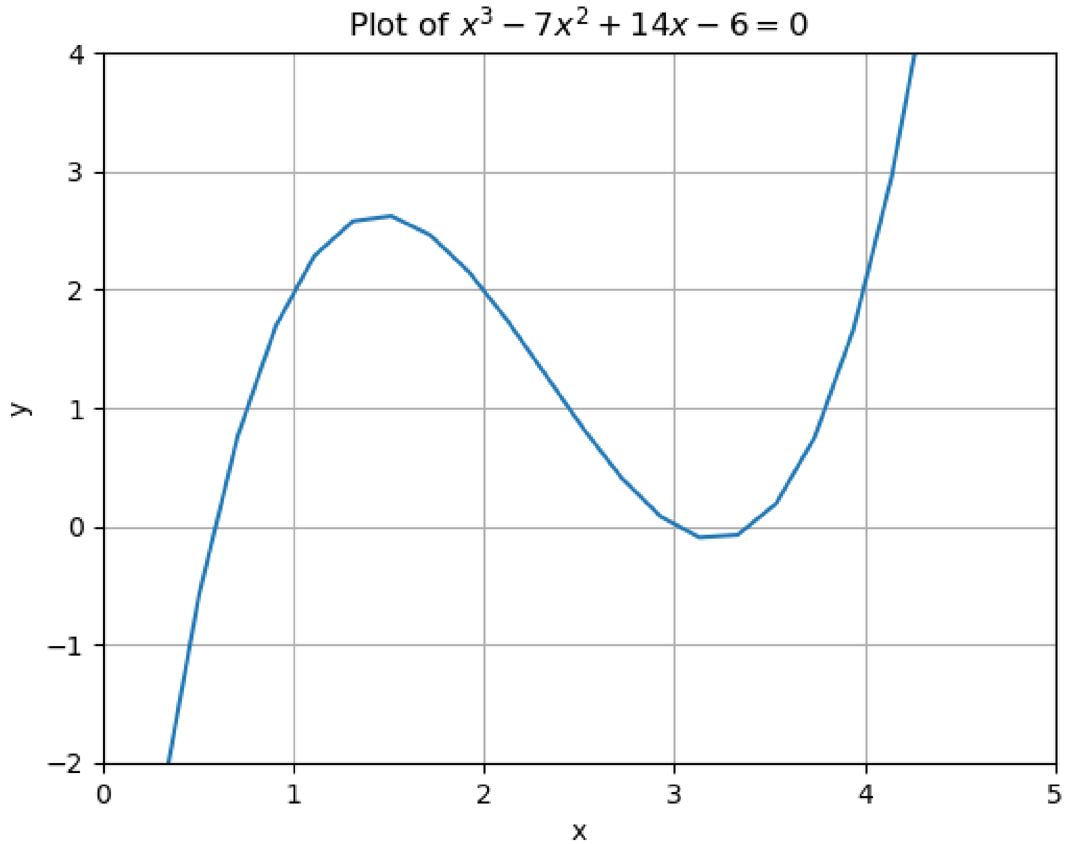
Ejercicios: Bisección

David Morales

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.

Con el objetivo de obtener una mejor visión del ejercicio, se procedió con el proceso de graficación.

```
import numpy as np

import matplotlib.pyplot as plt

def equation(x:float)->float:
    return (x**3 - 7*x**2 + 14*x - 6)

x = np.linspace(-10, 10, 100)

y = equation(x)

plt.plot(x, y)

plt.xlabel('x')
plt.ylabel('y')
plt.title('Plot of $ x^{3} - 7x^{2} + 14x - 6 = 0$')
ax = plt.gca()
ax.set_ylim([-2, 4])
ax.set_xlim([0, 5])
plt.grid(True)
plt.show()
```



El siguiente método aplica la bisección a la función dada en el ejercicio.

```python
from typing import Callable

# Función para determinar el signo
def get_sign(x: float) -> int:
    return 1 if x > 0 else (-1 if x < 0 else 0)

# Implementación del método de bisección
def biseccion(a: float, b: float, *, func: Callable[[float], float], tol: float, N: int) ->
    i = 1
    if a >= b:
        raise ValueError("Intervalo no válido: 'a' debe ser menor que 'b'")

    f_a = func(a)
    for i in range(N):
        midpoint = (a + b) / 2
        f_mid = func(midpoint)
```

```
        if f_mid == 0 or abs(b - a) / 2 < tol:
            return midpoint, a, b, i
        if get_sign(f_a) * get_sign(f_mid) > 0:
            a, f_a = midpoint, f_mid
        else:
            b = midpoint
    return midpoint, a, b, i
```

a. Como resultado de la ejecución se obtiene el siguiente par ordenado: $[0, 1]$

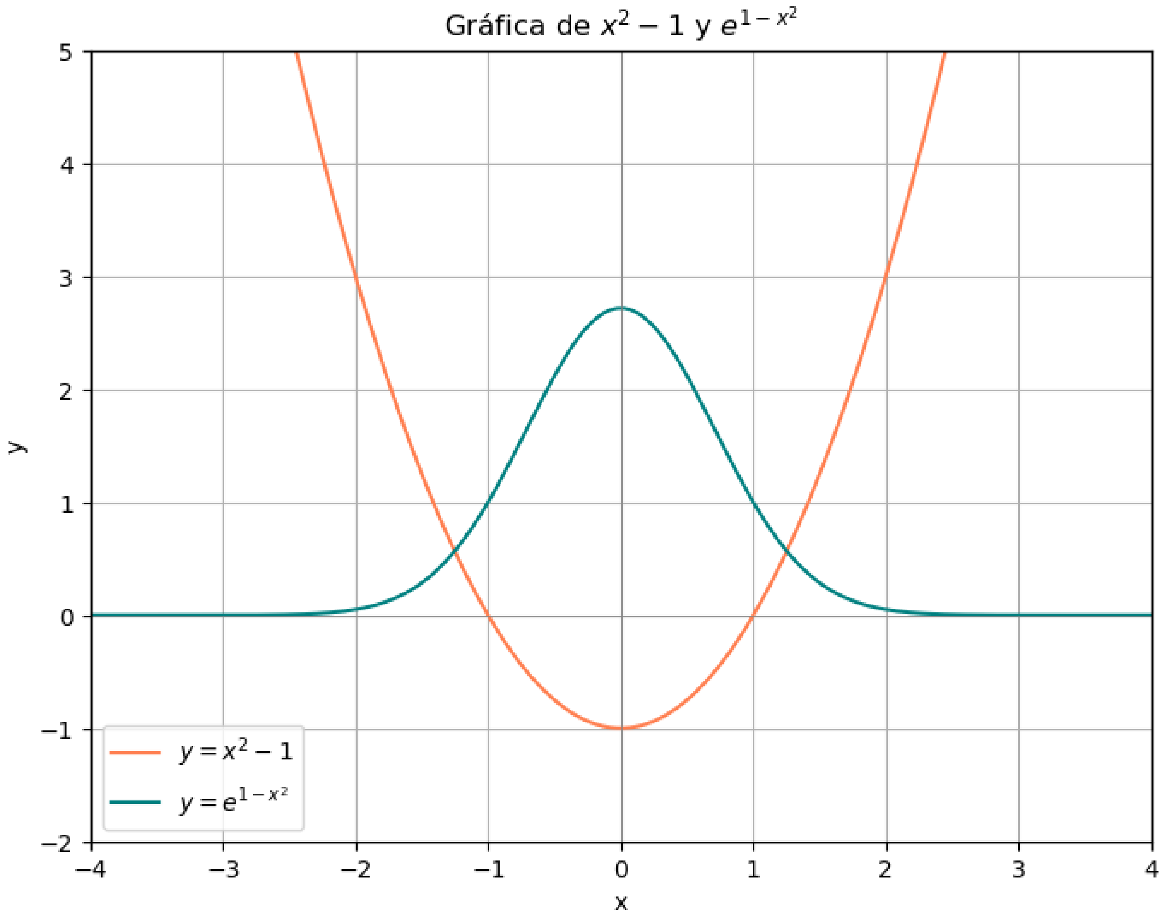
```
import math
from typing import Callable

# Parámetros
a = 0
b = 1
tol = 10**(-2)
L = 10
r = 1
V_dado = 12.4

# Definir la función de volumen
def equation(h):
    V_calculado = L * (0.5 * math.pi * r**2 - r**2 * math.asin(h / r) - h * math.sqrt(r**2 -
    return V_calculado - V_dado

# Función para el método de bisección
def bisection(a: float, b: float, *, equation: Callable[[float], float], tol: float, N: int)
    i = 1
    if a >= b:
        raise ValueError("Intervalo no válido: 'a' debe ser menor que 'b'")

    f_a = equation(a)
    for i in range(N):
        midpoint = (a + b) / 2
        f_mid = equation(midpoint)
        if f_mid == 0 or abs(b - a) / 2 < tol:
            return midpoint, a, b, i
        if f_a * f_mid > 0:
            a, f_a = midpoint, f_mid
        else:
            b = midpoint
```

```python
    return midpoint, a, b, i

# Ejecutar la función de bisección
resultado = bisection(a=a, b=b, equation=equation, tol=tol, N=20)

# Imprimir el resultado
print(f"En el rango [{a}, {b}], la raíz encontrada en la iteración {resultado[3]} con precisi
```

```
En el rango [0, 1], la raíz encontrada en la iteración 6 con precisión 1e-02 es: 0.1640625
```

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

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

# Definir las ecuaciones
def quadratic_eq(x: float) -> float:
    return x**2 - 1

def exponential_eq(x: float) -> float:
    return np.exp(1 - x**2)

# Generar datos
x_vals = np.linspace(-5, 5, 200)
y1_vals = quadratic_eq(x_vals)
y2_vals = exponential_eq(x_vals)

# Gráfico
plt.figure(figsize=(8, 6))
plt.plot(x_vals, y1_vals, label=r'$y = x^2 - 1$', color='coral')
plt.plot(x_vals, y2_vals, label=r'$y = e^{1 - x^2}$', color='teal')
plt.xlabel('x')
plt.ylabel('y')
plt.title(r'Gráfica de $x^2 - 1$ y $e^{1 - x^2}$')
plt.axhline(0, color='gray', linewidth=0.5)
plt.axvline(0, color='gray', linewidth=0.5)
plt.legend()
plt.grid(True)
plt.ylim([-2, 5])
plt.xlim([-4, 4])
plt.show()
```

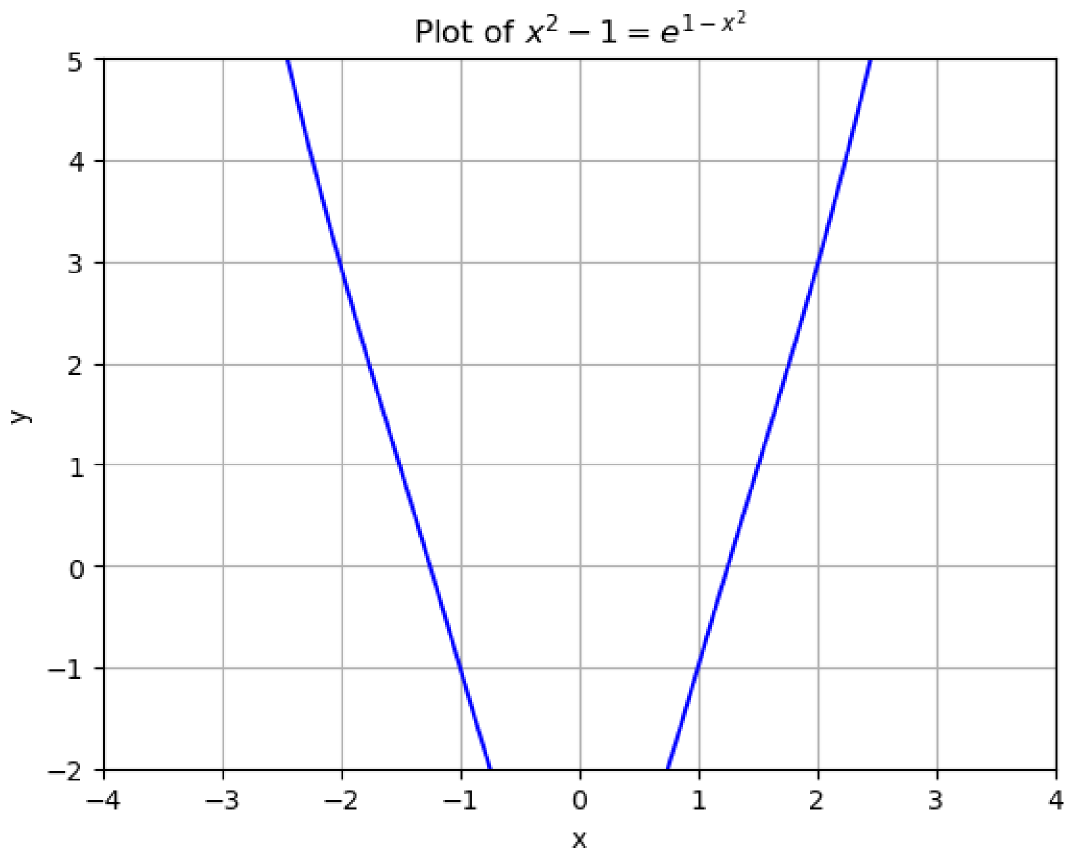


4.b. Use el método de bisección para encontrar una aproximación dentro de 10^{-3} para un valor de $[-2, 0]$ con $x^2 - 1 = e^{1-x^2}$

Se grafica la función.

```
def eq(x):
    return ((x**2) - 1)-(np.exp(1-x**2))


x = np.linspace(-5, 5, 100)

y = eq(x)

plt.plot(x, y, label = '$x^{2} - 1 = e^{1-x^{2}}$', color = 'blue')

plt.xlabel('x')
```

```
plt.ylabel('y')
plt.title('Plot of $x^{2} - 1 = e^{1-x^{2}}$')
ax = plt.gca()
ax.set_ylim([-2, 5])
ax.set_xlim([-4, 4])
plt.grid(True)
plt.show()
```



Aplicando el método de la bisección

```
a = -2
b = 0
tol = 10**(-3)

result= bisection(a = a, b=b,equation=eq,tol = tol, N = 20)
print("En el rango ["+str(a)+","+str(b)+"], en la iteración nº: "+str(result[3])+" se encont
      " dentro de la precisión de "+format(tol, ".0e")+ " es: "+str(result[0]))
```

```
En el rango [-2,0], en la iteración n°: 10 se encontró que la raíz dentro de la precisión de
```

Ejercicios Aplicados

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

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

Suponga que = 10 , = 1 y =12.4 . Encuentre la profundidad del agua en el abrevadero dentro de 0.01 *cm*

Datos del Ejercicio:

- Tolerancia: $0.01cm$
- Intervalo: $[h_{min}, h_{max}]$, es decir: $[0, 1]$

```python
import math
from typing import Callable

L = 10
r = 1
V_dado = 12.4
tol = 0.01

def f_h(h):
    V_calculado = L * (0.5 * math.pi * r**2 - r**2 * math.asin(h / r) - h * math.sqrt(r**2 -
    return V_calculado - V_dado

def bisection(a: float, b: float, *, equation: Callable[[float], float], tol: float, N: int)
    i = 1
    if a >= b:
        raise ValueError("Intervalo no válido: 'a' debe ser menor que 'b'")

    f_a = equation(a)
    for i in range(N):
        midpoint = (a + b) / 2
        f_mid = equation(midpoint)
        if f_mid == 0 or abs(b - a) / 2 < tol:
            return midpoint, a, b, i
        if f_a * f_mid > 0:
            a, f_a = midpoint, f_mid
```

```
        else:
            b = midpoint
    return midpoint, a, b, i

result = bisection(a=0, b=r, equation=f_h, tol=tol, N=20)

print(f"En el intervalo [{0}, {r}], la raíz encontrada en la iteración {result[3]} con precis
```

```
En el intervalo [0, 1], la raíz encontrada en la iteración 6 con precisión 1e-02 es: 0.164062
```

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 cae desde una altura y que la altura del objeto después de segundos es

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

donde $(g = 9.81, \mathbf{m/s}^2)$ y (k) representa el coeficiente de la resistencia del aire en $(\mathbf{Ns/m})$. Suponga $(s_0 = 300, \mathbf{m})$, $(m = 0.25\,\mathbf{kg})$ y $(k = 0.1, \mathbf{Ns/m})$. Encuentre, dentro de $(0.01\,\mathbf{segundos})$, el tiempo que tarda un cuarto de kg en golpear el piso.

```
s0 = 300
m = 0.25
k = 0.1
g = 9.81
tol = 0.01

def f_t(t):
    s_t = s0 - (m * g / k) * t + (m**2 * g / k**2) * (1 - math.exp(-k * t / m))
    return s_t

result = bisection(a=0,b=s0,equation=f_t,tol=tol,N=20)

print("En el rango ["+str(0)+","+str(s0)+"], en la iteración n°: "+str(result[3])+" se encon
      " dentro de la precisión de "+format(tol, ".0e")+ " es: "+str(result[0]) + " seg.")
```

```
En el rango [0,300], en la iteración n°: 14 se encontró que la raíz dentro de la precisión de
```

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.

```
a = 1
b = 2
tol = 10**(-4)
def equation3(x):
    return (x**(3)-x-1)

result = bisection(a=a,b=b,equation=equation3,tol=tol,N=20)

print("Después de " + str(result[3]+1) + " iteraciones la solución aproximada en la precisión
```

```
Después de 14 iteraciones la solución aproximada en la precisión de 1e-04 es: 1.324768066406
```

GitHub: git@github.com:DavidME1604/MetodosNumericos2024B_MoralesDavid.git