, inicial):
        if (paridad == 0):
                phi = [inicial]
                Dphi = [0]
        else:
                phi = [0]
                Dphi = [inicial]
        for n in range(1,len(x)):
                        Dphi.append(Dphi[n - 1] + ((U(x[n
                            - 1]) - E) * phi[n - 1] * (x[
                           n] - x[n - 1])))
                        phi.append(phi[n - 1] + (Dphi[n -
                             1] * (x[n] - x[n - 1])))
```

```
        return [phi, Dphi]

def buscador(E_iniciales, paridad):
        E = []
        for n in range(0, len(E_iniciales) - 1):
                Dsol1 =  psi(x, U0, E_iniciales[n],
                    paridad, inicial)[1]
                Dsol2 =  psi(x, U0, E_iniciales[n + 1],
                    paridad, inicial)[1]
                if(np.sign(Dsol1[300]) != np.sign(Dsol2
                    [300])):
                        E.append(E_iniciales[n])
                        E.append(E_iniciales[n + 1])
        return E

Epar = buscador(E_ligado, 0)
Eimpar = buscador(E_ligado, 1)

for n in range(0, len(Epar[::2])):
        valor = criterio + 1
        while (abs(valor) > criterio):
                Energias = np.linspace(Epar[2*n], Epar[2*
                    n + 1], res)
                [Epar[2*n], Epar[2*n + 1]] = buscador(
                    Energias, 0)

for n in range(0, len(Eimpar[::2])):
        valor = criterio + 1
        while (abs(valor) > criterio):
                Energias = np.linspace(Eimpar[2*n],
                    Eimpar[2*n + 1], res)
                [Eimpar[2*n], Eimpar[2*n + 1]] = buscador
                    (Energias, 1)
                valor = psi(x, U0, (Eimpar[2*n] + Eimpar
                    [2*n + 1])/2, 1, inicial)[0][300]
                print valor

plt.figure()
plt.title(r"Comportamiento en estados ligados de part\'
    iculas pares")
for n in range(0, len(Epar) - 1):
        if ((2*n + 1) < len(Epar)):
                E = (Epar[2*n + 1] + Epar[2*n])/2
                phi = psi(x, U0, E, 0, inicial)[0]
                plt.plot(x, phi, label = r"$%f \leq E \
                    leq %f$" % (Epar[2*n], Epar[2*n + 1]))
```

```
        else:
                break
plt.ylabel(r"$\psi$", fontsize = 15)
plt.xlabel(r"$x$", fontsize = 15)
plt.legend()
plt.savefig("comportamiento_ligado_par.pdf", format = "
    pdf")

plt.figure()
plt.title(r"Comportamiento en estados ligados de part\'
    iculas impares")
for n in range(0, len(Eimpar) - 1):
        if ((2*n + 1) < len(Eimpar)):
                E = (Eimpar[2*n + 1] + Eimpar[2*n])/2
                phi = psi(x, U0, E, 1, inicial)[0]
                plt.plot(x, phi, label = r"$%f \leq E \
                    leq %f$" % (Eimpar[2*n], Eimpar[2*n +
                    1]))
        else:
                break
plt.ylabel(r"$\psi$", fontsize = 15)
plt.xlabel(r"$x$", fontsize = 15)
plt.legend()
plt.savefig("comportamiento_ligado_impar.pdf", format = "
    pdf")

f, axarr = plt.subplots(2, 2)
axarr[0, 0].set_title(r"$E = 100$")
axarr[0, 0].plot(x, psi(x, U0, 100, 0, inicial)[0], label
    = "par")
axarr[0, 0].plot(x, psi(x, U0, 100, 1, inicial)[0], label
    = "impar")
axarr[0, 0].legend()
axarr[0, 0].set_xlabel(r"Posici\'on")
axarr[0, 0].set_ylabel(r"Funci\'on de onda")
axarr[0, 1].set_title(r"$E = 200$")
axarr[0, 1].plot(x, psi(x, U0, 200, 0, inicial)[0], label
    = "par")
axarr[0, 1].plot(x, psi(x, U0, 200, 1, inicial)[0], label
    = "impar")
axarr[0, 1].set_xlabel(r"Posici\'on")
axarr[0, 1].set_ylabel(r"Funci\'on de onda")
axarr[1, 0].set_title(r"$E = 700$")
axarr[1, 0].plot(x, psi(x, U0, 700, 0, inicial)[0], label
    = "par")
axarr[1, 0].plot(x, psi(x, U0, 700, 1, inicial)[0], label
```

```
    = "impar")
axarr[1, 0].set_xlabel(r"Posici\'on")
axarr[1, 0].set_ylabel(r"Funci\'on de onda")
axarr[1, 1].set_title(r"$E = 1000$")
axarr[1, 1].plot(x, psi(x, U0, 1000, 0, inicial)[0],
    label = "par")
axarr[1, 1].plot(x, psi(x, U0, 1000, 1, inicial)[0],
    label = "impar")
axarr[1, 1].set_xlabel(r"Posici\'on")
axarr[1, 1].set_ylabel(r"Funci\'on de onda")
f.tight_layout()
plt.savefig("comportamiento_libre.pdf")
```

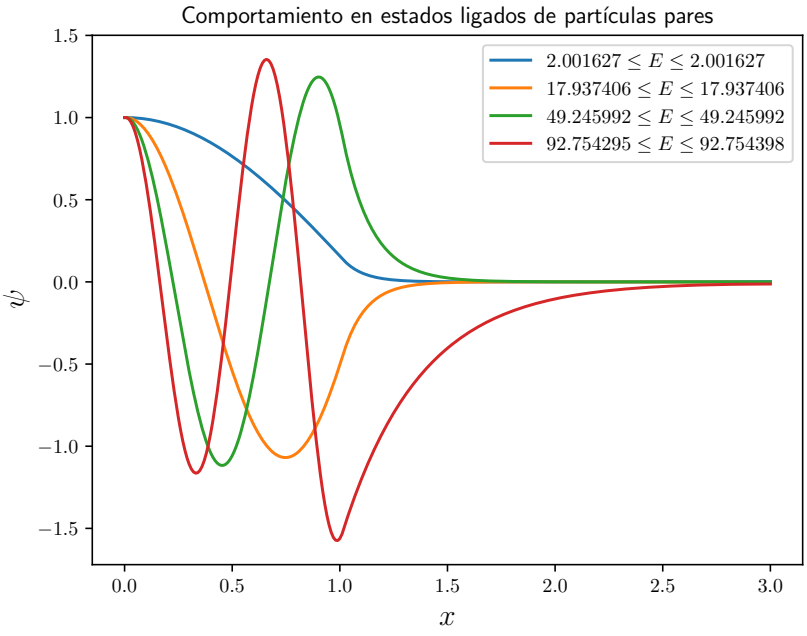
La tecnica consistió en hace un recorrido de "res" valores de energía entre 0 y 100 (el rango de energias en el que sabemos que se encuentra una partícula ligada) para hacer una primera identificación de valores tentativos. Al principio fue bastante difícil encontrar algún valor. Sin embargo, una vez se encontró que alrededor de 2 había uno, se notó que cuando se pasa por él, el signo de la derivada en el valor $x = 3$ cambiaba. En efecto, de un lado del valor que entregaba una función de onda aceptable la función crecía en 3 mientras que en el otro la función decrecía. Por lo tanto se creó la función "buscador" que toma una lista de valores de energía y los compara en cadena identificando cuando cambia la derivada de signo. Las dos energías entre las cuales hay tal cambio se guardan en una lista de manera que sabemos que entre ambas energías se encuentra una de las permitidas por la teoría cuántica.

Una vez se identifican valores tentativos (o mejor dicho, intervalos tentativos) para la energía (se encontraron 4 para soluciones pares y 3 para soluciones impares) se repite el algoritmo solo que ahora se hace un recorrido sobre cada uno de estos intervalos subdividiendolos una vez más con "res". De esta manera se cierra el intervalo en el cual pueden estar la energía buscada. Esto se hace de manera iterativa hasta que el valor absoluto de la función de onda en $x = 3$ es menor a un "criterio" que se determina al comienzo del código. Una vez se cumple esto, se grafican las soluciones con la energías promedio de cada intervalo.

Los resultados fueron bastante satisfactorios. El código permite hallar los valores de energía con una presición de hasta 7 cifras significativas en segundos. Ademas, se puede apreciar el comportamiento oscilatorio (de frecuencia constante para aquellas con más de un pico) para $x \leq 1$ seguido de un decaimiento exponencial. Sin embargo, el metodo de integración usado (regla de Euler) puede mejorarse pues las ondas obtenidas con varios picos no tienen amplitud constante. En resumen, lo que hace a este código tan fuerte es el algoritmo "buscador" y su implementación tanto en la busqueda de valores tentativos de energía como en el depuramiento de estos.

Por completitud se experimentó tambien con valores de energía mayores a 100.

Se puede observar un comportamiento sinusoidal como es esperado. En par-



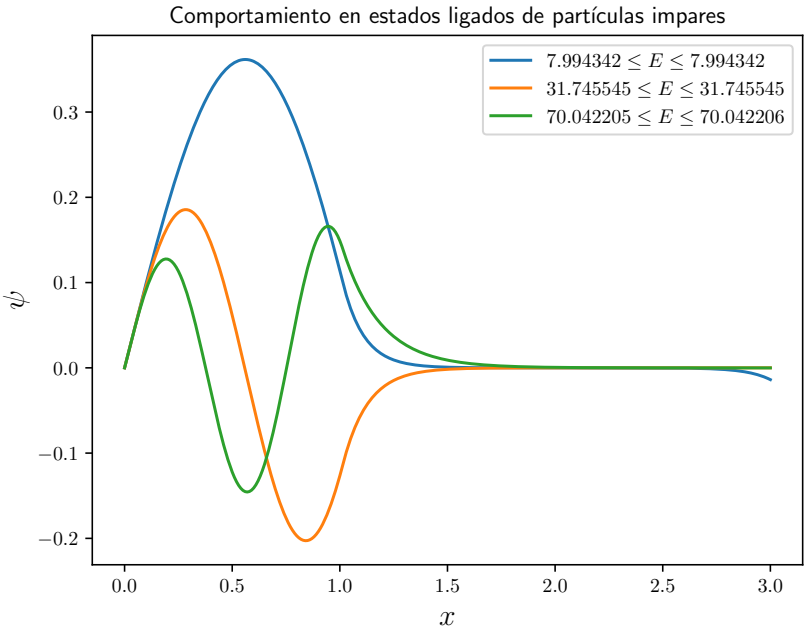


Figure 1: Las funciones de onda junto con sus energías halladas mediante el código diseñado

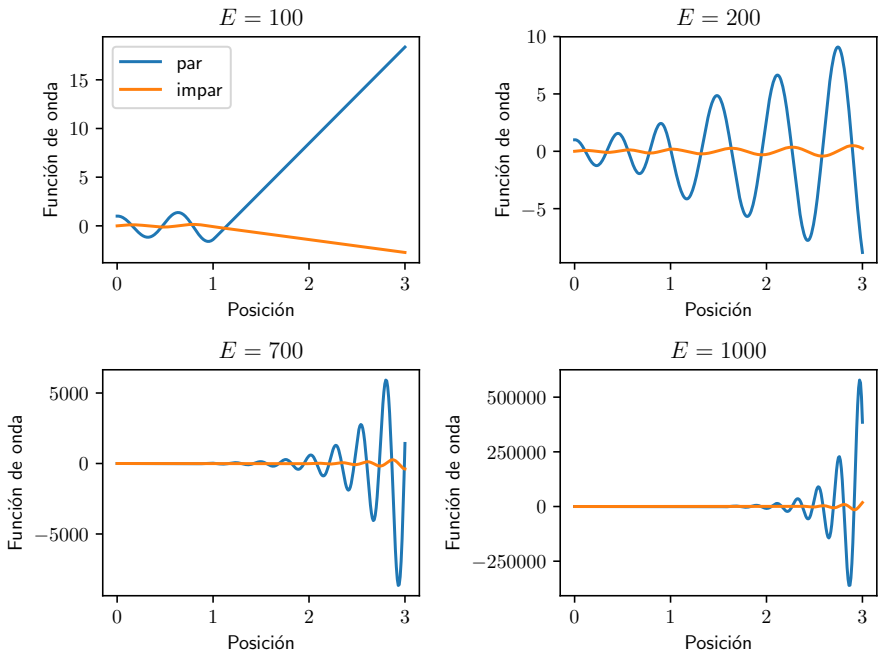


Figure 2: Funciones de onda libres

ticular, para las de menor energía, se puede observar el cambio en la frecuencia de la onda al salir del pozo de potencial. Notese que estas funciones no son cuadrado integrables y por lo tanto las energías que les corresponden no son valores propios del hamiltoniano. Este es un ejemplo donde se evidencia que el espectro de un operador (en este caso el hamiltoniano) no necesariamente coincide con sus valores propios.