Metodo Numérico: Método de Bisección

HENRY CCOARITE DUEÑAS

UNIVERSIDAD NACIONAL DEL ALTIPLANO PUNO

I. Definición y Fundamento Teórico

El **Método de Bisección** (o del intervalo medio) es un algoritmo iterativo para hallar las raíces (ceros) de una función. Su fundamento reside en el **Teorema del Valor Intermedio (TVI)**:

Teorema del Valor Intermedio: Si $f(x)$ es una función **continua** en el intervalo cerrado $[a, b]$, y si $f(a)$ y $f(b)$ tienen signos opuestos ($f(a) \cdot f(b) < 0$), entonces existe al menos una raíz ξ en el intervalo abierto (a, b) tal que $f(\xi) = 0$.

II. Algoritmo Iterativo

El método procede dividiendo repetidamente el intervalo a la mitad, asegurando en cada paso que la raíz se mantenga dentro del nuevo subintervalo.

Paso 1: Estimación del Punto Medio Se calcula la aproximación a la raíz, c, como el punto medio del intervalo $[a, b]$:

$$c = \frac{a + b}{2}$$

Paso 2: Evaluación y Nuevo Intervalo Se evalúa $f(c)$ y se compara su signo con $f(a)$ y $f(b)$:

- Si $f(a) \cdot f(c) < 0$: La raíz se encuentra en $[a, c]$. El nuevo intervalo será $[a, c]$, y se actualiza $b \leftarrow c$.
- Si $f(c) \cdot f(b) < 0$: La raíz se encuentra en $[c, b]$. El nuevo intervalo será $[c, b]$, y se actualiza $a \leftarrow c$.

III. Función de Estudio y Verificación Inicial

La función utilizada en el código Python es el polinomio cúbico:

$$\mathbf{f(x) = x^3 - x - 4}$$

El intervalo inicial de análisis es $[\mathbf{1}, \mathbf{2}]$ con una tolerancia $\epsilon = 10^{-5}$.

Verificación Inicial: $f(1) = -4$ y $f(2) = 2$. Dado que $f(1) \cdot f(2) < 0$, se garantiza la existencia de una raíz en $(1, 2)$.

IV. Código de Implementación (Python)

El código utilizado para implementar el método se presenta a continuación:

```
import math

def biseccion(f, a, b, tol=1e-6, max_iter=100):
    if f(a) * f(b) > 0:
        raise ValueError("Error: f(a) y f(b) deben tener signos opuestos para asegurar la existencia de una ra z en [a, b].")

    print(f"{'Iter':<5} {'a':<18} {'b':<18} {'c':<18} {'f(c)':<18} {'Error':<18}")
    print("-" * 95)

    for i in range(1, max_iter + 1):
        c = (a + b) / 2
```

```
        fc = f(c)

        error = abs(b - a) / 2

        print(f"{i:<5} {a:<18.9f} {b:<18.9f} {c:<18.9f} {fc:<18.9f} {error:<18.9f}")

        if abs(fc) < tol or error < tol:
            print("---")
            print(f" xito  : Ra z aproximada encontrada en {c:.9f}")
            print(f"N mero de iteraciones: {i}")
            return c, i

        if f(a) * fc < 0:
            b = c
        else:
            a = c

    print("-" * 95)
    print(f"Aviso: Se alcanz  el m ximo de {max_iter} iteraciones sin cumplir la
    tolerancia.")
    return c, max_iter

def funcion(x): # DEFINIR LA FUNCI N AQU
    return x**3 - x - 4

print("--- M TODO DE BISECCI N ---")
print("Funci n: f(x) = x^3 - x - 4")
print("Intervalo inicial: [1, 2], Tolerancia: 1e-5\n")

try:
    raiz, iteraciones = biseccion(funcion, 1, 2, tol=1e-5)
    print(f"\nVerificaci n de la ra z: f({raiz:.10e}) = {funcion(raiz):.10e}")
except ValueError as e:
    print(e)
```

Listing 1: Código Python para el Método de Bisección.

V. Salida (Output) del Programa

La siguiente es la salida generada en la consola para el intervalo $[1, 2]$ y tolerancia $\epsilon = 10^{-5}$:

```
--- MÉTODO DE BISECCIÓN ---
Función: f(x) = x^3 - x - 4
Intervalo inicial: [1, 2], Tolerancia: 1e-5

Iter  a                  b                  c                  f(c)               Error
-----------------------------------------------------------------------------------------------
1     1.000000000        2.000000000        1.500000000        -1.375000000        0.500000000
2     1.500000000        2.000000000        1.750000000        -0.015625000        0.250000000
3     1.750000000        2.000000000        1.875000000         0.931640625        0.125000000
4     1.750000000        1.875000000        1.812500000         0.440673828        0.062500000
5     1.750000000        1.812500000        1.781250000         0.207855225        0.031250000
6     1.750000000        1.781250000        1.765625000         0.094593048        0.015625000
7     1.750000000        1.765625000        1.757812500         0.038930893        0.007812500
8     1.750000000        1.757812500        1.753906250         0.011504993        0.003906250
9     1.750000000        1.753906250        1.751953125        -0.002082875        0.001953125
10    1.751953125        1.753906250        1.752929688         0.004705336        0.000976562
11    1.751953125        1.752929688        1.752441406         0.001310574        0.000488281
12    1.751953125        1.752441406        1.752197266        -0.000385966        0.000244141
13    1.752197266        1.752441406        1.752319336         0.000462293        0.000122070
14    1.752197266        1.752319336        1.752258301         0.000038161        0.000061035
15    1.752197266        1.752258301        1.752227783        -0.000173909        0.000030518
16    1.752227783        1.752258301        1.752243042        -0.000067876        0.000015259
17    1.752243042        1.752258301        1.752250671        -0.000014858        0.000007629
---
```

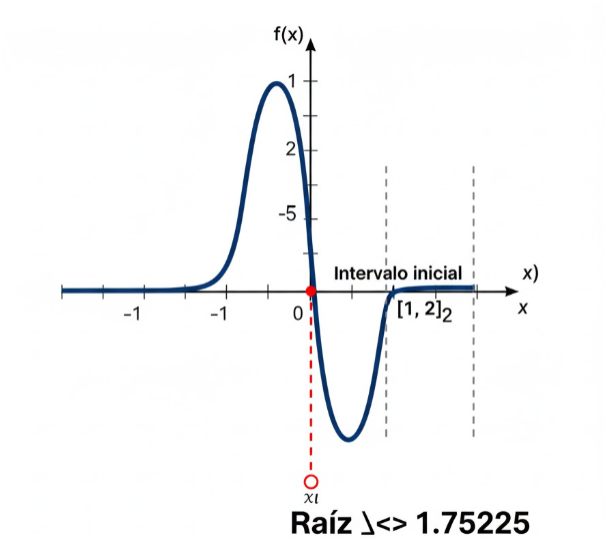

Figure 1: Enter Caption

```
Éxito : Raíz aproximada encontrada en 1.752250671
Número de iteraciones: 17

Verificación de la raíz: f(1.7522506714e+00) = -1.4858485203e-05
```