ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA EN COMPUTACION

PERÍODO ACADÉMICO: 2025-A
ASIGNATURA: ICCD412 Métodos Numéricos **GRUPO:** GR2
TIPO DE INSTRUMENTO: Deber N°5
FECHA DE ENTREGA LÍMITE: 04/05/2025
ALUMNO: Lema Luis

TEMA

Método de la bisección

OBJETIVOS

- Aplicar el método de bisección para encontrar raíces aproximadas de funciones dentro de un intervalo, usando una tolerancia dada y verificando su comportamiento con ayuda de gráficos.

DESARROLLO

- Codificar el pseudocódigo dado en clase

```python
def biseccion(f, a, b, TOL, N0):
    i = 1
    FA = f(a)

    while i <= N0:
        p = a + (b - a) / 2
        FP = f(p)

        if FP == 0 or (b - a) / 2 < TOL:
            print("Procedimiento completado exitosamente.")
            return p

        i = i + 1

        if FA * FP > 0:
            a = p
            FA = FP
        else:
            b = p

    print("El método fracasó después de", N0,
"iteraciones.")
    return None
def funcion_prueba(x):
    return x**3 - 7*x**2 + 14*x - 6

raiz = biseccion(funcion_prueba, 0, 1, 1e-2, 100)

if raiz is not None:
    print("Raíz aproximada encontrada:", round(raiz, 4))
```

```
PS C:\Users\luisl\Documents\Metodos Numericos 2025> & C:/Users/luisl/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/luisl/Documents/Me
cos 2025/Metodos-Numericos-2025/Primer Bimestre/Deber_5/Deber_3_Latex/limitedecaracterescompudeber.py"
Procedimiento completado exitosamente.
Raíz aproximada encontrada: 0.5859
PS C:\Users\luisl\Documents\Metodos Numericos 2025> 
```

- Use el método de bisección para encontrar soluciones precisas

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

def biseccion(a, b, tol):
    if f(a) * f(b) >= 0:
        print(
"No se puede aplicar bisección en el intervalo [", a, ",",
b, "]")
        return None
    while abs(b - a) > tol:
        c = (a + b) / 2
        if f(c) == 0:
            break
        elif f(a) * f(c) < 0:
            b = c
        else:
            a = c
    return c

# Intervalos del ejerccio
intervalos = [(0, 1), (1, 3.2), (3.2, 4)]
for a, b in intervalos:
    raiz = biseccion(a, b, 1e-2)
    print(f"Raíz aproximada en [{a}, {b}]: {raiz:.4f}")
```

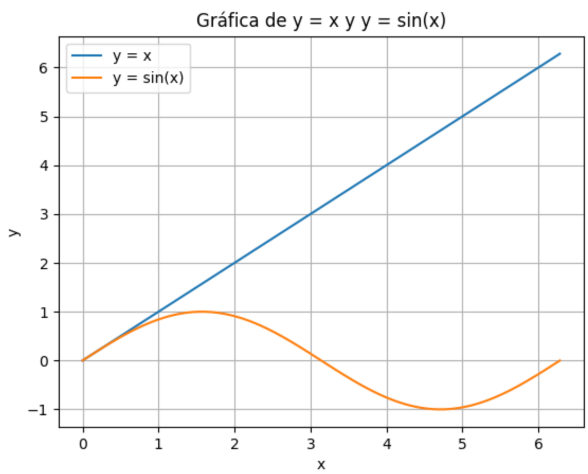
```
PS C:\Users\luisl\Documents\Metodos Numericos 2025> & C:/Users/luisl/AppData/Local/Programs
cos 2025/Metodos-Numericos-2025/Primer Bimestre/Deber_5/Deber_3_Latex/ejercicio1.py"
Raíz aproximada en [0, 1]: 0.5859
Raíz aproximada en [1, 3.2]: 3.0023
Raíz aproximada en [3.2, 4]: 3.4188
PS C:\Users\luisl\Documents\Metodos Numericos 2025>
```

- Dibuje las gráficas para y = x y y= sin x

```python
# Importamos matplotlib y numpy solo para hacer el gráfico
# matplotlib sirve para graficar y numpy para generar los valores
import matplotlib.pyplot as plt
import math
import numpy as np

x = np.linspace(0, 2*np.pi, 100)  # Se generan 100 valores desde
y1 = x
y2 = np.sin(x)

plt.plot(x, y1, label='y = x')
plt.plot(x, y2, label='y = sin(x)')
plt.legend()
plt.title('Gráfica de y = x y y = sin(x)')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.show()
```



- Use el método de bisección para encontrar soluciones precisas dentro de 10-5 para el primer valor positivo de x con x = 2 sin x

```python
import math

def f(x):
    return x - 2 * math.sin(x)

def biseccion(f, a, b, TOL, N0):
    i = 1
    FA = f(a)

    while i <= N0:
        p = a + (b - a) / 2
        FP = f(p)

        if FP == 0 or (b - a) / 2 < TOL:
            print("Procedimiento completado exitosamente.")
            return p

        i += 1

        if FA * FP > 0:
            a = p
            FA = FP
        else:
            b = p

    print("El método fracasó después de", N0,
"iteraciones.")
    return None

# Llamada correcta
raiz = biseccion(f, 1, 2, 1e-5, 100)
print("Raíz encontrada:", raiz)
```

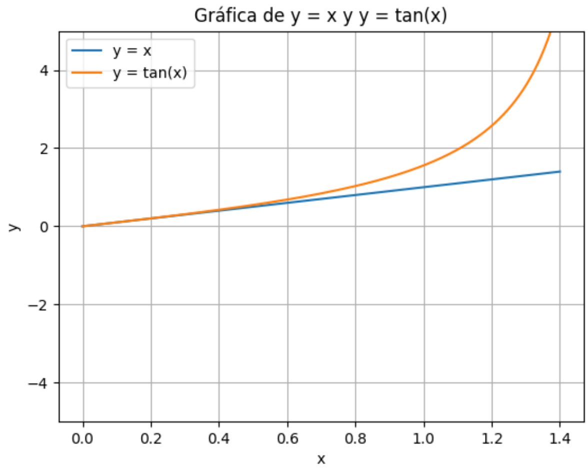
```
PS C:\Users\luisl\Documents\Metodos Numericos 2025> & C:/Users/luisl/AppData/Local/Programs/
cos 2025/Metodos-Numericos-2025/Primer Bimestre/Deber_5/Deber_3_Latex/ejercicio2.py"
Procedimiento completado exitosamente.
Raíz encontrada: 1.8955001831054688
PS C:\Users\luisl\Documents\Metodos Numericos 2025>
```

- Dibuje las gráficas para y = x y y = tan x

```python
# usamos matplotlib y numpy solo para graficar
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 1.4, 400)
# Elegimos 400 puntos entre 0 y 1.4 porque tan(x) tiende a infini
y1 = x
y2 = np.tan(x)

plt.plot(x, y1, label='y = x')
plt.plot(x, y2, label='y = tan(x)')
plt.ylim(-5, 5)  # Limitamos el eje y para que no se vea tan brus
plt.legend()
plt.title('Gráfica de y = x y y = tan(x)')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.show()
```



- Use el método de bisección para encontrar una aproximación dentro de 10-5 para el primer valor positivo de x con x = tan x

```python
import math

# Función f(x) = x - tan(x), se busca el punto donde x = tan(x)
def f(x):
    return x - math.tan(x)

# Método de bisección
def biseccion(f, a, b, TOL, N0):
    i = 1
    FA = f(a)

    while i <= N0:
        p = a + (b - a) / 2
        FP = f(p)

        if FP == 0 or (b - a) / 2 < TOL:
            print("Procedimiento completado exitosamente.")
            return p

        i += 1

        if FA * FP > 0:
            a = p
            FA = FP
        else:
            b = p

    print("El método fracasó después de", N0, "iteraciones.")
    return None

# Intervalo adecuado (evitamos la asíntota vertical de tan(x))
raiz = biseccion(f, 0.0, 1.4, 1e-5, 100)

if raiz is not None:
    print("Raíz aproximada encontrada:", round(raiz, 6))

## precaución no usar mas de b= 1.57o mas de la tan(x)
```

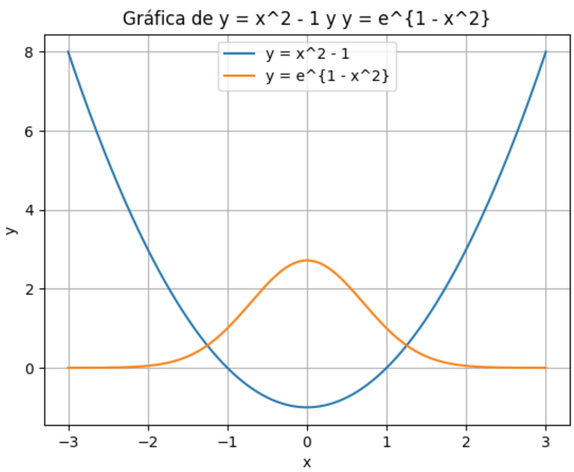
```
PS C:\Users\luisl\Documents\Metodos Numericos 2025> & C:/Users/luisl/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/luisl/Documents/Metodos Numericos 2025/Metodos-Numericos-2025/Primer Bimestre/Deber_5/Deber_3_Latex/ejercicio3.py"
Procedimiento completado exitosamente.
Raíz aproximada encontrada: 5e-06
PS C:\Users\luisl\Documents\Metodos Numericos 2025>
```

- Dibuje las gráficas para y $= x^2 - 1yy = e^1 - x^2$

```
# Nuevamente solo se usa matplotlib y numpy para graficar
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-3, 3, 400)
y1 = x**2 - 1
y2 = np.exp(1 - x**2)

plt.plot(x, y1, label='y = x^2 - 1')
plt.plot(x, y2, label='y = e^{1 - x^2}')
plt.legend()
plt.title('Gráfica de y = x^2 - 1 y y = e^{1 - x^2}')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.show()
```



- Use el método de bisección para encontrar una aproximación dentro de 10- para un valor -2,0 con $x^2 - 1 = e^1 - x^2$

```python
import math

# Definimos la función f(x) = x^2 - 1 - e^(1 - x^2)
def f4(x):
    return x**2 - 1 - math.exp(1 - x**2)

# Método de bisección
def biseccion(f, a, b, TOL, N0):
    i = 1
    FA = f(a)

    while i <= N0:
        p = a + (b - a) / 2
        FP = f(p)

        if FP == 0 or (b - a) / 2 < TOL:
            return p

        i += 1

        if FA * FP > 0:
            a = p
            FA = FP
        else:
            b = p

    return None

# Se busca una raíz en el intervalo [-2, 0] con tolerancia 10^-3
raiz = biseccion(f4, -2, 0, 1e-3, 100)

if raiz is not None:
    print(f"Raíz aproximada en [-2, 0]: {raiz:.4f}")
else:
    print("No se encontró solución en el intervalo dado.")
```

```
PS C:\Users\luisl\Documents\Metodos Numericos 2025> & C:/Users/luisl/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/luisl/Do
cos 2025/Metodos-Numericos-2025/Primer Bimestre/Deber_5/Deber_3_Latex/ejercicio4.py"
Raíz aproximada en [-2, 0]: -1.2510
PS C:\Users\luisl\Documents\Metodos Numericos 2025>
```

- Sea $f(x) = \frac{(x+2)(x+1)^2 x(x-1)^3}{(x-2)}$. ¿En qué cero de f converge el método de bisección cuando se aplica en los siguientes intervalos?

```python
def f(x):
    if x == 2:
        return float('inf')  # evitar división por cero
    return ((x + 2) * (x + 1)**2 * x * (x - 1)**3) / (x - 2
)

def biseccion(f, a, b, tol, max_iter):
    fa = f(a)
    fb = f(b)

    if fa * fb > 0:
        return None  # no hay cambio de signo

    for i in range(max_iter):
        p = (a + b) / 2
        fp = f(p)

        if abs(fp) < tol or (b - a) / 2 < tol:
            return p

        if fa * fp < 0:
            b = p
            fb = fp
        else:
            a = p
            fa = fp

    return (a + b) / 2

# Intervalos a analizar
intervalos = [
    [-1.5, 2.5],
    [-0.5, 2.4],
    [-0.5, 3],
    [-3, -0.5]
]

for i, intervalo in enumerate(intervalos):
    a, b = intervalo
    raiz = biseccion(f, a, b, 1e-5, 100)
    if raiz is not None:
        print(f"Intervalo {chr(97+i)}) [{a}, {b}
] -> Raíz aproximada: {raiz:.5f}")
    else:
        print(f"Intervalo {chr(97+i)}) [{a}, {b}
] -> No hay cambio de signo")
```

```
os 2025/Metodos-Numericos-2025/Primer Bimestre/Deber_5/Deber_3_Latex/ejercicio4.py"
aíz aproximada en [-2, 0]: -1.2510
S C:\Users\luisl\Documents\Metodos Numericos 2025> & C:/Users/luisl/AppData/Local/Programs/Python/Python31
os 2025/Metodos-Numericos-2025/Primer Bimestre/Deber_5/Deber_3_Latex/ejercicio5.py"
ntervalo a) [-1.5, 2.5] -> Raíz aproximada: 0.00000
ntervalo b) [-0.5, 2.4] -> Raíz aproximada: 0.00001
ntervalo c) [-0.5, 3] -> Raíz aproximada: 2.00000
ntervalo d) [-3, -0.5] -> Raíz aproximada: -2.00001
S C:\Users\luisl\Documents\Metodos Numericos 2025>
```

CONCLUSIONES

- El método de bisección funciona bien para encontrar raíces cuando la función cambia de signo en el intervalo. Aunque es un poco lento, es fácil de aplicar y da buenos resultados si se respetan las condiciones.