IMPLEMENTACIÓN DEL MÉTODO DE SHOOTING PARA RESOLVER LA ECUACIÓN DE SCHRÖDINGER: TEORÍA Y APLICACIÓN

Pontificia Universidad Católica de Chile
Estudiante: Nicolás Ulloa Gatica
Profesor: Rafael Benguria Donoso

I. Resumen

El método de shooting (o *disparo*) es un método numérico que permite resolver un problema de valores de frontera (PVF) al transformarlo en un problema de valores iniciales (PVI) equivalente.

Para transformar el problema, se supone una condición inicial para el PVI (*disparo*). Luego, se resuelve el PVI. Si la solución del PVI satisface las condiciones de frontera del PVF, corresponde a la solución para el PVF. Si la solución no satisface las condiciones de frontera del PVF, se realiza un proceso iterativo para obtener una nueva condición inicial al PVI a partir de la suposición original.

II. Método

A. Método de Shooting

Se tiene el siguiente PVF[1]:

$$\begin{aligned} u'' &= f(x, u, u') \\ u(a) &= A \\ u(b) &= B \end{aligned}$$

Con $a, b, A, B \in \mathbb{R}$. Primero, se debe transformar este problema a uno de valores iniciales. Para esto, se escoge algún valor inicial para la derivada:

$$u'(a) = \alpha, \quad \alpha \in \mathbb{R}$$

Con lo que se obtiene el siguiente PVI:

$$\begin{aligned} u'' &= f(x, u, u') \\ u(a) &= A \\ u'(a) &= \alpha \end{aligned}$$

Luego, se resuelve el PVI integrando desde $x = a$ hasta $x = b$, con lo que se obtiene $u_(b)$. Si este valor coincide con el falor de frontera correspondiente, se encontró la solución. De lo contrario, se debe realiza una nueva suposición para la condición inicial $u'(a)$, y resolver nuevamente el PVI. Este proceso se repite iterativamente hasta encontrar la solución correspondiente.

Adicionalmente, se utiliza alguna técnica o método numérico para que la iteración sea convergente a la solución real. A continuación se detalla el método de Newton-Raphson, que permite lograr este objetivo.

[1]Notación: $f' = \frac{df}{dx}$

B. Método de Newton-Raphson

Es un método numérico iterativo que permite encontrar las raíces de una función a partir aproximaciones iniciales.

Desde el punto de vista gráfico, el método consiste en:

1) Escoger un punto x que sea cercano a alguna raíz de la función $f(x)$ (*función objetivo*).
2) Obtener la recta tangente a la función en el punto escogido.
3) El punto de intersección entre la recta tangente y el eje horizontal estará más cerca del valor real que el punto escogido inicialmente. Así, se utiliza el punto encontrado para volver al paso 1 e iterar tantas veces como sea necesario.

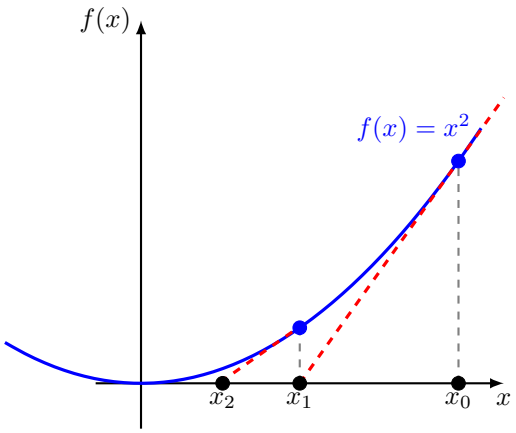

Figura 1: Representación gráfica del método Newton-Raphson

En la figura 1 se ilustra el método. Primero, se escoge un punto cercano a alguna raíz de la función f. Luego, se obtiene la recta tangente a f en el punto x_0. La intersección entre la recta tangente y el eje horizontal corresponderá al siguiente punto, x_1. Es decir,

$$f'(x_0) = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = -\frac{f(x_0)}{x_1 - x_0}$$

$$\Rightarrow$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Finalmente, si x_1 está lo suficientemente cerca de cero, se encontró la raíz. De lo contrario, se deberá repetir el mismo procedimiento utilizando x_1 como punto inicial.

Así, la aproximación obtenida de la enésima iteración está dada por,

$$\boxed{x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}}$$

Para obtener la derivada de la función f se pueden utilizar otros métodos númericos como el de diferencias finitas o Runge -Kutta. A continuación, se explica el método de diferencias finitas.

C. Diferencias Finitas

El método de diferencias finitas consiste en aproximar las derivadas de una función utilizando diferencias sobre un dominio discreto. Por definición, la derivada de la función f es:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

La aproximación se obtiene al tomar un h muy cercano a cero:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}, \qquad h << 1 \tag{1}$$

Además, se tendrá una función y dominio discreto con dos condiciones iniciales para la función f. Por tanto, $x \in [a,b] \mid a,b \in \mathbb{R}$. De igual forma, $f(x_0) = f_0, \;\; f(x_1) = f_1 \mid f_0, f_1 \in \mathbb{R}$.

Luego, se toman intervalos regulares de medida h para definir los puntos que pertenecen al dominio. Esto se ilustra en la figura 2.

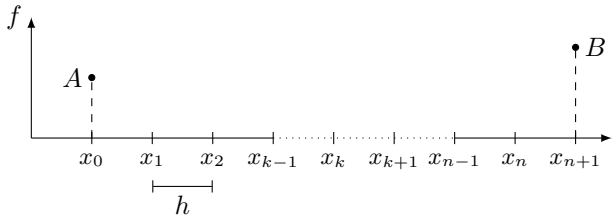

Figura 2: Discretización del intervalo $[a,b]$ en puntos equidistantes.

Así, con $k \in \{0,1,...,n+1\}$ donde $n \in \mathbb{N}$, se tiene que $x_k = k \cdot h$. Por tanto, $x_k + h = x_{k+1}$. Luego, la ecuación (1) se puede reescribir como:

$$f'(x_k) \approx \frac{f(x_{k+1}) - f(x_k)}{h} \tag{2}$$

De aquí en adelante, para explicitar que las funciones son discretas y facilitar la lectura del algoritmo, se utilizará la siguiente notación:

$$f[k] \equiv f(x_k)$$

Por tanto, si $f'[k]$ es la aproximación para $f'(x_k)$ dada por la ecuación (2), se tendrá que:

$$f'[k] = \frac{f[k+1] - f[k]}{h} \tag{3}$$

De igual forma, para la segunda derivada se tendrá:

$$\begin{aligned} f''[k] &= \frac{f'[k+1] - f'[k]}{h} \\ &= \frac{1}{h^2}\left(f[k+2] - f[k+1] + f[k] - f[k+1]\right) \\ &= \frac{1}{h^2}\left(f[k+2] - 2f[k+1] + f[k]\right) \end{aligned}$$

Es decir,

$$f''[k] = \frac{1}{h^2}\left(f[k+2] - 2f[k+1] + f[k]\right) \tag{4}$$

D. Aplicación del método

Se quiere resolver la ecuación (simplificada) de Schrödinger, dada por:

$$-u'' + Vu = Eu \tag{5}$$

Típicamente, esta ecuación estará sujeta a un problema de valores de frontera[2]:

$$\begin{aligned} u'' &= f(E,u,u') \\ u(a) &= A \\ u(b) &= B \end{aligned}$$

Con $f(E,u,u') = (V-E)u$. Luego, este problema se transforma a uno de valores iniciales. Para esto, se escoge algún valor inicial para la derivada:

$$u'(a) = \alpha, \quad \alpha \in \mathbb{R}$$

Con lo que se obtiene el siguiente PVI:

$$\begin{aligned} u'' &= f(E,u,u') \\ u(a) &= A \\ u'(a) &= \alpha \end{aligned}$$

Ahora, se quiere resolver este problema y encontrar una solución que dependerá de E. Esto se hace integrando desde $x = a$ hasta $x = b$ para algún E cualquiera (*disparo*). Expresando la derivada en términos de diferencias finitas, la ecuación (5) queda reescrita como,

$$-\frac{1}{h^2}\left(u[k+2] - 2u[k+1] + u[k]\right) + Vu[k] = Eu[k] \tag{6}$$

Con $k = 0,1,2,n-1$ y n la cantidad de puntos para realizar la aproximación. Dado que se tienen dos condiciones iniciales, se conocen los primeros dos valores de la función (o mejor dicho, vector) u. El primer valor de u será[3],

$$u[0] = A$$

Mientras que para el segundo valor de u se tiene que,

$$\begin{aligned} u'[0] = \alpha &= \frac{u[k+1] - u[k]}{h} \\ &\Rightarrow \\ u[1] &= \alpha h + u[0] = \alpha h + A \end{aligned}$$

[2]Notar que la variable independient es E (energía) y no x

[3]Esto es, por supuesto, tomando a como punto de partida ($a = 0$)

Luego, de la ecuación (6) se sigue que:

$$u[k+2] = u[k]\left((V-E)h^2 - 1\right) + 2u[k+1]$$

Con esta relación de recurrencia se pueden obtener todos los valores para u. Estos valores también dependerán del parámetro E, por lo que es más preciso denotar a la solución como u_E.

Luego, se debe revisar si el último valor de la solución encontrada, $u_E(b)$, satisface la condición de borde del PVF para algún E. Los valores de E para los que la condición de frontera se satisface corresponden a valores propios y su función respectiva, u_E, a la solución. Así, encontrar los valores propios equivale a encontrar las raíces de la siguiente función

$$f(E) = u_E(b) - B \tag{7}$$

Esto se hace mediante el método de Newton - Rhapson, aplicado directamente a la función $f(E)$.

III. Algoritmo

A continuación se muestran dos algoritmos. El algoritmo 1 se encarga de obtener la función objetivo $f[k]$, de forma discreta a partir de la ecuación de Schrödinger. El algoritmo 2 se encarga de aplicar el método de Newton - Rhapson a la función objetivo, con una aproximación inicial α. En la tabla I se encuentra la descripción de los parámetros utilizados en cada algoritmo y sugerencias para escogerlos.

Algoritmo 1 Cálculo de E

$V(x) \leftarrow \mathbf{v}(x)$
$h \leftarrow (x_{n+1} - x_0)/n$
$f \leftarrow [\ ]$
for all $j \in \{0, \ldots, n-1\}$ **do**
  for all $k \in \{0, \ldots, n-2\}$ **do**

$$u[k+2] = u[k]\left((V - j\cdot h)h^2 - 1\right) + 2u[k+1]$$

  end for
  $f[k] \leftarrow u[n]$
end for
return f

Algoritmo 2 Aplicación de Newton - Rhapson

input f, α
$h \leftarrow (x_{n+1} - x_0)/n$

$m \leftarrow \frac{f[\alpha]\cdot h}{f[\alpha+h]-f[\alpha]}$

while $f[\alpha]^2 > t^2$:
  $\alpha \leftarrow \alpha - m$
  $m \leftarrow \frac{f[\alpha]\cdot h}{f[\alpha+h]-f[\alpha]}$
return α

Parámetros	Descripción	Sugerencia
$\mathbf{v}(x)$	Función potencial	Comenzar con potencial nulo (partícula libre)
x_0, x_{n+1}	Primer y último valor de la preimagen, respectivamente	Comenzar con $x_0 = 0$ y $x_{n+1} > 0$
n	Cantidad de puntos con los que se realiza la aproximación	$n \geq 1000$
l	Constante de integración	$l = 1$
t	Tolerancia del error permitido para la aproximación	$t = 0.001$
α	Primera aproximación a la raíz de f	$\alpha = 1$
f	Secuencia discreta que representa a la función objetivo	Se obtiene con el algoritmo 1

Tabla I: Parámetros para algoritmo 1 y 2

IV. Código

A continuación se muestra un módulo de Python en el que se ejecutan los algoritmos presentado, con adecuaciones al lenguaje, y el output correspondiente.

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

# Parámetros
k = 100 # máximo valor en el dominio E
n = 2000
xa = 0
xb = 1
h = (xb - xa) / n
tolerancia = 0.0001
u0 = 0      # primera condición inicial
    (u)
du0 = 1     # segunda condición inicial
    (du/dx)

# Definición del potencial
def potencial(x):
    return 0

# Obtener función objetivo
```

```
def funcion_objetivo(E):
    u = [u0, du0 * h + u0]
    # Integrar discretamente hacia
       delante
    for i in range(0, n-1):
        u2 = 2 * u[i+1] - u[i] + (h **
            2) * (potencial(xa + i*h) -
            E) * u[i+1]
        u.append(u2)
    return u[-1] # Retornar último
       valor

# Obtener derivada de función objetivo
def derivar(funcion,x , h: int):
    # se utiliza diferencias finitas
    derivada = (funcion_objetivo(x + h)
        - funcion_objetivo(x))/h
    return derivada


# Obtención de función objetivo
dominio = np.linspace(0, k, n)
f = [funcion_objetivo(E) for E in
   dominio]

# Método de Newton-Rhapson

# Primera aproximación
x_aprox = float(input("Primera
   aproximación: "))

# Ajustar aproximación
m = funcion_objetivo(x_aprox)/derivar(
   funcion_objetivo,x_aprox,h)

while True:
    if np.abs(funcion_objetivo(x_aprox)
       )<tolerancia:
        break
    print(funcion_objetivo(x_aprox))
    m = funcion_objetivo(x_aprox)/
       derivar(funcion_objetivo,x_aprox
       ,h)
    x_aprox -= m

print("E = ", x_aprox)

#Graficar f(E)
plt.plot(dominio, f)
plt.axhline(0, color='red', linestyle='
   --')
plt.grid(True)
plt.title('Función Objetivo')
plt.xlabel('E')
plt.ylabel('f(E)')
plt.show()
```

A. *Output*

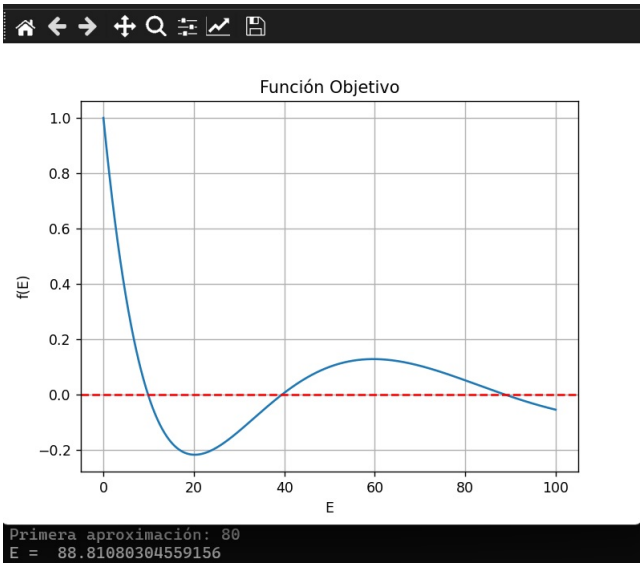

Figura 3: Resultado obtenido al ejecutar el código