Escuela Politécnica Nacional

Metodos Numericos - [Tarea 04] Ejercicios Unidad 02-A | Bisección

Quilumba Morocho Joel Patricio

2024-05-27

Tabla de contenidos

1 Conjunto De Ejercicios

1.1 Pregunta 1

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

a.[0,1]

Definimos la funcion $f(x) = x^3 - 7x^2 + 14x - 6$

Verificamos los valores iniciales $f(0) = -6$ y $f(1) = 2$
Como $f(0) \cdot f(1) < 0$, sabemos que hay una raíz en el intervalo [0, 1].

El primer punto medio
$m = \frac{0+1}{2} = 0.5$
$f(0.5) = 2.875$
Como $f(0.5) < 0$, la raíz está en [0.5, 1]

El segundo punto medio
$m = \frac{0.5+1}{2} = 0.75$
$f(0.75) = -0.234375$
Como $f(0.75) < 0$, la raíz está en [0.75, 1]

El tercer punto medio
$m = \frac{0.75+1}{2} = 0.875$
$f(0.875) = 1.123046875$
Como $f(0.875) > 0$, la raíz está en [0.75, 0.875].

El cuarto punto medio
$m = \frac{0.75+0.875}{2} = 0.8125$
$f(0.8125) = 0.388427734375$
Como $f(0.8125) > 0$, la raíz está en [0.75, 0.8125]

El quinto punto medio
$m = \frac{0.75+0.8125}{2} = 0.78125$
$f(0.78125) = 0.073272705078125$
Como $f(0.78125) > 0$, la raíz está en [0.75, 0.78125].

Siguiendo este proceso hasta que la diferencia entre los extremos del intervalo sea menor que 10^{-2}, se obtiene: $x \approx 0.5859$

b.[1,3.2]

Verificamos los valores iniciales $f(1) = 2$ y $f(3.2) = -1.488$
Como $f(1) \cdot f(3.25) < 0$, sabemos que hay una raíz en el intervalo [1,3.2]

El primer punto medio
$m = \frac{1+3.2}{2} = 2.1$
$f(2.1) = 1.539$
Como $f(2.1) < 0$, la raíz está en [2.1, 3.2]

El segundo punto medio
$m = \frac{2.1+3.2}{2} = 2.65$
$f(2.65) = -0.027$
Como $f(2.65) < 0$, la raíz está en [2.1, 2.65]

El tercer punto medio
$m = \frac{2.1+2.65}{2} = 2.375$
$f(2.375) = 0.451$
Como $f(2.375) > 0$, la raíz está en [2.375, 2.65].

Siguiendo este proceso hasta que la diferencia entre los extremos del intervalo sea menor que 10^{-2}, se obtiene: $x \approx 3.001$

c.[3.2, 4]

Verificamos los valores iniciales $f(3.2) = -1.488$ y $f(4) = 6$
Como $f(3.2) \cdot f(4) < 0$, sabemos que hay una raíz en el intervalo [3.2, 4]

El primer punto medio
$m = \frac{3.2+4}{2} = 3.6$
$f(3.6) = 0.896$
Como $f(3.6) > 0$, la raíz está en [3.2, 3.6].

El segundo punto medio
$m = \frac{3.2+3.6}{2} = 3.4$
$f(3.4) = -0.177$
Como $f(3.4) < 0$, la raíz está en [3.4, 3.6].

Siguiendo este proceso hasta que la diferencia entre los extremos del intervalo sea menor que 10^{-2}, se obtiene: $x \approx 3.419$

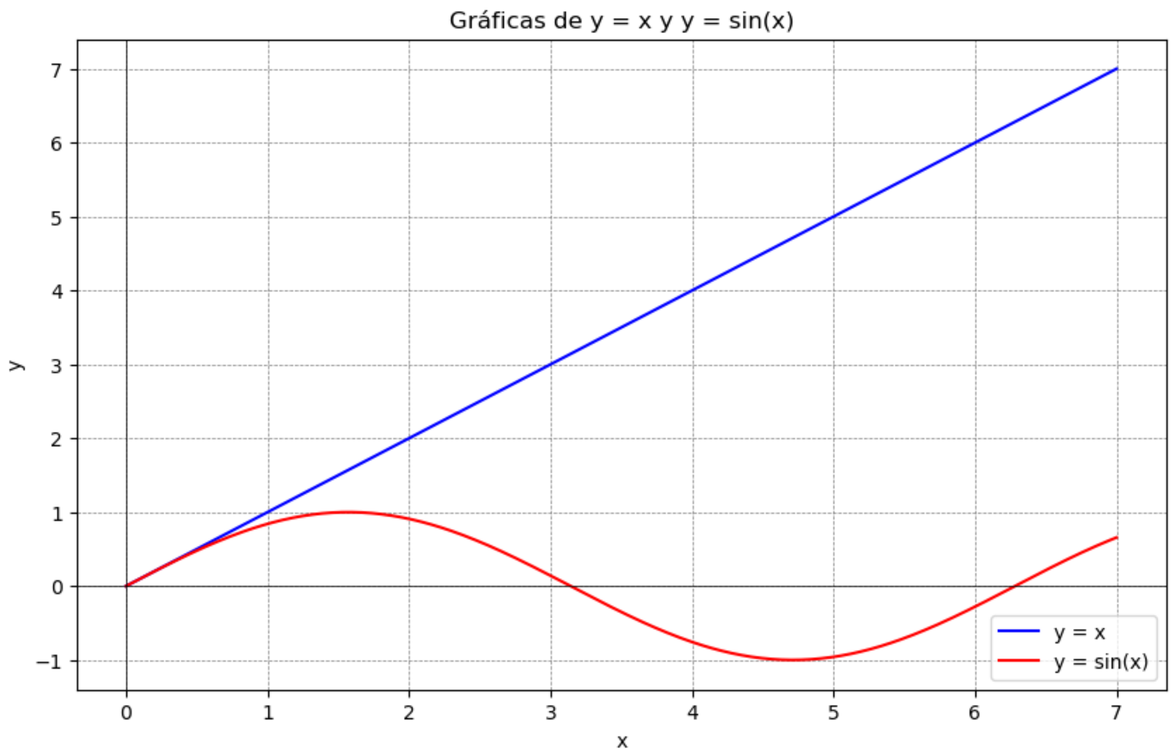
1.2 Pregunta 2

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

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

# Definir la función para la ecuación
def f(x):
    return x - 2 * math.sin(x)

# Método de bisección
def bisection_method(a, b, tol):
    if f(a) * f(b) >= 0:
        print("El método de bisección no puede aplicarse.")
        return None
    c = a
    while (b - a) / 2 > tol:
        c = (a + b) / 2
        if f(c) == 0.0:
            break
        if f(a) * f(c) < 0:
            b = c
        else:
            a = c
    return c

# Definir los extremos del intervalo y la tolerancia
a = 1
b = 2
tol = 1e-5

# Encontrar la raíz usando el método de bisección
root = bisection_method(a, b, tol)
if root is not None:
    print(f"Con el pirmer valor positivo se obtiene : {root:.5f}")

# Gráfica
x = np.linspace(0, 7, 400)
y1 = x
```

```
y2 = np.sin(x)
plt.figure(figsize=(10, 6))
plt.plot(x, y1, label='y = x', color='blue')
plt.plot(x, y2, label='y = sin(x)', color='red')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Gráficas de y = x y y = sin(x)')
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.grid(color='gray', linestyle='--', linewidth=0.5)
plt.legend()
plt.show()
```

```
Con el pirmer valor positivo se obtiene : 1.89549
```



1.3 Pregunta 3

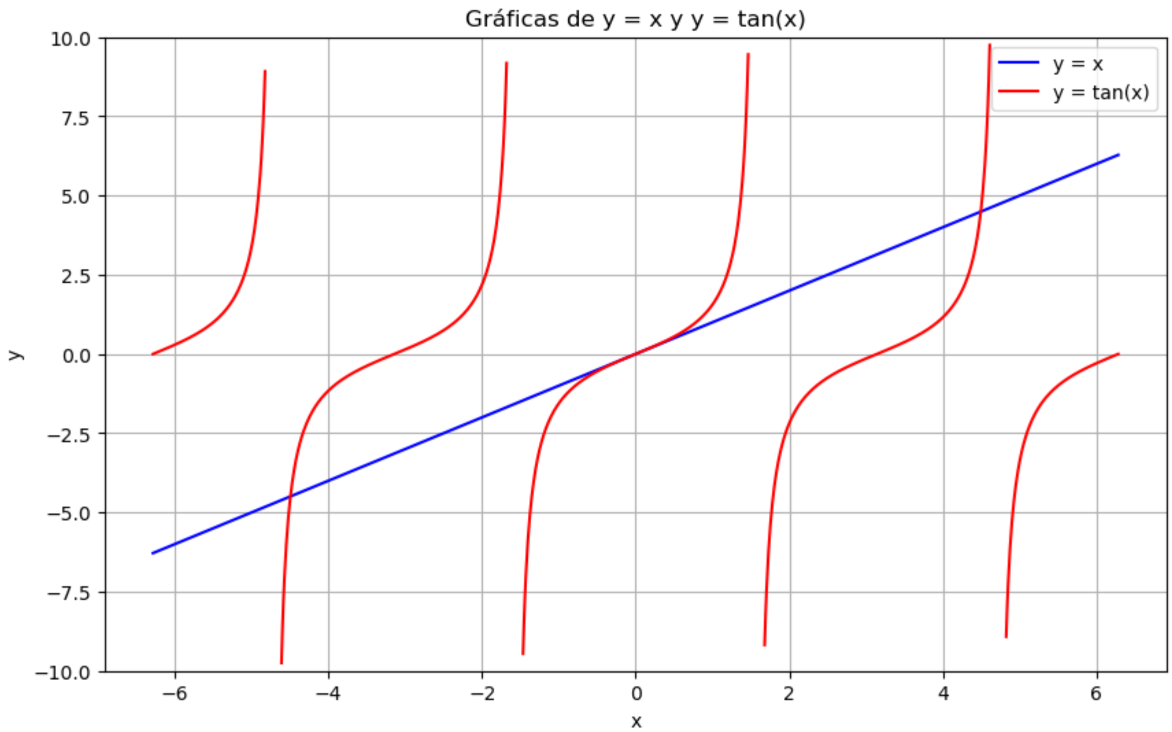
a. Dibuje las gráficas para $y = x$ **y** $y = tanx$
b. Use el método de bisección para encontrar una aproximación dentro de 10^{-5}

para el primer valor positivo de x con $x = tanx$

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

# Definir la función para la ecuación x = tan(x)
def f(x):
    return x - math.tan(x)

# Método de bisección
def bisection_method(a, b, tol):
    if f(a) * f(b) >= 0:
        print("El método de bisección no puede aplicarse.")
        return None
    c = a
    while (b - a) / 2 > tol:
        c = (a + b) / 2
        if f(c) == 0.0:
            break
        if f(a) * f(c) < 0:
            b = c
        else:
            a = c
    return c

# Definir los extremos del intervalo y la tolerancia
a = 4.2
b = 4.6
tol = 1e-5

# Encontrar la raíz usando el método de bisección
root = bisection_method(a, b, tol)
if root is not None:
    print(f"Con el pirmer valor positivo se obtiene: {root:.7f}")

# Parte de la gráfica
x = np.linspace(-2 * np.pi, 2 * np.pi, 1000)
y1 = x
y2 = np.tan(x)

# Evitar los puntos de discontinuidad de tan(x)
y2 = np.where(np.abs(y2) > 10, np.nan, y2)
```

```python
plt.figure(figsize=(10, 6))

# Graficar y = x
plt.plot(x, y1, label='y = x', color='blue')

# Graficar y = tan(x)
plt.plot(x, y2, label='y = tan(x)', color='red')

# Añadir título y etiquetas a los ejes
plt.title('Gráficas de y = x y y = tan(x)')
plt.xlabel('x')
plt.ylabel('y')

# Añadir una rejilla
plt.grid(True)

# Añadir una leyenda
plt.legend()

# Establecer los límites para el eje y
plt.ylim(-10, 10)

# Mostrar la gráfica
plt.show()
```

```
Con el pirmer valor positivo se obtiene: 4.4934204
```



1.4 Pregunta 4

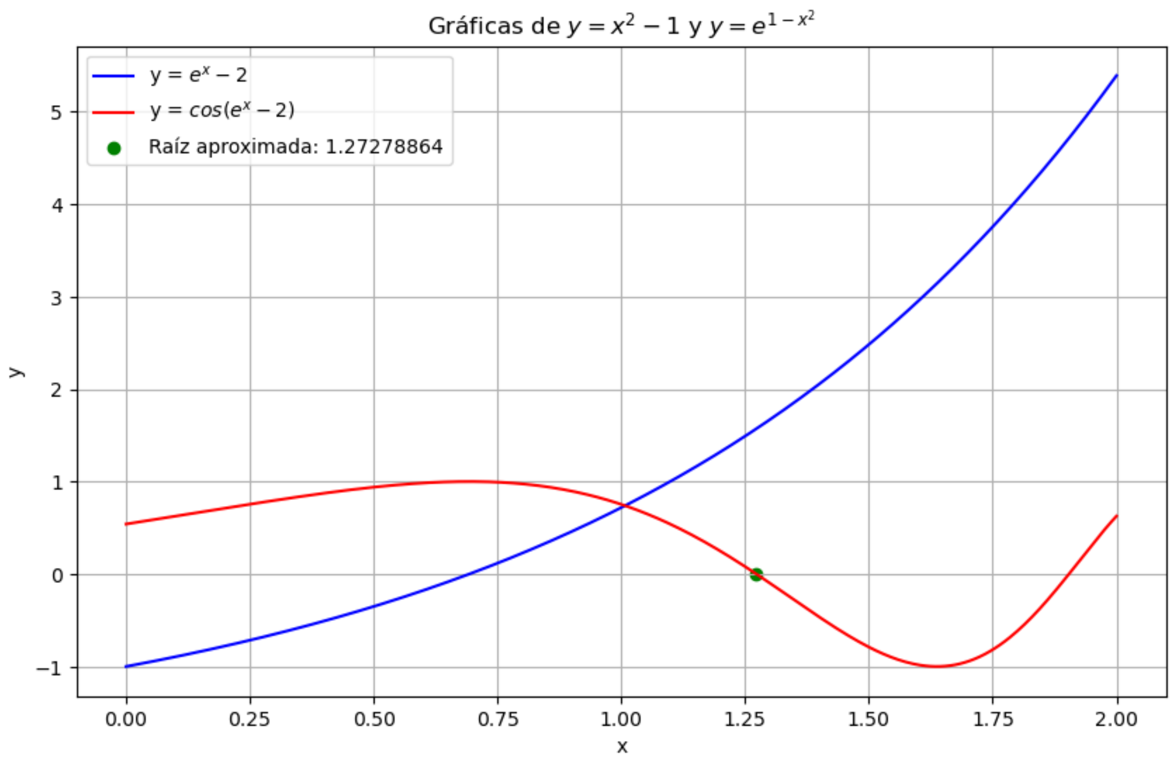
a. Dibuje las gráficas para $y = x^2 - 1$ **y** $y = e^{1-x^2}$
b. Use el metodo de la biseccio para encontrar una aproximaciondentro de 10^{-3} **para el un valor en [-2,0] con** $x^2 - 1 = e^{1-x^2}$

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

# Definir las funciones
def f1(x):
    return np.exp(x) - 2

def f2(x):
    return np.cos(np.exp(x) - 2)

# Método de bisección
def bisection_method(func, a, b, tol):
    if func(a) * func(b) >= 0:
```

```
        print("El método de bisección no puede aplicarse.")
        return None
    c = a
    while (b - a) / 2 > tol:
        c = (a + b) / 2
        if func(c) == 0.0:
            break
        if func(a) * func(c) < 0:
            b = c
        else:
            a = c
    return c

# Definir los extremos del intervalo y la tolerancia
a = 0.5  # Ajustamos el extremo izquierdo del intervalo
b = 1.5  # Ajustamos el extremo derecho del intervalo
tol = 1e-7

# Encontrar la raíz para f2 usando el método de bisección
root = bisection_method(f2, a, b, tol)
if root is not None:
    print(f"La raíz aproximada es: {root:.8f}")

    # Parte de la gráfica
    x = np.linspace(0, 2, 400)  # Ajustamos el rango de valores de x para la gráfica
    y1 = f1(x)
    y2 = f2(x)

    plt.figure(figsize=(10, 6))

    # Graficar y = e^x - 2
    plt.plot(x, y1, label='y = $e^{x} - 2$', color='blue')

    # Graficar y = cos(e^x - 2)
    plt.plot(x, y2, label='y = $cos(e^x - 2)$', color='red')

    # Graficar la raíz encontrada
    plt.scatter(root, f2(root), color='green', label=f'Raíz aproximada: {root:.8f}')

    # Etiquetas y título
    plt.title('Gráficas de $y=x^{2}-1$ y $y=e^{1-x^{2}}$')
    plt.xlabel('x')
```

```
    plt.ylabel('y')

    # Agregar una leyenda
    plt.legend()

    # Mostrar la rejilla
    plt.grid(True)

    # Mostrar la gráfica
    plt.show()
else:
    print("No se encontró ninguna raíz en el intervalo dado.")
```

```
La raíz aproximada es: 1.27278864
```



1.5 Pregunta 5

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

```
def f(x):
    return (x + 3) * ((x + 1) ** 2) * x * ((x - 1) ** 3) * (x - 3)

def biseccion(f, a, b, tol=1e-5, max_iter=100):
    if f(a) * f(b) > 0:
        print(f"El intervalo {intervalo} no contiene una raíz.")
        return None
    c = 0
    while (b - a) / 2.0 > tol and c < max_iter:
        c = (a + b) / 2.0
        if f(c) == 0:
            return c
        elif f(c) * f(a) < 0:
            b = c
        else:
            a = c
        c += 1
    return (a + b) / 2
```

a.$[-1.5, 2.5]$

```
intervalo = (-1.5, 2.5)
a, b = intervalo
root = biseccion(f, a, b)
print(f"\nConverge en el intervalo {intervalo}: {root}")
```

```
El intervalo (-1.5, 2.5) no contiene una raíz.

Converge en el intervalo (-1.5, 2.5): None
```

b.$[-0.5, 2.4]$

```
intervalo = (-0.5, 2.4)
a, b = intervalo
root = biseccion(f, a, b)
print(f"\nConverge en el intervalo {intervalo}: {root}")
```

```
El intervalo (-0.5, 2.4) no contiene una raíz.

Converge en el intervalo (-0.5, 2.4): None
```

c.$[-0.5, 3]$

```
intervalo = (-0.5, 3)
a, b = intervalo
root = biseccion(f, a, b)
print(f"\nConverge en el intervalo {intervalo}: {root}")
```

```
Converge en el intervalo (-0.5, 3): 2.999993324279785
```

d.$[-3, -0.5]$

```
intervalo = (-3, -0.5)
a, b = intervalo
root = biseccion(f, a, b)
print(f"\nConverge en el intervalo {intervalo}: {root}")
```

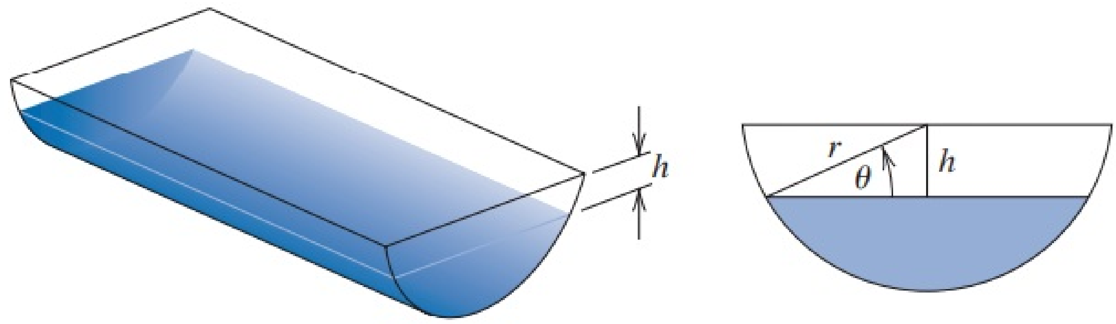
```
Converge en el intervalo (-3, -0.5): -0.5000095367431641
```

2 Ejercicios Aplicados

2.1 Pregunta 1

Un abrevadero de longitud L tiene una sección transversal en forma de semicírculo con radio r. (Consulte la figura adjunta.) Cuando se llena con agua hasta una distancia h a partir de la parte superior, el volumen V de agua es:

$$V = L\left[0.5\pi^2 - r^2 \arcsin\left(\frac{h}{r}\right) - h\left(r^2 - h^2\right)^{1/2}\right]$$



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

```
import math

# Datos
L = 10
r = 1
V = 12.4

def f(h):
    return L * (0.5 * math.pi * r**2 - r**2 * math.asin(h/r) - h * (r**2 - h**2)**0.5) - V

def bisection_method(func, a, b, tol=0.01):
    if func(a) * func(b) > 0:
        print("No se puede garantizar la convergencia del método en el intervalo dado.")
        return None
    while (b - a) / 2.0 > tol:
        c = (a + b) / 2.0
        if func(c) == 0:
            return c
        elif func(c) * func(a) < 0:
            b = c
        else:
            a = c
    return (a + b) / 2

a = 0
b = r
```

```
h = bisection_method(f, a, b)
print("La profundidad del agua en el abrevadero es:", round(h, 3), "cm")
```

```
La profundidad del agua en el abrevadero es: 0.164 cm
```

2.2 Pregunta 2

Un objeto que cae verticalmente a través del aire está sujeto a una resistencia viscosa, así como a la fuerza de gravedad. Suponga que un objeto con masa m cae desde una altura x_0 y que la altura del objeto después de t segundos es

$$s(t) = s_0 - \frac{mg}{k}t + \frac{m^2g}{k^2}\left(1 - e^{-\frac{kt}{m}}\right)$$

donde $g = 9.81m/s^2$y k representa el coeficiente de la resistencia del aire en Ns/m. Suponga $s0 = 300m$, $m = 0.25kg$ y $k = 0.1Ns/m$. Encuentre, dentro de 0.01 , el tiempo que tarda un cuarto de kg en golpear el piso.

```
import math

# Constants
s0 = 300  # m
m = 0.25  # kg
g = 9.81  # m/s^2
k = 0.1   # Ns/m

# Función para calcular s(t)
def s(t):
    return s0 - (m*g/k)*t + (m**2 * g / k**2) * (1 - math.exp(-k*t/m))

# Método de bisección para encontrar la raíz de s(t) = 0
def bisection_method(func, a, b, tol=1e-2):
    if func(a) * func(b) >= 0:
        print("No se puede aplicar el método de bisección.")
        return None

    c = a
    while (b - a) / 2 > tol:
        c = (a + b) / 2
        if func(c) == 0.0:
            break
```

```
        elif func(a) * func(c) < 0:
            b = c
        else:
            a = c

    return c

# Intervalo inicial para el método de bisección
a = 0
b = 20  # Tiempo suficientemente grande para que el objeto caiga al suelo

# Encontrar la raíz usando el método de bisección
root = bisection_method(s, a, b)

# Imprimir el resultado
if root is not None:
    print(f"El tiempo que tarda en golpear el piso es aproximadamente {root:.2f} segundos.")
else:
    print("No se encontró una raíz dentro de la tolerancia especificada.")
```

```
El tiempo que tarda en golpear el piso es aproximadamente 14.71 segundos.
```

3 Ejercicios Teoricos

3.1 Pregunta 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.

```
def f(x):
    return x**3- x-1

def biseccion(f,a, b, tol=1e-4):
    if f(a) * f(b) >= 0:
        print("El intervalo [a, b] no contiene una raíz")
        return None
    iter = 0
    while (b - a) / 2.0 > tol:
```

```
        c = (a + b) / 2.0
        if f(c) == 0:
            return c
        elif f(c) * f(a) < 0:
            b = c
        else:
            a = c
        iter += 1
    return (a + b) / 2, iter

a = 1
b=2

raiz, iteraciones = biseccion(f, a, b)

if raiz is not None:
    print(f"La raíz aproximada de x^3 - x - 1 en el intervalo [{a}, {b}] es {raiz:.5f}")
    print("Número de iteraciones necesarias:", iteraciones)
else:
    print(f"No se encontró una raíz en el intervalo [{a}, {b}]")
```

```
La raíz aproximada de x^3 - x - 1 en el intervalo [1, 2] es 1.32477
Número de iteraciones necesarias: 13
```

3.2 Pregunta 2

La función definida por $f(x) = sin \pi x$ tiene ceros en cada entero. Muestre cuando $-1 < a < 0$ y $2 < b < 3$, el método de bisección converge a

a. $0, si a + b < 2$

b. $2, si a + b > 2$

c. $1, si a + b = 2$

```
import math

def f(x):
    return math.sin(math.pi * x)

def bisection_method(a, b, tol=1e-6):
```

```python
    n = 0
    while (b - a) / 2 > tol:
        midpoint = (a + b) / 2
        if f(midpoint) == 0:
            return midpoint, n
        elif f(a) * f(midpoint) < 0:
            b = midpoint
        else:
            a = midpoint
        n += 1
    return (a + b) / 2, n

def find_zero(a, b, tol=1e-6):
    midpoint = (a + b) / 2
    if midpoint < 1:
        return 0, a, b
    elif midpoint > 1:
        return 2, a, b
    else:
        return 1, a, b

# Parámetros iniciales
a_values = [-0.5, -0.5, -0.5]
b_values = [2.5, 2.5, 2.5]
sum_values = [1.5, 3.5, 2]

for a, b, sum_ab in zip(a_values, b_values, sum_values):
    if sum_ab < 2:
        expected_zero = 0
    elif sum_ab > 2:
        expected_zero = 2
    else:
        expected_zero = 1

    root, iterations = bisection_method(a, b)
    zero, _, _ = find_zero(a, b)

    print(f"Para a + b = {sum_ab}:")
    print(f"  Cero esperado: {expected_zero}")
    print(f"  Cero encontrado por bisección: {zero}")
    print(f"  Raíz aproximada: {root} después de {iterations} iteraciones")
    print()
```

```
Para a + b = 1.5:
  Cero esperado: 0
  Cero encontrado por bisección: 1
  Raíz aproximada: 2.384185791015625e-07 después de 21 iteraciones

Para a + b = 3.5:
  Cero esperado: 2
  Cero encontrado por bisección: 1
  Raíz aproximada: 2.384185791015625e-07 después de 21 iteraciones

Para a + b = 2:
  Cero esperado: 1
  Cero encontrado por bisección: 1
  Raíz aproximada: 2.384185791015625e-07 después de 21 iteraciones
```