ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

INGENIERÍA *EN COMPUTACION*

PERÍODO ACADÉMICO: 2025-A

ASIGNATURA: ICCD412 Métodos Numéricos GRUPO: GR2

TIPO DE INSTRUMENTO: *Tarea N°5*

FECHA DE ENTREGA LÍMITE: *[04/05/2025]*

ALUMNO: *Kevin Eduardo Garcia Rodríguez*

TEMA

Método de bisección

OBJETIVOS

- *Encontrar raíces de funciones con precisión controlada*
- *Desarrollar habilidades en análisis numérico y resolución de problemas no lineales*

DESARROLLO

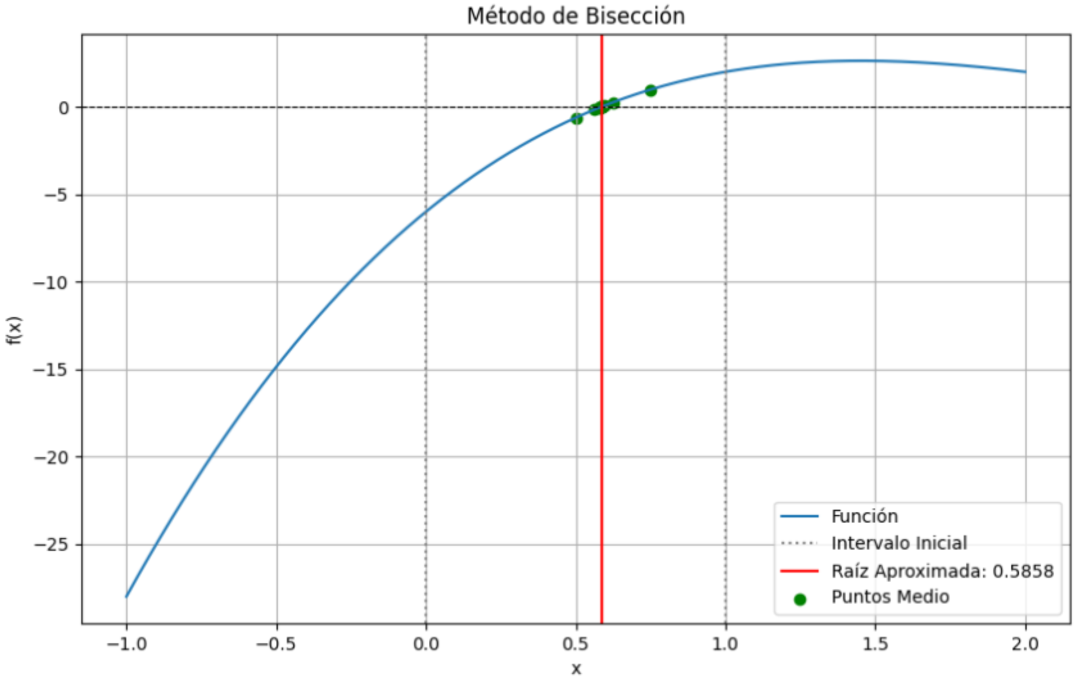
1. Use el método de bisección para encontrar soluciones precisas dentro de 10^{-2} para $x^3 - 7x^2 + 14x - 6 = 0$ en cada intervalo.

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

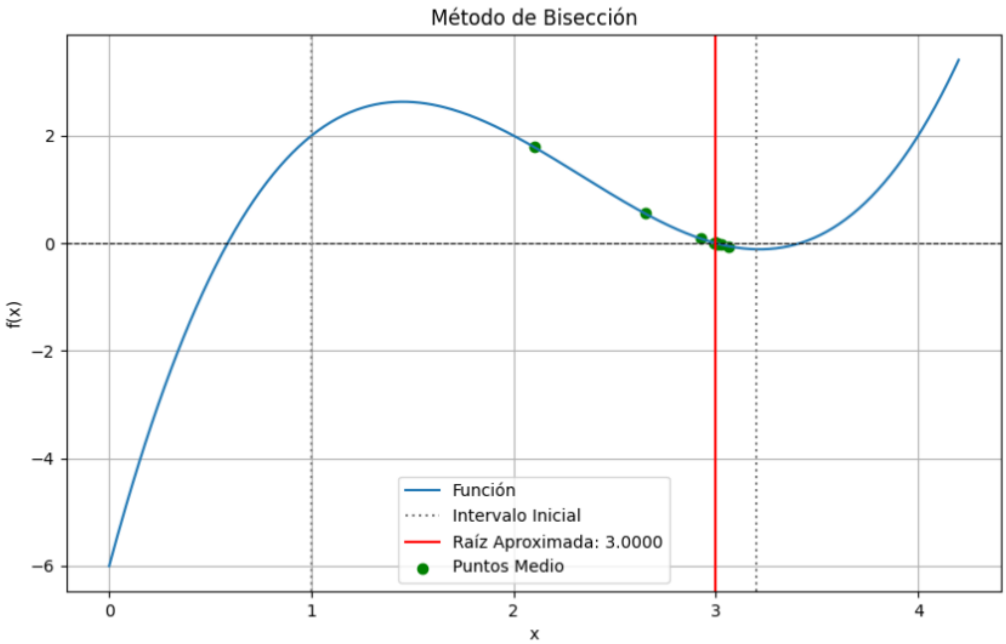
```python
import numpy as np
import matplotlib.pyplot as plt

def biseccion(funcion, a, b, tolerancia=1e-7, max_iter=100):
    """
    Encuentra una raíz de la función utilizando el método de bisección.

    Args:
        funcion (callable): La función para la cual se busca la raíz.
                           Debe tomar un único argumento numérico y devolver un valor numérico.
        a (float): El límite inferior del intervalo inicial.
        b (float): El límite superior del intervalo inicial.
        tolerancia (float, opcional): La tolerancia deseada para la raíz.
                                      El algoritmo se detiene cuando |b - a| < tolerancia.
                                      Por defecto es 1e-7.
        max_iter (int, opcional): El número máximo de iteraciones permitidas.
                                  Por defecto es 100.

    Returns:
        float or None: La aproximación de la raíz si se encuentra dentro de la tolerancia
                       y el número máximo de iteraciones, o None si no se cumple.
    """
    if funcion(a) * funcion(b) >= 0:
        print("El intervalo inicial [a, b] no contiene una raíz o contiene múltiples raíces del mismo signo.")
        return None

    iteraciones = 0
    puntos_medio = []
    while (b - a) / 2 > tolerancia and iteraciones < max_iter:
        c = (a + b) / 2
        puntos_medio.append(c)
        if funcion(c) == 0:
            return c
        elif funcion(a) * funcion(c) < 0:
            b = c
        else:
            a = c
        iteraciones += 1

    raiz_aproximada = (a + b) / 2
    print(f"Raíz aproximada encontrada en {iteraciones} iteraciones.")
    return raiz_aproximada, puntos_medio
```

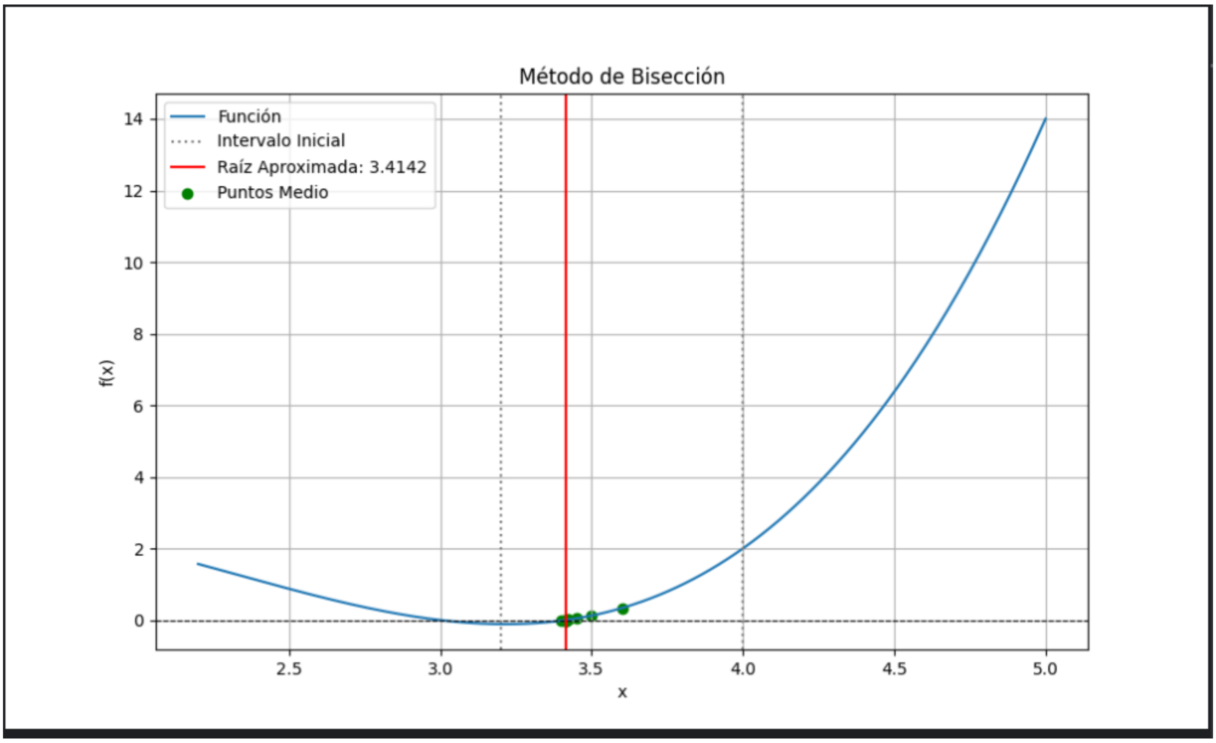
```python
def biseccion(funcion, a, b, tolerancia=1e-7, max_iter=100):
    print(f"Raíz aproximada encontrada en {iteraciones} iteraciones.")
    return raiz_aproximada, puntos_medio

def visualizar_biseccion(funcion, a_inicial, b_inicial, raiz_encontrada=None, puntos_medio=None):
    """
    Visualiza la función y el proceso de bisección.

    Args:
        funcion (callable): La función graficada.
        a_inicial (float): El límite inferior del intervalo inicial.
        b_inicial (float): El límite superior del intervalo inicial.
        raiz_encontrada (float, opcional): La raíz encontrada por el método de bisección.
                                          Se mostrará con una línea vertical.
        puntos_medio (list, opcional): Lista de los puntos medios calculados en cada iteración.
                                      Se mostrarán con puntos en la gráfica.
    """
    x = np.linspace(a_inicial - 1, b_inicial + 1, 400)
    y = funcion(x)

    plt.figure(figsize=(10, 6))
    plt.plot(x, y, label='Función')
    plt.axhline(0, color='black', linewidth=0.8, linestyle='--')
    plt.axvline(a_inicial, color='gray', linestyle=':', label='Intervalo Inicial')
    plt.axvline(b_inicial, color='gray', linestyle=':')

    if raiz_encontrada is not None:
        plt.axvline(raiz_encontrada, color='red', linestyle='-', label=f'Raiz Aproximada: {raiz_encontrada:.4f}')

    if puntos_medio is not None:
        y_vals = [funcion(pm) for pm in puntos_medio]
        plt.scatter(puntos_medio, y_vals, color='green', marker='o', label='Puntos Medio')

    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.title('Método de Bisección')
    plt.legend()
    plt.grid(True)
    plt.show()

if __name__ == "__main__":
    # Define la función cuya raíz quieres encontrar
    def mi_funcion(x):
        return (x + 2) * (x + 1)**2 * x * (x - 1)**3 * (x - 2)

    # Define el intervalo inicial [a, b] donde se espera encontrar una raíz
    a_inicial = -1.5
    b_inicial = 2.5
```

```
Raíz aproximada encontrada en 23 iteraciones.
La raíz aproximada es: 0.58578640

Process finished with exit code 0
```



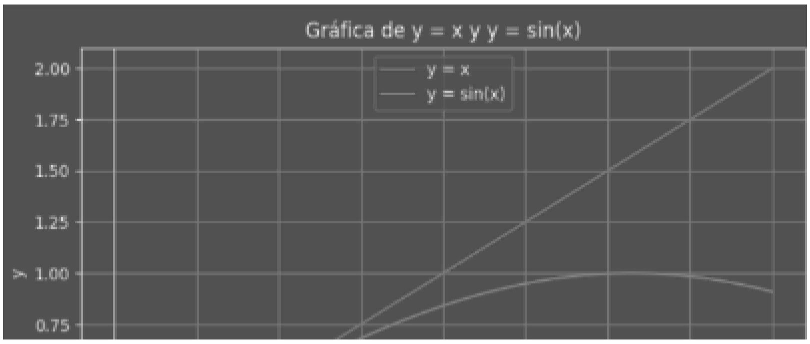

```
Raíz aproximada encontrada en 24 iteraciones.
La raíz aproximada es: 2.99999999

Process finished with exit code 0
```






```
Raíz aproximada encontrada en 22 iteraciones.
La raíz aproximada es: 3.41421366

Process finished with exit code 0
```

2. a. Dibuje las gráficas para $y = x$ y $y = \sin x$.
b. Use el método de bisección para encontrar soluciones precisas dentro de 10^{-5} para el primer valor positivo de x con $x = 2 \sin x$.

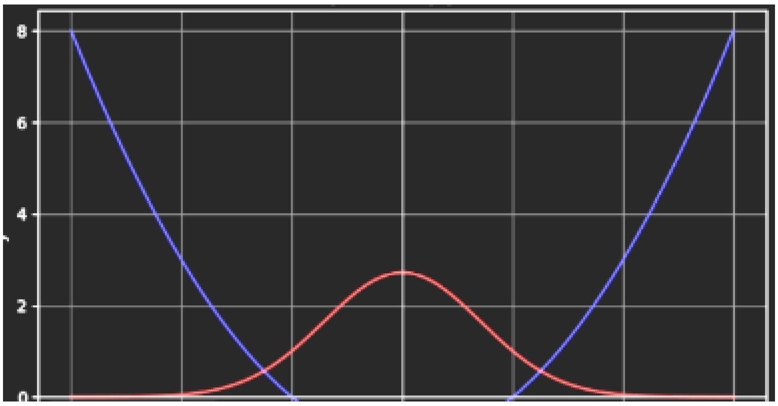



```
Procedimiento completado exitosamente en la iteración 17.
La raíz aproximada es: 1.8955001831054688
```

3. a. Dibuje las gráficas para $y = x$ y $y = \tan x$.
b. Use el método de bisección para encontrar una aproximación dentro de 10^{-5} para el primer valor positivo de x con $x = \tan x$.


```
Procedimiento completado exitosamente en la iteración 17.
La raíz aproximada es: 1.5707931518554688
```

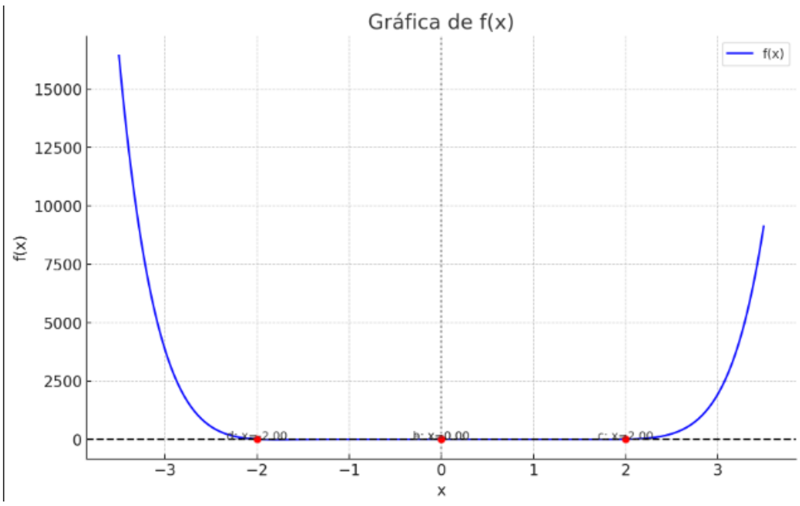
4. a. Dibuje las gráficas para $y = x^2 - 1$ y $y = e^{1-x^2}$.
b. Use el método de bisección para encontrar una aproximación dentro de 10^{-3} para un valor en $[-2, 0]$ con $x^2 - 1 = e^{1-x^2}$.



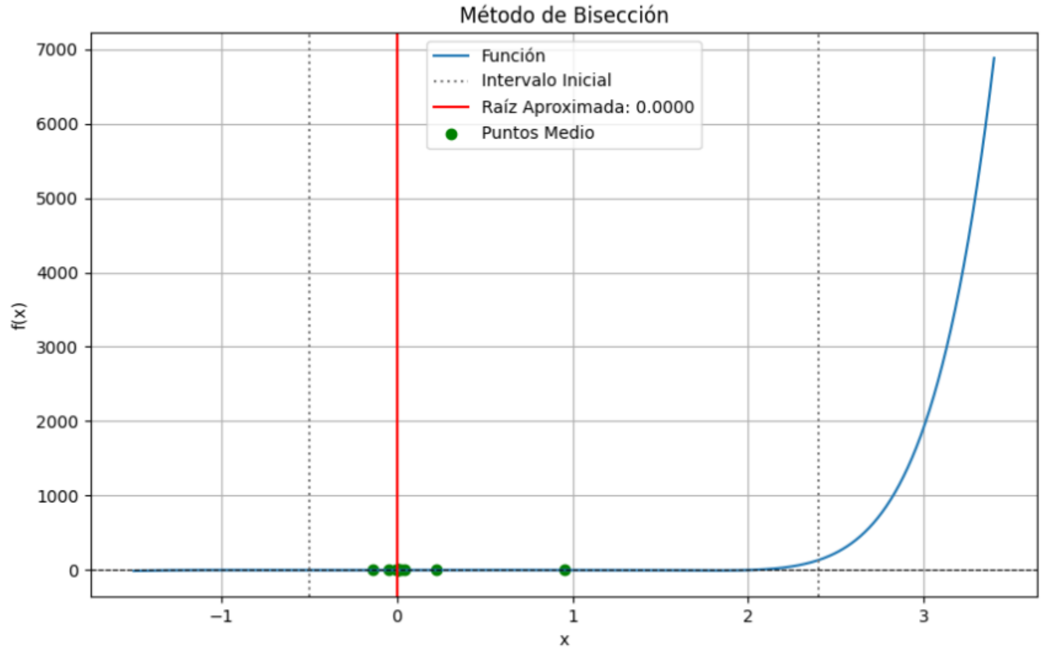

```
Procedimiento completado exitosamente en la iteración 3.
Intervalo a: converge a raíz ≈ 0.00000
```

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

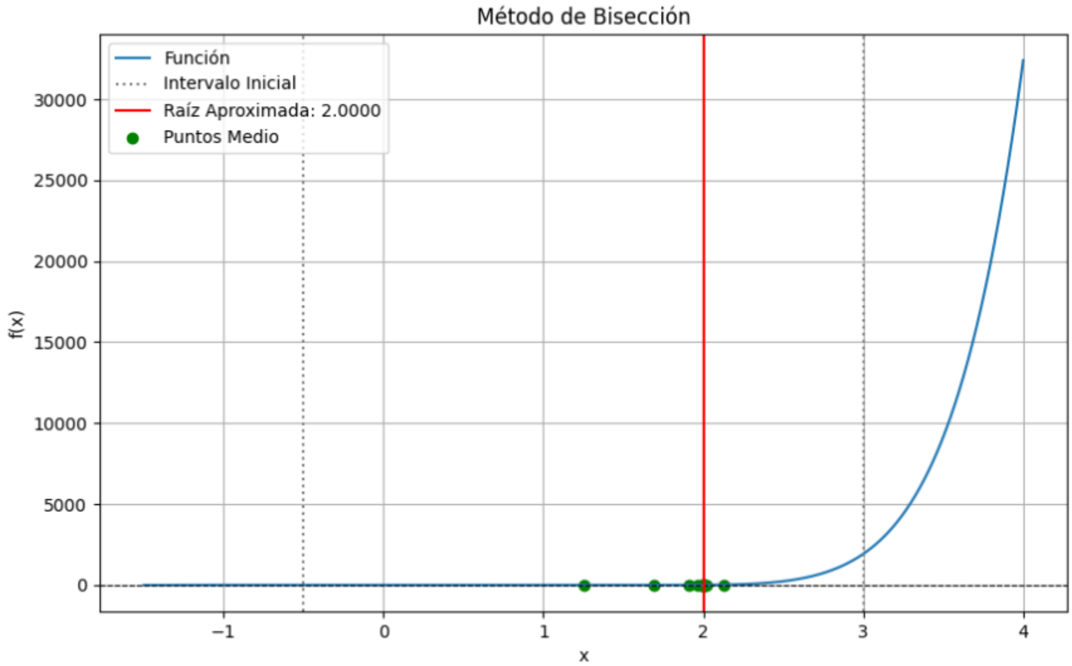
a. $[-1.5, 2.5]$ b. $[-0.5,\ 2.4]$ c. $[-0.5, 3]$ d. $[-3, -0.5]$



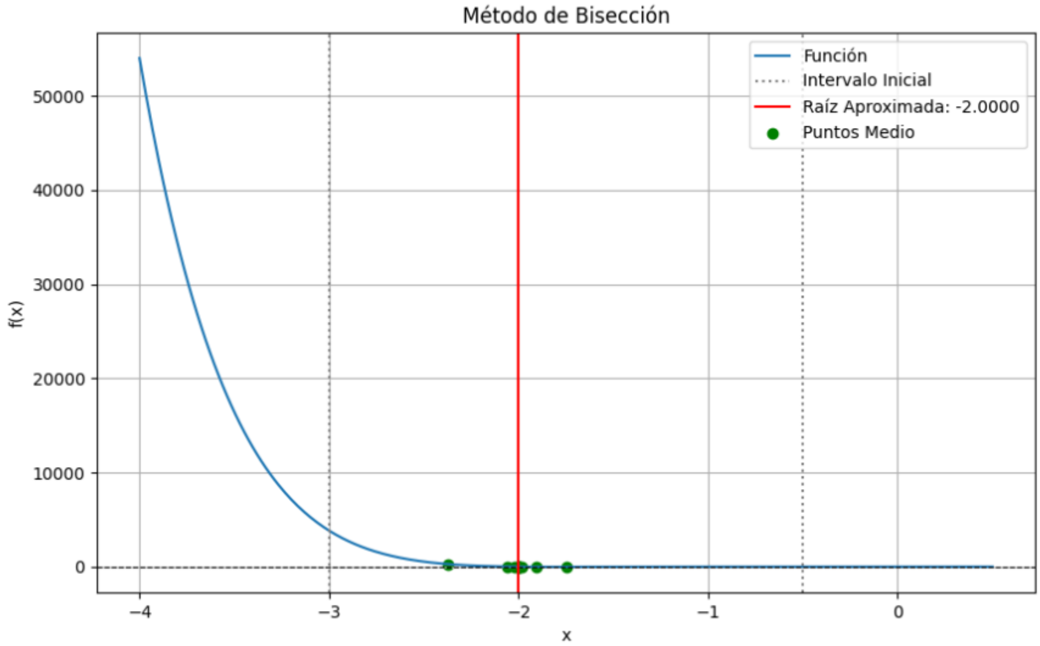

```
Procedimiento completado exitosamente en la iteración 3.
Intervalo a: converge a raíz ≈ 0.00000
Procedimiento completado exitosamente en la iteración 18
```




```
Raíz aproximada encontrada en 24 iteraciones.
La raíz aproximada es: 0.00000001
```




```
Raíz aproximada encontrada en 25 iteraciones.
La raíz aproximada es: 2.00000001
```



```
Raíz aproximada encontrada en 24 iteraciones.
La raíz aproximada es: -1.99999999
```