UNIVERSIDAD DE LOS ANDES, DEPARTAMENTO DE FÍSICA
FÍSICA ATÓMICA

Pozo finito de potencial

Juan Barbosa, 201325901
Febrero 16, 2017

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

Para el caso del potencial finito la ecuación de Schrödinger se aplica a las tres regiones de interes. Para las regiones con potencial $V_0 > E$ la solución a la función de onda Ψ será:

$$\Psi_I(x) = Ae^{kx} + Be^{-kx} \qquad , \qquad k = \sqrt{\frac{2m(V_0-E)}{\hbar^2}} \tag{1}$$

Teniendo en cuenta que la función debe ser acotada para la región I la función $\Psi(x)$ debe anularse para $x = -\infty$, lo anterior implica que $B = 0$. Para la región III se usa un argumento análogo con lo cual se obtiene:

$$\begin{aligned}\Psi_I(x) &= Ae^{kx}\\ \Psi_{III}(x) &= Ce^{-kx}\end{aligned} \tag{2}$$

En la región II la solución es oscilante.

$$\Psi_{II}(x) = Fe^{i\kappa x} + Ge^{-i\kappa x} \qquad , \qquad \kappa = \sqrt{\frac{2mE}{\hbar^2}} \tag{3}$$

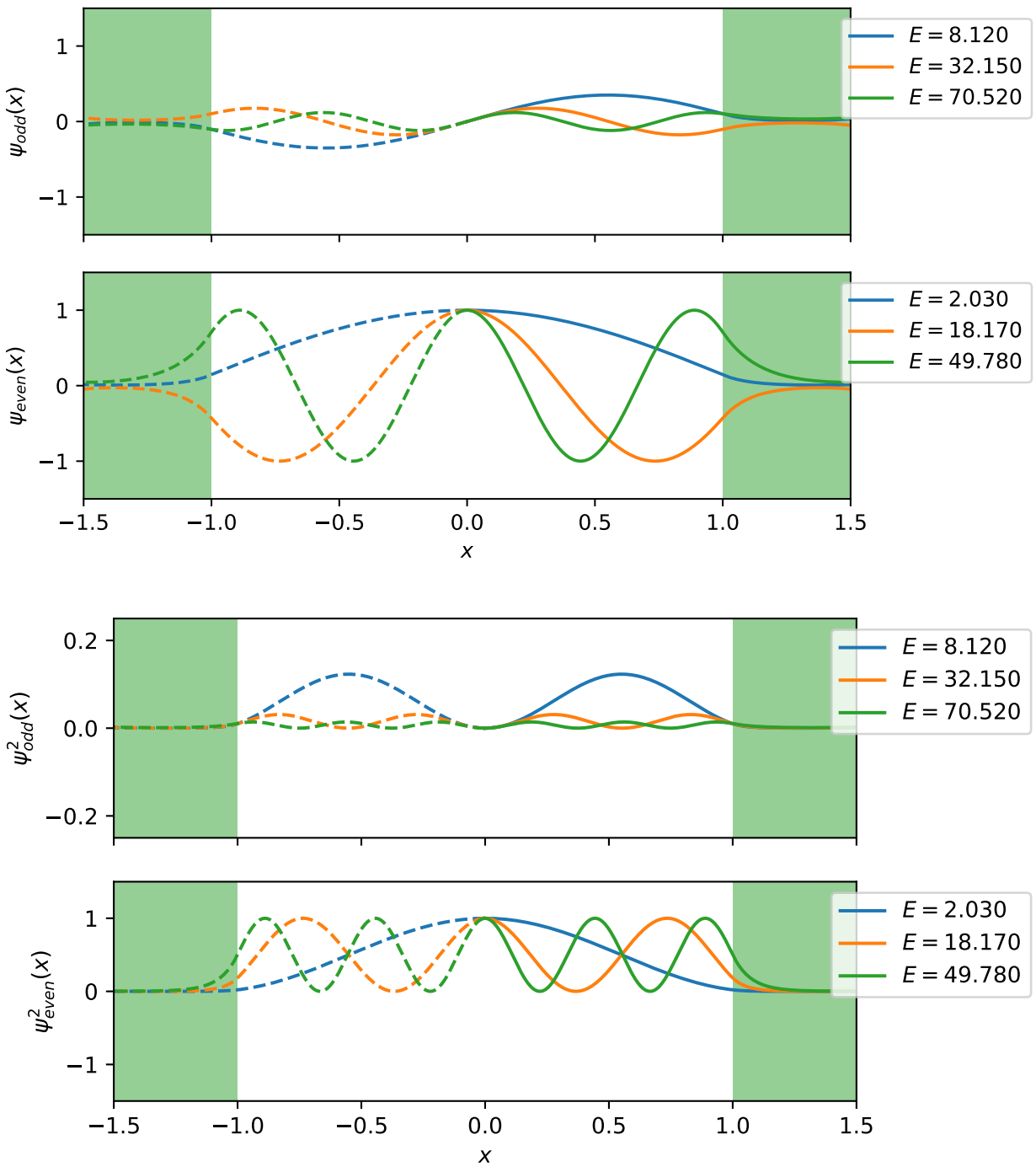
Usando el método de Euler para resolver numéricamente se obtiene:

$$\begin{aligned}\dot{\Psi}_n &= \dot{\Psi}_{n-1} + \ddot{\Psi}_{n-1}\Delta x\\ \Psi_n &= \Psi_{n-1} + \dot{\Psi}_n\Delta x\end{aligned} \qquad \text{donde} \qquad \ddot{\Psi} = \frac{2m(V_0-E)}{\hbar^2}\Psi \tag{4}$$

El sistema de ecuaciones diferenciales es resuelto usando $N = 1001$ puntos, $dx = 0.003$, $L = 2$ y un potencial $V_0 = 100$. Las unidades usadas son arbitrarias tales que $\hbar^2 = 2m = 1$.

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

def potential(x):
    U = np.zeros_like(x)
    pos = np.where(x > 1)
    U[pos] = U0
    return U

def equation(U, E, f):
    return (U - E)*f

def solver(psi, psi_prime, E):
    for i in range(1,N):
        psi_prime[i] = psi_prime[i-1] + equation(U[i-1], E, psi[i-1])*dx
        psi[i] = psi[i-1] + psi_prime[i]*dx
    return psi

def seeker(odd):
    functions = []
    energies = []
    E = 1
    value = 0.05
    for i in range(3):
        psi = np.ones(N)
        psi_prime = np.zeros(N)
        check = 1
        if odd:
            psi[0] = 0
            psi_prime[0] = 1
        if E > U0:
            break
        while check > value and E < U0:
            E += 0.01
            psi = solver(psi, psi_prime, E)
            check = abs(psi[500])
        if check < value:
            functions.append(psi)
            energies.append(E)
            E += 6
    return functions, energies

U0 = 100
N = 1001
dx = 0.003
x = np.linspace(0, (N-1)*dx, N)
U = potential(x)

odd, E_odd = seeker(True)
even, E_even = seeker(False)

fig, axes = plt.subplots(2, sharex=True)
axes[0].set_ylabel("$\psi_{odd}(x)$")
axes[1].set_ylabel("$\psi_{even}(x)$")
axes[1].set_xlabel("$x$")
for i in range(3):
    parent = axes[0].plot(x, odd[i], label = "$E = %.3f$"%E_odd[i])[0]
    axes[0].plot(-x, -odd[i], "--", c=parent.get_color())
    parent = axes[1].plot(x, even[i], label = "$E = %.3f$"%E_even[i])[0]
    axes[1].plot(-x, even[i], "--", c=parent.get_color())
for ax in axes:
    ax.set_xlim(-1.5, 1.5)
    ax.set_ylim(-1.5, 1.5)
    ax.axvspan(-1.0, -1.5, facecolor='#2ca02c', alpha=0.5)
    ax.axvspan(1.0, 1.5, facecolor='#2ca02c', alpha=0.5)
    extra = ax.legend(loc=1, bbox_to_anchor=(1.25, 1.0))
fig.savefig("finite-box.pdf", bbox_extra_artists=(extra,), bbox_inches='tight')
```

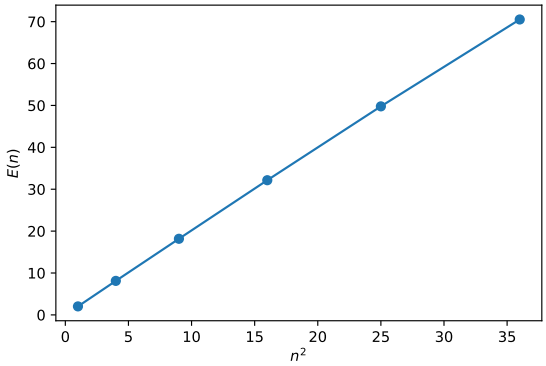
```
fig, axes = plt.subplots(2, sharex=True)
axes[0].set_ylabel("$\psi^2_{odd}(x)$")
axes[1].set_ylabel("$\psi^2_{even}(x)$")
axes[1].set_xlabel("$x$")
for i in range(3):
    parent = axes[0].plot(x, odd[i]**2, label = "$E␣=␣%.3f$"%E_odd[i])[0]
    axes[0].plot(-x, odd[i]**2, "--", c=parent.get_color())
    parent = axes[1].plot(x, even[i]**2, label = "$E␣=␣%.3f$"%E_even[i])[0]
    axes[1].plot(-x, even[i]**2, "--", c=parent.get_color())
for ax in axes:
    ax.set_xlim(-1.5, 1.5)
    ax.set_ylim(-1.5, 1.5)
    ax.axvspan(-1.0, -1.5, facecolor='#2ca02c', alpha=0.5)
    ax.axvspan(1.0, 1.5, facecolor='#2ca02c', alpha=0.5)
    extra = ax.legend(loc=1, bbox_to_anchor=(1.25, 1.0))
axes[0].set_ylim(-0.25, 0.25)
fig.savefig("psi-squared.pdf", bbox_extra_artists=(extra,), bbox_inches='tight')

E = sorted(E_odd + E_even)
n = np.arange(1, 1 + len(E))**2
plt.plot(n, E, "-o")
plt.ylabel("$E(n)$")
plt.xlabel("$n^2$")
plt.savefig("energy.pdf")

E = 110
psi = np.ones(N)
psi_prime = np.zeros(N)
even = solver(psi, psi_prime, E)
psi = np.zeros(N)
psi_prime = np.ones(N)
odd = solver(psi, psi_prime, E)

fig, axes = plt.subplots(2, sharex=True)
axes[0].set_ylabel("$\psi_{odd}(x)$")
axes[1].set_ylabel("$\psi_{even}(x)$")
axes[1].set_xlabel("$x$")
for i in range(3):
    axes[0].plot(x, odd, c="b")
    axes[0].plot(-x, -odd, "--", c="b")
    axes[1].plot(x, even, c="b")
    axes[1].plot(-x, even, "--", c="b")
for ax in axes:
    ax.axvspan(-1.0, -3, facecolor='#2ca02c', alpha=0.5)
    ax.axvspan(1.0, 3, facecolor='#2ca02c', alpha=0.5)
fig.savefig("greater.pdf", bbox_extra_artists=(extra,), bbox_inches='tight')
```



La energía depende del cuadrado de n como número cuántico. Resultado análogo al de un pozo de potencial infinito. La solución analítica a la ecuación es de la forma:

$$\begin{aligned} k_{even} &= \kappa \tan\frac{\kappa L}{2} = \sqrt{\beta^2 - \kappa^2} \\ k_{odd} &= -\frac{\kappa}{\tan\frac{\kappa L}{2}} = \sqrt{\beta^2 - \kappa^2} \end{aligned} \tag{5}$$

$$\begin{aligned} \Psi_{II_{even}}(x) &= F\cos(\kappa x) \\ \Psi_{II_{odd}}(x) &= F\sin(\kappa x) \end{aligned} \tag{6}$$



n	E
1	2.030
2	8.120
3	18.170
4	32.150
5	49.780
6	70.520

Finalmente para el caso con $E > V_0$ se obtienen soluciones oscilantes para toda región del espacio. Para las regiones con $V = V_0$ la amplitud se ve reforzada.