Escuela Politécnica Nacional

[Tarea 04] Ejercicios Unidad 02-A | Bisección

Nombre: Wellington Barros

CONJUNTO DE EJERICIOS

1. Use el método de bisección para encontrar soluciones precisas dentro de 10^{-4} para la ecuación:

$$x^3 - 7x^2 + 14x - 6 = 0 \qquad (1)$$

en cada uno de los intervalos siguientes:

a. $[0, 1]$
b. $[1, 3.2]$
c. $[3.2, 4]$

a. [0,1]

Utilizando el método de bisección tenemos las siguientes funciones

```python
def f(x):
    return x**3 - 7 * x**2 + 14 * x - 6

# Método de bisección para encontrar una raíz de f(x) en el intervalo [a, b] con precisión de 10^-2
def metodo_biseccion(f, a, b, tol=1e-2):
    # Verificamos que haya un cambio de signo en los extremos del intervalo
    if f(a) * f(b) >= 0:
        return None  # No garantiza que haya una raíz en el intervalo

    # Inicializamos variables
    c = (a + b) / 2.0  # Punto medio
    iter_count = 0  # Contador de iteraciones para seguimiento

    # Iteramos hasta alcanzar la precisión deseada
    while (b - a) / 2.0 > tol:
        iter_count += 1 #Actualizar el contador de iteraciones:
        c = (a + b) / 2.0  # Punto medio
        fc = f(c)

        # Si la función en el punto medio es 0, encontramos la raíz exacta
        if fc == 0 or (b - a) / 2.0 < tol:
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
PS C:\Users\Admin Sistema\Desktop\Metodos Numericos\Tarea 4> & "C:/Users/Admin Sistema/AppData/Local/Programs/Python/Python312/
.exe" "c:/Users/Admin Sistema/Desktop/Metodos Numericos/Tarea 4/Biseccion.py"
0.578125 6
La raiz aproximada con el metodo de biseccion con una presicion de 10*-2 es:0.578125, ademas se lo hizo en 6 iteraciones
PS C:\Users\Admin Sistema\Desktop\Metodos Numericos\Tarea 4>
```

La raíz de la función en el intervalo [0,1]es aproximadamente x=0,578125 con una precisión de 10^{-2} y se lo hizo con 6 iteraciones

b. [1, 3.2]

```python
# Aplicamos el método en el intervalo [1, 3.2]
raiz, iteraciones = metodo_biseccion(f, 1,3.2)
print(raiz, iteraciones)
```

```
PS C:\Users\Admin Sistema\Desktop\Metodos Numericos\Tarea 4> & "C:/Users/Admin Sistema/AppData/Local/Programs/Python/Python312/pyt
.exe" "c:/Users/Admin Sistema/Desktop/Metodos Numericos/Tarea 4/Biseccion.py"
3.0109375000000003 7
La raiz aproximada con el metodo de biseccion con una presicion de 10*-2 es:3.0109375000000003, ademas se lo hizo en 7 iteraciones
PS C:\Users\Admin Sistema\Desktop\Metodos Numericos\Tarea 4>
```

Se aplico nuevamente el mismo método solo que se cambio el intervalo en el que se va a realizar el método

En este caso La raíz de la función en el intervalo [1,3.2]es aproximadamente x= es:3:109375000000003 con una precisión de 10^{-2} y se lo hizo con 7 iteraciones

c. [3.2, 4]

```
def metodo_biseccion(f, a, b, tol=1e-2):
    # Verificamos que haya un cambio de signo en los extremos del intervalo
    if f(a) * f(b) >= 0:
        print("El intervalo no contiene una raíz, ya que no hay un cambio de signo en sus extremos.")
        return None  # No garantiza que haya una raíz en el intervalo
```

```
# Aplicamos el método en un intervalo que contenga la raíz
raiz, iteraciones = metodo_biseccion(f, 3.2, 4)  # Usamos el intervalo [2, 3]
if raiz is not None:
    print(f"La raíz aproximada con el método de bisección con una precisión de 10^-2 es: {raiz}, obtenida en {i
else:
    print("No se encontró una raíz en el intervalo dado.")
```

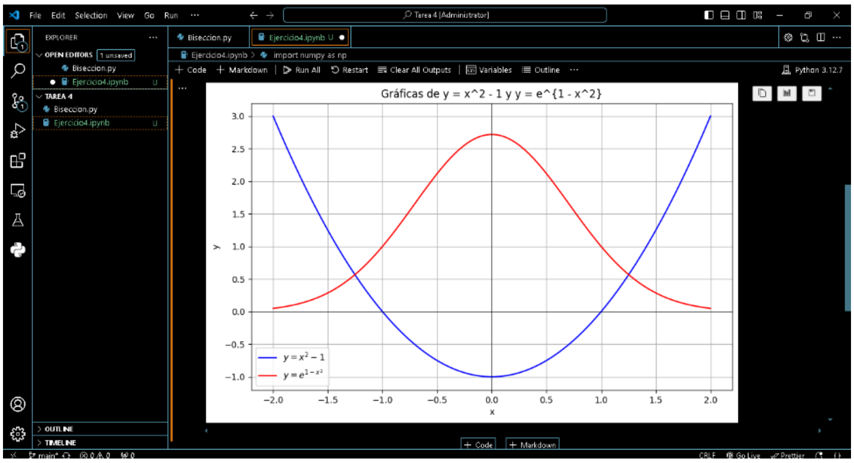
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Admin Sistema\Desktop\Metodos Numericos\Tarea 4> & "C:/Users/Admin Sistema/AppData/Local/Programs/Python/Python312/py
.exe" "c:/Users/Admin Sistema/Desktop/Metodos Numericos/Tarea 4/Biseccion.py"
La raíz aproximada con el método de bisección con una precisión de 10^-2 es: 3.4125000000000005, obtenida en 6 iteraciones.
PS C:\Users\Admin Sistema\Desktop\Metodos Numericos\Tarea 4>
```

Se añadió una nueva condición a la función para que cuando el intervalo en cuestión no contenga una raíz por no haber existencia de cambio de signo en los extremos.

En este caso La raíz de la función en el intervalo [3.4, 4]es aproximadamente x= es: 3.4125000000000005 con una precisión de 10^{-2} y se lo hizo con 6 iteraciones

4. a. Dibuje las gráficas para $y = x^2 - 1$ y $y = e^{1-x^2}$.




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

# Definición de los valores de x
x = np.linspace(-2, 2, 100)

# Definición de las funciones
y1 = x**2 - 1
y2 = np.exp(1 - x**2)

# Graficar ambas funciones
plt.figure(figsize=(10, 6))
plt.plot(x, y1, label=r"$y = x^2 - 1$", color='blue')
plt.plot(x, y2, label=r"$y = e^{1 - x^2}$", color='red')
plt.xlabel("x")
plt.ylabel("y")
plt.title("Gráficas de y = x^2 - 1 y y = e^{1 - x^2}")
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.legend()
plt.grid(True)
plt.show()
```

Se utilizaron las bibliotecas numpy y matplotlib para graficar dos funciones matemáticas en el mismo plano cartesiano x = np.linspace(-2, 2, 100) utiliza la función linspace de la biblioteca numpy para crear un arreglo de valores equidistantes entre dos números, en este caso entre -2 y 2, donde -2 es el valor inicial del eje x, 2 el final del eje x y 100 son la cantidad de puntos que se generan dentro del rango.

EJERCICIOS APLICADOS

1. Un abrevadero de longitud L tiene una sección transversal en forma de semicírculo con radio r. Cuando se llena con agua hasta una distancia h desde la parte superior, el volumen V de agua es:

$$V = L\left(0.5\pi r^2 - r^2 \arcsin\frac{h}{r} - h\sqrt{r^2 - h^2}\right) \qquad (2)$$

Suponga que $L = 10$ cm, $r = 1$ cm y $V = 12.4$ cm^3. Encuentre la profundidad del agua en el abrevadero dentro de 0.01 cm.

Para resolver esto, podemos implementar un método numérico como el método de bisección o Newton-Raphson para hallar la raíz de la ecuación $f(h) = V - V(h) = 0$

```python
L = 10  # Longitud del abrevadero en cm
r = 1   # Radio en cm
V_dado = 12.4  # Volumen dado en cm^3
tolerancia = 0.01  # Tolerancia en cm

# Definición de la función V(h) - V_dado para encontrar la raíz
def volumen(h):
    term1 = 0.5 * np.pi * r**2
    term2 = r**2 * np.arcsin(h / r)
    term3 = h * np.sqrt(r**2 - h**2)
    return L * (term1 - term2 - term3)

def funcion_objetivo(h):
    return volumen(h) - V_dado

# Método de Bisección para encontrar h tal que volumen(h) ≈ V_dado
def biseccion(f, a, b, tol):
    while (b - a) / 2 > tol:
        c = (a + b) / 2
        if f(c) == 0:
            return c  # Encontramos la raíz exacta
        elif f(a) * f(c) < 0:
            b = c
        else:
            a = c
    return (a + b) / 2
```

Respuesta: La profundidad del agua h en el abrevadero es aproximadamente: 0.16 cm

2. Un objeto que cae verticalmente a través del aire está sujeto a una resistencia viscosa y a la fuerza de gravedad. La altura del objeto después de t segundos es:

$$s(t) = s_0 - \frac{mg}{k}t + \frac{mg}{k}\left(1 - e^{-\frac{k}{m}t}\right) \tag{3}$$

donde $g = 9.81\, m/s^2$ y k representa el coeficiente de resistencia del aire en N·s/m. Suponga que $s_0 = 300$ m, $m = 0.25$ kg y $k = 0.1$ N·s/m. Encuentre, dentro de 0.01 segundos, el tiempo que tarda el objeto en tocar el suelo.

Código:

```python
import numpy as np

# Definición de constantes
g = 9.81  # Gravedad en m/s^2
s0 = 300  # Altura inicial en metros
m = 0.25  # Masa en kg
k = 0.1   # Coeficiente de resistencia en Ns/m
tolerancia = 0.01  # Tolerancia en segundos

# Definición de la función s(t) - altura del objeto en función del tiempo
def s(t):
    term1 = s0
    term2 = -(m * g / k) * t
    term3 = (m**2 * g / k**2) * (1 - np.exp(-k * t / m))
    return term1 + term2 + term3

# Definición de la función objetivo f(t) = s(t) - 0, para que busquemos f(t) = 0
def funcion_objetivo(t):
    return s(t)

# Método de Bisección para encontrar t tal que s(t) ≈ 0
def biseccion(f, a, b, tol):
    while (b - a) / 2 > tol:
        c = (a + b) / 2
        if f(c) == 0:
            return c  # Encontramos la raíz exacta
        elif f(a) * f(c) < 0:
            b = c
        else:
            a = c
    return (a + b) / 2
```

```python
# Rango inicial para t
a = 0  # Tiempo inicial
b = 100  # Estimación de tiempo máximo (ajústalo si es necesario)

# Calcular t con la tolerancia deseada
t_aproximado = biseccion(funcion_objetivo, a, b, tolerancia)
print(f"El tiempo aproximado que tarda el objeto en caer al suelo es: {t_aproximado:.2f} segundos")
```

Dado que se busca el valor del tiempo que hace s(t)=0s, se deberá encontra la raíz de la ecuación. Se utiliza un método numérico, como el método de bisección o Newton-Raphson

Respuesta: El tiempo aproximado que tarda el objeto en caer al suelo es: 14.73 segundos

EJERCICIOS TEÓRICOS

1. Use el teorema 2.1 para encontrar una cota para el número de iteraciones necesarias para lograr una aproximación con precisión de 10^{-4} para la solución de $x^3 - x - 1 = 0$ que se encuentra dentro del intervalo $[1, 2]$. Encuentre una aproximación para la raíz con este grado de precisión.

Teorema para la cota de iteraciones en el método de bisección

El Teorema 2.1 nos dice que el número mínimo de iteraciones, n, necesario para alcanzar una precisión ϵ en el intervalo [a,b], se calcula con la siguiente fórmula:

$$n \geq \frac{\log \frac{b-a}{\epsilon}}{\log 2}$$

En este caso la estimación del numero de interaciones se hará mediante el método de bisección. Pues se requiere que la aproximación tenga una precisión de $\epsilon = 10^{-5}$, es decir, que el error máximo permitido debe ser menor que dicha precisión.

1. **Definir el intervalo y la precisión**:

   Se tiene el intervalo [a,b]=[1,2] y una precisión deseada de $\epsilon = 10^5$.

2. **Cálculo del cociente inicial**

   Primero, se encuentra el ancho del intervalo: b−a=2−1=1. Luego, dividimos el ancho del intervalo por la precisión deseada

$$\frac{1}{10^{-5}} = 10^5$$

3. **Tomar el logaritmo del cociente**:

   Se calcula el logaritmo en base 10 de 10^5, lo cual nos da: $\log(10^5) = 5$

4. **Dividir entre log (2)**

   Ahora, para obtener el valor de n, dividimos 5 entre log(2). Se utiliza un valor aproximado para log (2)≈0.30103

$$n \geq \frac{5}{0.30103} \approx 16.61$$

Redondeando ese número, entonces el número de iteraciones serian 17.

Link de GitHub – Repositorio

https://github.com/wilypoli/Tarea-4