**Universidad Nacional del Altiplano**
**Facultad de Ingeniería Estadística e Informática**

**E.P.de Ingeniería Estadística e Informática**

**Docente:** Ing. Torres Cruz Fred

**Presentado por:** Quispe Ito Luz Leidy

**Metodo de Biseccion en Python**

# 1. objetivo del programa

El programa desarrollado en Python implementa el **método numérico de bisección**, cuyo propósito es encontrar una raíz real de una función continua $f(x) = 0$ dentro de un intervalo $[a, b]$. El método permite aproximar la solución con una precisión determinada por el usuario mediante una **tolerancia de error**.

# 2. Funcionamiento general

El programa solicita al usuario los siguientes datos desde la consola:

- La función $f(x)$ (por ejemplo: `e^(3x)` $- 4$, `(x+1)(x-2)`, `cos(x)-x`).
- `Los límites del intervalo` $[a, b]$ `donde se busca la raíz.`
- `La tolerancia o error máximo permitido.`

Posteriormente, el algoritmo ejecuta el proceso iterativo del **método de bisección**, basado en el *Teorema del Valor Intermedio*, el cual establece que:

$$f(a) \cdot f(b) < 0 \Rightarrow \text{existe al menos una raíz en el intervalo } [a, b].$$

# 3. Procedimiento del algoritmo

El funcionamiento del programa puede resumirse en las siguientes etapas:

1. **Verificación inicial:** Se evalúan los valores $f(a)$ y $f(b)$. Si $f(a) \cdot f(b) > 0$, el programa muestra un mensaje de error indicando que la función no cambia de signo y, por tanto, no se puede aplicar el método.
2. **Cálculo iterativo:** Mientras el error sea mayor que la tolerancia establecida, se realizan los siguientes pasos:

*a*) Se calcula el punto medio del intervalo:

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

*b*) Se evalúa $f(m)$.

*c*) Si $f(a) \cdot f(m) < 0$, la raíz se encuentra en el subintervalo $[a, m]$; de lo contrario, en $[m, b]$.

*d*) Se actualizan los límites del intervalo.

*e*) Se calcula el error relativo entre dos aproximaciones consecutivas:

$$E = |m_n - m_{n-1}|$$

Si $E <$ tolerancia, el proceso termina.

3. **Presentación de resultados:** En cada iteración se muestra una tabla con los valores:

$$a,\ b,\ m,\ f(a),\ f(b),\ f(m),\ E$$

Al finalizar, el programa presenta:

- La raíz aproximada encontrada.
- El valor de la función en la raíz.
- El número de iteraciones realizadas.
- El error estimado final.

# 4. Restricciones del programa

El algoritmo requiere que se cumplan las siguientes condiciones:

- La función $f(x)$ debe ser continua en el intervalo $[a, b]$.
- Es necesario que $f(a) \cdot f(b) < 0$, de modo que exista un cambio de signo.
- Los límites $a$, $b$ y la tolerancia deben ser valores numéricos válidos.

Además, el programa incluye un **preprocesador de expresiones matemáticas** que adapta las funciones ingresadas por el usuario. Este módulo inserta automáticamente los asteriscos omitidos (por ejemplo, convierte $3x$ en $3 * x$) y reemplaza expresiones comunes como:

`sin`, `cos`, `tan`, `log`, `exp`, `pi`, `e`

por sus equivalentes de la biblioteca `math` de Python.

# 5. Características adicionales y implementacion del codigo en python

- El programa es interactivo: permite ingresar cualquier función desde la consola sin modificar el código.
- Muestra una tabla detallada del proceso iterativo con los valores de cada variable.
- Calcula automáticamente el número de iteraciones necesarias para alcanzar la tolerancia deseada.

File Edit Format Run Options Window Help

```python
import math
import re

def preparar_funcion(expr):
    expr = expr.lower().replace("^", "**")

    expr = re.sub(r"e\^\(?([^)]+)\)?", r"math.exp(\1)", expr)

    expr = re.sub(r'(\d)(x)', r'\1*\2', expr)
    expr = re.sub(r'(x)(\d)', r'\1*\2', expr)
    expr = re.sub(r'(\))(\()', r'\1*\2', expr)
    expr = re.sub(r'(\d)(\()', r'\1*\2', expr)
    expr = re.sub(r'(\))(\d)', r'\1*\2', expr)

    # Reemplaza funciones conocidas para que evalue las reconozca
    expr = expr.replace("exp", "math.exp")
    expr = expr.replace("sin", "math.sin")
    expr = expr.replace("cos", "math.cos")
    expr = expr.replace("tan", "math.tan")
    expr = expr.replace("log", "math.log")
    expr = expr.replace("pi", "math.pi")
    expr = expr.replace("e", "math.e")

    return expr


def biseccion_tabla():
    funcion_str = input("Ingresa la función f(x) ejemplo: e^3x-4: ")
    a = float(input("Ingresa el límite inferior a: "))
    b = float(input("Ingresa el límite superior b: "))
    tol = float(input("Ingresa la tolerancia(Error) Ejemplo(0.01): "))

    funcion_str = preparar_funcion(funcion_str)

    def f(x):
        return eval(funcion_str, {"x": x, "math": math})

    fa, fb = f(a), f(b)
    if fa * fb > 0:
        print("\n Error: f(a) y f(b) tienen el mismo signo. No se puede aplicar bisección.")
        return

    print(f"\n{'Iter':<5}{'a':>10}{'b':>10}{'m':>12}{'f(a)':>12}{'f(b)':>12}{'f(m)':>12}{'Error':>12}")
    print("-"*85)
```

```
45
46      error = float("inf")
47      iteracion = 0
48      m_anterior = 0
49
50      while error > tol:
51          iteracion += 1
52          m = (a + b) / 2
53          fa, fb, fm = f(a), f(b), f(m)
54
55          if iteracion > 1:
56              error = abs(m - m_anterior)
57          m_anterior = m
58
59          print(f"{iteracion:<5}{a:>10.6f}{b:>10.6f}{m:>12.6f}{fa:>12.6f}{fb:>12.6f}{fm:>12.6f}{error:>12.6f}")
60
61          if fa * fm < 0:
62              b = m
63          else:
64              a = m
65
66          if error < tol:
67              break
68
69      print(f"Raíz aproximada: {m:.6f}")
70      print(f"f(raíz): {f(m):.6e}")
71      print(f"Iteraciones: {iteracion}")
72      print(f"Error estimado: {error:.6e}")
73
74
75  biseccion_tabla()
76
```

# 6. Resultados experimentales

En esta sección se presentan los resultados obtenidos al ejecutar el programa para distintas funciones. Las siguientes imágenes muestran el proceso iterativo y la convergencia hacia la raíz buscada.

```
Ingresa la función f(x) ejemplo: e^3x-4: e^3x-4
Ingresa el límite inferior a: 0
Ingresa el límite superior b: 1
Ingresa la tolerancia(Error) Ejemplo(0.01): 0.01

Iter          a          b           m         f(a)         f(b)         f(m)        Error
-------------------------------------------------------------------------------------------
1       0.000000   1.000000    0.500000   -4.000000   16.085537    6.042768          inf
2       0.000000   0.500000    0.250000   -4.000000    6.042768    1.021384     0.250000
3       0.000000   0.250000    0.125000   -4.000000    1.021384   -1.489308     0.125000
4       0.125000   0.250000    0.187500   -1.489308    1.021384   -0.233962     0.062500
5       0.187500   0.250000    0.218750   -0.233962    1.021384    0.393711     0.031250
6       0.187500   0.218750    0.203125   -0.233962    0.393711    0.079875     0.015625
7       0.187500   0.203125    0.195312   -0.233962    0.079875   -0.077044     0.007812
Raíz aproximada: 0.195312
f(raíz): -7.704357e-02
Iteraciones: 7
Error estimado: 7.812500e-03
```

```
Ingresa la función f(x) ejemplo: e^3x-4: e^(3x)-4
Ingresa el límite inferior a: 0
Ingresa el límite superior b: 1
Ingresa la tolerancia(Error) Ejemplo(0.01): 0.1

Iter          a         b           m        f(a)        f(b)        f(m)       Error
--------------------------------------------------------------------------------------
1      0.000000  1.000000    0.500000   -3.000000   16.085537    0.481689         inf
2      0.000000  0.500000    0.250000   -3.000000    0.481689   -1.883000    0.250000
3      0.250000  0.500000    0.375000   -1.883000    0.481689   -0.919783    0.125000
4      0.375000  0.500000    0.437500   -0.919783    0.481689   -0.284549    0.062500
Raíz aproximada: 0.437500
f(raíz): -2.845493e-01
Iteraciones: 4
Error estimado: 6.250000e-02
```

# 7. Conclusión

El programa desarrollado permite determinar de manera eficiente una raíz aproximada de funciones continuas mediante el **método de bisección**. Gracias al control de tolerancia y a la presentación tabular de resultados, se obtiene una visión clara de la convergencia del método. Su diseño flexible e interactivo lo hace una herramienta útil para la enseñanza y el análisis numérico de ecuaciones no lineales.