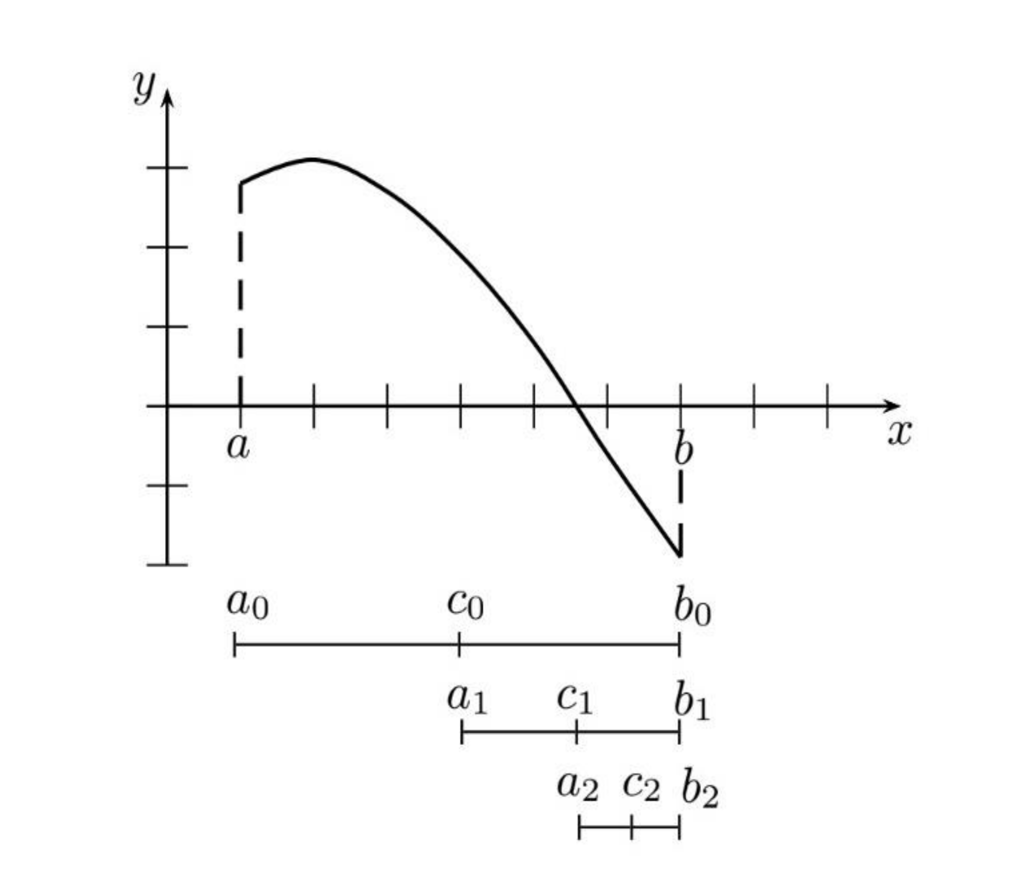
MODELADO Y SIMULACIÓN

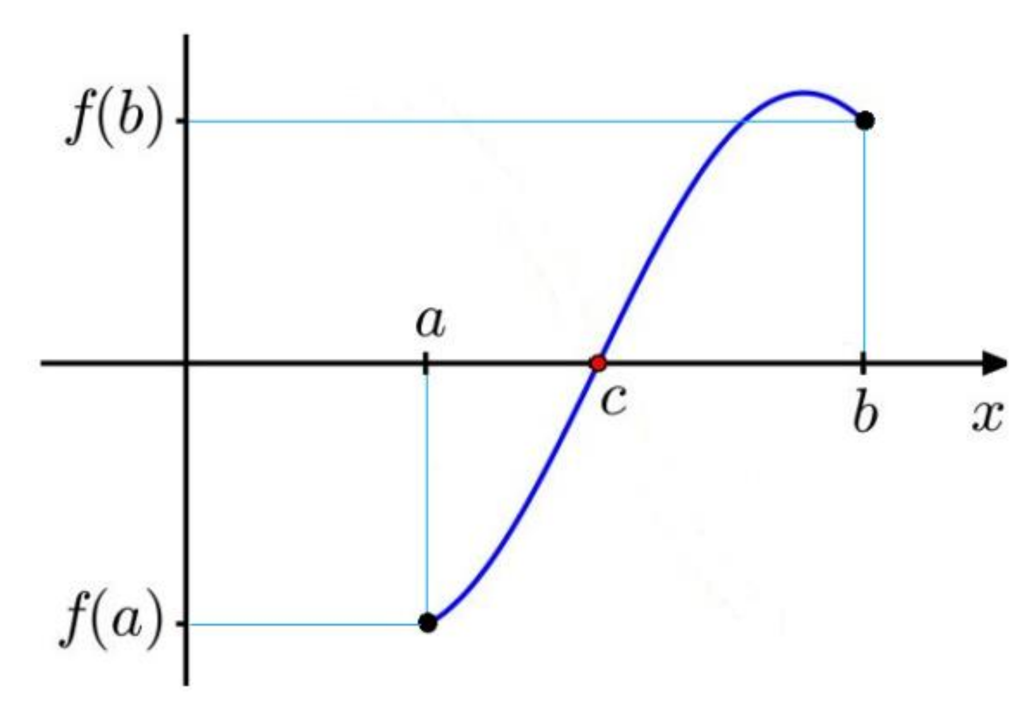
Búsqueda Binaria de Raíces

CLASE 1

Ing. Omar Cáceres



a búsqueda binaria de raíces, también conocida como el método de bisección, es un algoritmo numérico utilizado para encontrar una raíz de una función continua f(x) en un intervalo dado [a,b], donde f(a)y f(b) tienen signos opuestos f(a)·f(b)< 0. este método se basa en el teorema del valor intermedio, que garantiza la existencia de al menos una raíz en dicho intervalo.



Teorema de Bolzano:

sea f(x) una función continua en un intervalo cerrado [a,b]. si f(a)·f(b)<0, es decir, f(a) y f(b) tienen signos opuestos, entonces existe al menos un punto c∈(a,b) tal que: f(c)=0

Enunciado del Teorema del Valor Medio:

Sea f(x) una función que cumple las siguientes condiciones:
1.Es continua en un intervalo cerrado [a,b]
2.Es derivable en el intervalo abierto (a,b)
Entonces, existe al menos un punto c∈(a,b)) tal que:
f′(c)=(f(b)−f(a))/(b−a)

El algoritmo consiste en seguir los siguientes pasos:

1. Escoger el intervalo [a, b] donde f(a)·f(b)< 0, esto garantiza que halla una raíz en dicho intervalo.
2. Calcular el punto medio. $\boldsymbol{c} = \frac{a+b}{2}$
3. Evaluar c en la función: si $f_{(c)}$=0, entonces parar y hemos obtenido la raíz.
4. $Si\ f_{(a)}\ f_{(c)} < 0, entonces\ la\ raiz\ esta\ en\ [a,c]\ \boldsymbol{y}$ actualizamos $b = c$
5. $Si\ f_{(b)}\ f_{(c)} < 0, entonces\ la\ raiz\ esta\ en\ [c,b]\ actualizamos\ a = c$

El algoritmo consiste en seguir los siguientes pasos:

1. Escoger el intervalo [a, b] donde f(a)·f(b)< 0, esto garantiza que halla una raíz en dicho intervalo.
2. Calcular el punto medio. $\boldsymbol{c} = \frac{a+b}{2}$
3. Evaluar c en la función: si $f_{(c)}$=0, entonces parar y hemos obtenido la raíz.
4. $Si\ f_{(a)}\ f_{(c)} < 0, entonces\ la\ raiz\ esta\ en\ [a, c]\ \boldsymbol{y}$ actualizamos $b = c$
5. $Si\ f_{(b)}\ f_{(c)} < 0, entonces\ la\ raiz\ esta\ en\ [c, b]\ actualizamos\ a = c$

1. $f_{(x)} = x^3 - x - 2, \quad en\ el\ intervalo\ [1,2]$
2. $f_{(x)} = x^2 - 3, \quad en\ el\ intervalo\ [1,2]$
3. $f_{(x)} = e^x - 2 - x, \quad en\ el\ intervalo\ [1,2]$
4. $f_{(x)} = \cos x + 1 - x, \quad en\ el\ intervalo\ [1,2]$
5. $f_{(x)} = Ln(x) + x - 5, \quad en\ el\ intervalo\ [3,4]$
6. $f_{(x)} = \sqrt{x} - \cos x\,, \quad en\ el\ intervalo\ [0,1]$

Use el teorema de Bolzano par determinar si hay una raíz en los intervalos dados, en caso que si entonces aplique el método de búsqueda Binaria para aproximar la raíz.

Teorema de Bolzano

$$\boxed{\begin{array}{c} Sea\ f: [a,b] \rightarrow R, en\ una\ función\ continua, tal\ que\ f_{(a)}f_{(b)} < 0. \\ entonces\ existe\ almenos\ un\ número\ c\ en\ el\ intervalo\ (a,b) tal\ que\ f_{(c)} = 0 \end{array}}$$

Búsqueda Binaria de Raíces Código

CLASE 1

```
import numpy as np
import matplotlib.pyplot as plt
from tabulate import tabulate
# Método de Bisección
def biseccion(f, a, b, iteraciones=100, tolerancia=1e-6, precision=5):
    if f(a) * f(b) >= 0:
        raise ValueError("La función debe tener signos opuestos en los extremos del intervalo [a, b].")
    results = []
    for i in range(iteraciones):
        c = round((a + b) / 2.0, precision)
        fc = round(f(c), precision)
        results.append([i+1, a, b, c, fc])
        print(tabulate(results, headers=["Iteración", "a", "b", "c", "f(c)"], tablefmt="grid"))
        if abs(fc) < tolerancia or (b - a) / 2.0 < tolerancia:
            return c
        if f(a) * f(c) < 0:
            b = c
        else:
            a = c
    raise ValueError("El método no convergió o faltan iteraciones.")
def graficar_biseccion(f, a, b, raiz):
    # Graficar la función
    x = np.linspace(a - 1, b + 1, 400)
    y = f(x)
    plt.plot(x, y, label='$f(x)$')
    plt.axhline(0, color='black', linewidth=0.5)
    plt.axvline(0, color='black', linewidth=0.5)
    plt.grid(color='gray', linestyle='--', linewidth=0.5)

    # Marcar la raíz encontrada
    plt.plot(raiz, f(raiz), 'ro', label=f'Raíz: x = {raiz:.5f}')

    # Añadir leyenda y mostrar la gráfica
    plt.legend()
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.title('Gráfica de la función y su raíz')
    plt.show()

# Definir la función para la cual quieres encontrar la raíz
def f(x):
    return x**2 - 4

# Intervalo inicial
a = 0
b = 3

# Encontrar la raíz utilizando el método de bisección
raiz = biseccion(f, a, b)

# Imprimir la raíz encontrada
print(f"La raíz encontrada es: {raiz}")

# Graficar la función y la raíz
graficar_biseccion(f, a, b, raiz)
```