Lab 4: Root Finding (Bisection method) and brute force

Algorithms 2022-1

GRUPO 7: Santiago Cassiano - Juliana De Castro - Ivan Morales - Luis Mendez - Juan Oviedo - Nicolás Rincón

Facultad de INGENIERÍA
UNIVERSIDAD NACIONAL DE COLOMBIA

Brute Force vs. Root Finding (Bisection)

$$x^5 - 59x^4 + 35x^3 - 250x^2 + x - 70 = 0$$

Bisection -> O(log n) **->** **0.009537696838378906** segundos
Solución encontrada con algoritmo Bisection x = 58.474600315093994

Brute Force -> O(n) **->** **50.711068868637085** segundos
Solución encontrada con algoritmo Brute Force x = 58.47449973430116

Código realizado

https://colab.research.google.com/drive/1FNiiYuuwfhjRDPq1Yxmf8GoiZkQMyR61?usp=sharing

¿Qué hacer si la función no pasa por cero en el rango descrito?

Brute Force -> No se encuentran soluciones por la verificación del teorema de Bolzano.

Bisection -> No se realiza la búsqueda debido a la verificación en el intervalo inicial del teorema de Bolzano.

¿Puede su algoritmo encontrar más de una raíz?

Brute Force -> Si, estas se guardan en una lista.

Bisection -> No, cuando se encuentra una, se corta la iteración y se muestra la solución.

¿Qué variaciones en la bisección pueden mejorar la velocidad de convergencia?

Se puede mejorar aumentando la tolerancia

```python
def metodo_biseccion(f, a, b, tol=1**-4, n=50):
    """
    Método de bisección
    :param f: Funcion a la que se le intenta encontrar una solucion para la ecuacion f(x)=0, previamente definida
    :param a: limite inferior
    :param b: limite superior
    :param tol: toleracia, criterio de parada
    :param n: número máximo de iteraciones, criterio de parada
    :return: solución exacta o aproximada, si tiene.
    """
    # el intevalo escogido no sirve
    if f(a)*f(b) > 0:  # O(1)
        print('El intervalo no funciona, f(a)={:.2f} y f(b)={:.2f}'.format(f(a),f(b))) # O(1)
        return None # O(1)

    # Calculo inicial de la convergencia. Número de cifras significativas
    e_abs = abs(b-a) # O(1)

    # Variable de iteracion
    i = 0  # O(1)

    while i < n and e_abs > tol: # O(log n)
        # punto medio
        c = (a + b)/2  # O(1)
        print('iteracion {:<2}: a_{:<2}={:.7f} , b_{:<2}={:.7f}, c_{:<2}={:.7f}'.format(i,i-1,a,i-1,b,i,c)) # O(1)
         # solución exacta encontrada
        if f(c) == 0: # O(1)
            print('Solución encontrada x={:.7f}'. format(c)) # O(1)
            return c # O(1)
        # escoger intervalo izquierdo
        if f(a)*f(c)<=0:  # O(1)
            b = c         # O(1)
            c_t = a       # O(1)
        # escoger intervalo derecho
        else:
            a = c    # O(1)
            c_t = b # O(1)
        # error absoluto
        e_abs = abs(c_t - c)  # O(1)
        # criterio de parada. Precisión requerida.
        if e_abs <= tol:  # O(1)
            print('Solución encontrada x= {:.7f}, iteraciones: {}'. format(c,i)) #O(1)
            return c # O(1)
        i += 1 # O(1)
    print('Solución no encontrada, iteraciones agotadas: {}'.format(i-1)) # O(1)
    return None # O(1)
```

Facultad de
INGENIERÍA

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Garcias