

Escuela Politécnica Nacional

Nombre: Kevin Eduardo Garcia Rodriguez

Tema: [Tarea 04] Ejercicios Unidad 01-B-Biseccion

Repositorio de GitHub: <https://github.com/Nattyrd/Metodos-Numericos-2025B>

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. $[3.2, 4]$

```
In [8]: import numpy as np
import matplotlib.pyplot as plt
```

```
In [12]: def funcion_ejemplo(x):
        """Define la función cuya raíz se busca."""
        return x**3 - 7*x**2 + 14*x - 6 # Ejemplo con funcion cubica
```

```
In [8]: def biseccion(f, a, b, tol, max_iter=50):
        print("\nIteraciones del método de Biseccion:\n")
        print("{:<10}{:<15}{:<20}{:<20}{:<20}".format("Iter", "a", "b", "c", "f(c)"))
        print("-"*85)
        # 1. Comprobar si el intervalo es válido (cambio de signo)
        if f(a) * f(b) >= 0:
            print(f"Error: La función no cambia de signo en el intervalo [{a}, {b}].")
            return None

        # 2. Inicializar variables
        iteracion = 0
        c = a # Inicializamos c (la raíz aproximada)

        # Bucle principal
        while abs(b - a) > tol and iteracion < max_iter:

            # 3. Calcular el punto medio
            c = (a + b) / 2

            # 4. Verificar si c es la raíz
            if f(c) == 0:
                print(f"Raíz exacta encontrada en {c} en la iteración {iteracion}.")
                return c

            # 5. Redefinir el intervalo
            elif f(a) * f(c) < 0:
                # La raíz está en el subintervalo izquierdo [a, c]
                b = c
                iteracion += 1
            else:
                # La raíz está en el subintervalo derecho [c, b]
```

```
a = c
iteracion += 1

print("{:<10}{:<15.6f}{:<20.10f}{:<20.10f}{:<20.10f}".format(iteracion,

# 6. Devolver la mejor aproximación
c = (a + b) / 2
print(f"El método converge a la raíz aproximada: {c} después de {iteracion}")
print(f"El error máximo es: {(b - a) / 2}")
print()
return c
```

```
In [38]: # Búsqueda de raíces en diferentes intervalos
print("\n=== Búsqueda en [0, 1] ===")
raiz = biseccion(funcion_ejemplo, 0, 1, tol=1e-2)
print(f"Raíz aproximada: {raiz:.6f}")

print("\n=== Búsqueda en [1, 3.2] ===")
raiz = biseccion(funcion_ejemplo, 1, 3.2, tol=1e-2)
print(f"Raíz aproximada: {raiz:.6f}")

print("\n=== Búsqueda en [3.2, 4] ===")
raiz = biseccion(funcion_ejemplo, 3.2, 4, tol=1e-2)
print(f"Raíz aproximada: {raiz:.6f}")
```

=== Búsqueda en [0, 1] ===

Iteraciones del método de Biseccion:

Iter	a	b	c	f(c)

1	0.500000	1.0000000000	0.5000000000	-0.6250000000
2	0.500000	0.7500000000	0.7500000000	0.9843750000
3	0.500000	0.6250000000	0.6250000000	0.2597656250
4	0.562500	0.6250000000	0.5625000000	-0.1618652344
5	0.562500	0.5937500000	0.5937500000	0.0540466309
6	0.578125	0.5937500000	0.5781250000	-0.0526237488
7	0.578125	0.5859375000	0.5859375000	0.0010313988

El método converge a la raíz aproximada: 0.58203125 después de 7 iteraciones.

El error máximo es: 0.00390625

Raíz aproximada: 0.582031

=== Búsqueda en [1, 3.2] ===

Iteraciones del método de Biseccion:

Iter	a	b	c	f(c)

1	2.100000	3.2000000000	2.1000000000	1.7910000000
2	2.650000	3.2000000000	2.6500000000	0.5521250000
3	2.925000	3.2000000000	2.9250000000	0.0858281250
4	2.925000	3.0625000000	3.0625000000	-0.0544433594
5	2.993750	3.0625000000	2.9937500000	0.0063278809
6	2.993750	3.0281250000	3.0281250000	-0.0265207214
7	2.993750	3.0109375000	3.0109375000	-0.0106969337
8	2.993750	3.0023437500	3.0023437500	-0.0023327508

El método converge a la raíz aproximada: 2.9980468750000004 después de 8 iteraciones.

El error máximo es: 0.004296875000000089

Raíz aproximada: 2.998047

=== Búsqueda en [3.2, 4] ===

Iteraciones del método de Biseccion:

Iter	a	b	c	f(c)

1	3.200000	3.6000000000	3.6000000000	0.3360000000
2	3.400000	3.6000000000	3.4000000000	-0.0160000000
3	3.400000	3.5000000000	3.5000000000	0.1250000000
4	3.400000	3.4500000000	3.4500000000	0.0461250000
5	3.400000	3.4250000000	3.4250000000	0.0130156250
6	3.412500	3.4250000000	3.4125000000	-0.0019980469
7	3.412500	3.4187500000	3.4187500000	0.0053815918

El método converge a la raíz aproximada: 3.4156250000000004 después de 7 iteraciones.

El error máximo es: 0.0031249999999998224

Raíz aproximada: 3.415625

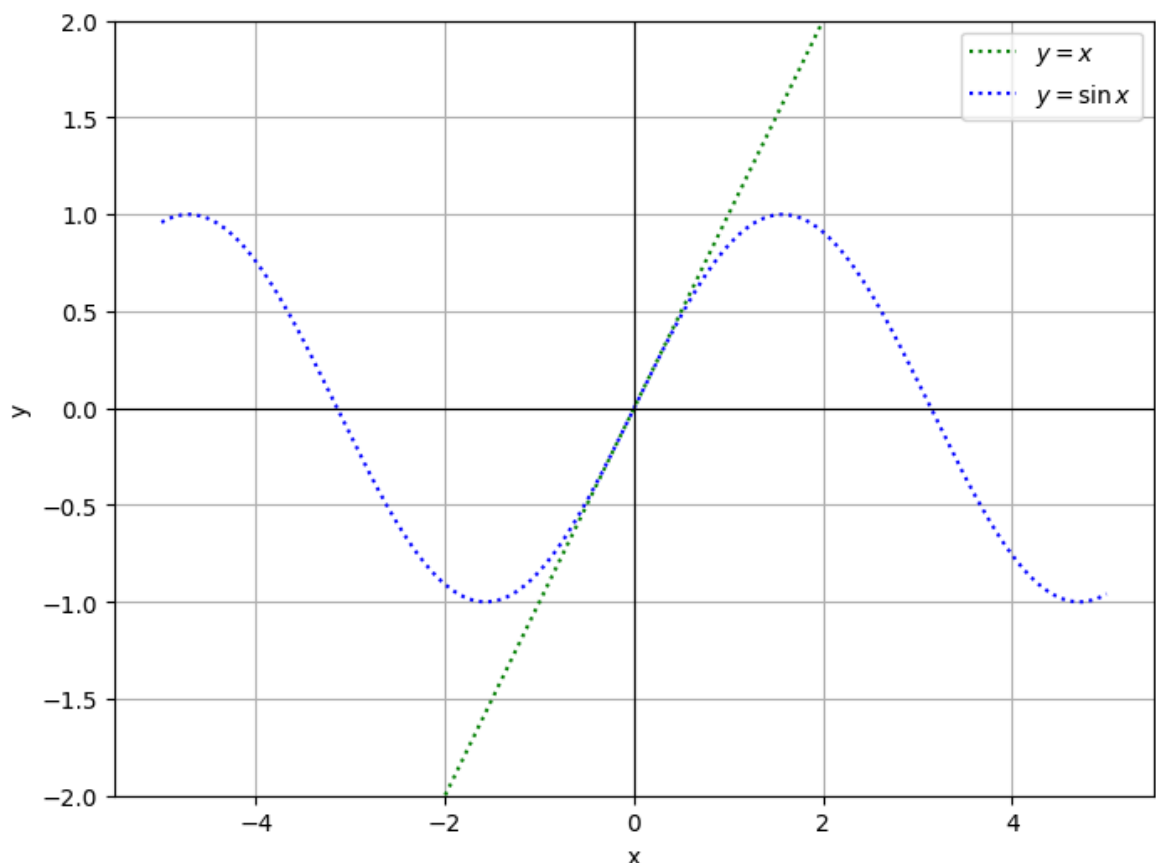
2. a. Dibuje las gráficas para $y = x$ y $y = \sin x$.

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

```
In [5]: import numpy as np
x=np.linspace(-5, 5, 800)
y1 = x
y2 = np.sin(x)

plt.figure(figsize=(8, 6)) #tamaño de la figura
plt.plot(x, y1, label='$y = x$', color='green', linestyle=':')
plt.plot(x, y2, label='$y = \sin x$', color='blue', linestyle=':')
plt.xlabel('x')
plt.ylabel('y')

plt.grid(True)
plt.axhline(0, color='black', linewidth=0.8) # Eje X
plt.axvline(0, color='black', linewidth=0.8) # Eje Y
plt.legend()
plt.ylim(-2, 2) # Limita el eje Y para evitar asíntotas demasiado altas
plt.show()
```



```
In [12]: # Gráfica de la función
x=np.linspace(-10, 4, 800)
y1 = 2 * np.sin(x) - x

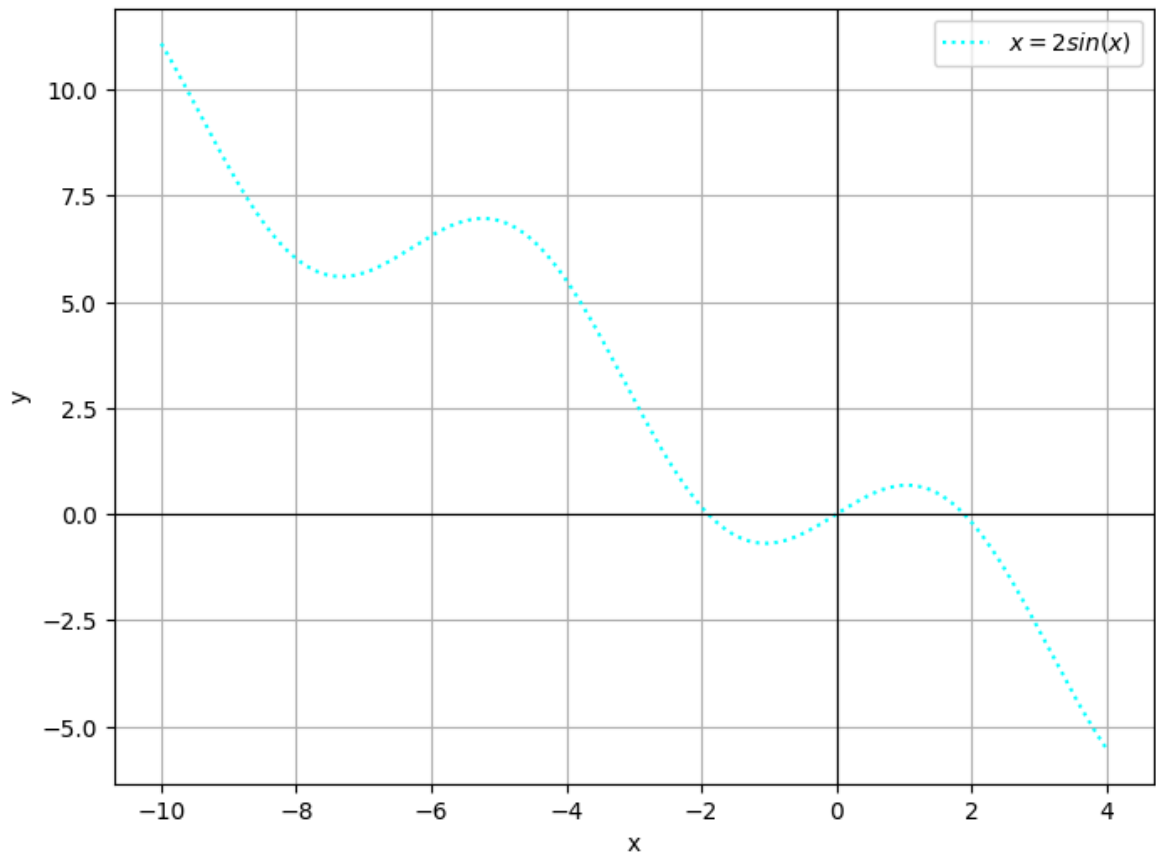
plt.figure(figsize=(8, 6)) #tamaño de la figura
plt.plot(x, y1, label='$x = 2\sin(x)$', color='cyan', linestyle=':')
plt.xlabel('x')
```

```
plt.ylabel('y')

plt.grid(True)
plt.axhline(0, color='black', linewidth=0.8) # Eje X
plt.axvline(0, color='black', linewidth=0.8) # Eje Y
plt.legend()
plt.show()

# b)

def f2(x):
    return 2 * np.sin(x) - x
```



```
In [13]: raiz = biseccion(f2, 1, 2, tol=1e-5)
print(f"Raíz aproximada: {raiz:.6f}")
```

Iteraciones del método de Biseccion:

Iter	a	b	c	f(c)

1	1.500000	2.00000000000	1.50000000000	0.4949899732
2	1.750000	2.00000000000	1.75000000000	0.2179718937
3	1.875000	2.00000000000	1.87500000000	0.0331715632
4	1.875000	1.93750000000	1.93750000000	-0.0704714383
5	1.875000	1.90625000000	1.90625000000	-0.0177278826
6	1.890625	1.90625000000	1.89062500000	0.0079535957
7	1.890625	1.89843750000	1.89843750000	-0.0048293554
8	1.894531	1.89843750000	1.89453125000	0.0015765862
9	1.894531	1.89648437500	1.89648437500	-0.0016227704
10	1.894531	1.89550781250	1.89550781250	-0.0000221883
11	1.895020	1.89550781250	1.89501953125	0.0007774250
12	1.895264	1.89550781250	1.89526367190	0.0003776749
13	1.895386	1.89550781250	1.89538574222	0.0001777574
14	1.895447	1.89550781250	1.89544677733	0.0000777881
15	1.895477	1.89550781250	1.89547729490	0.0000278008
16	1.895493	1.89550781250	1.89549255370	0.0000028065
17	1.895493	1.89550018310	1.89550018310	-0.0000096908

El método converge a la raíz aproximada: 1.8954963684082031 después de 17 iteraciones.

El error máximo es: 3.814697265625e-06

Raíz aproximada: 1.895496

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 x con $x = \tan x$

```
In [14]: ## a)

x = np.linspace(-10, 4, 800)

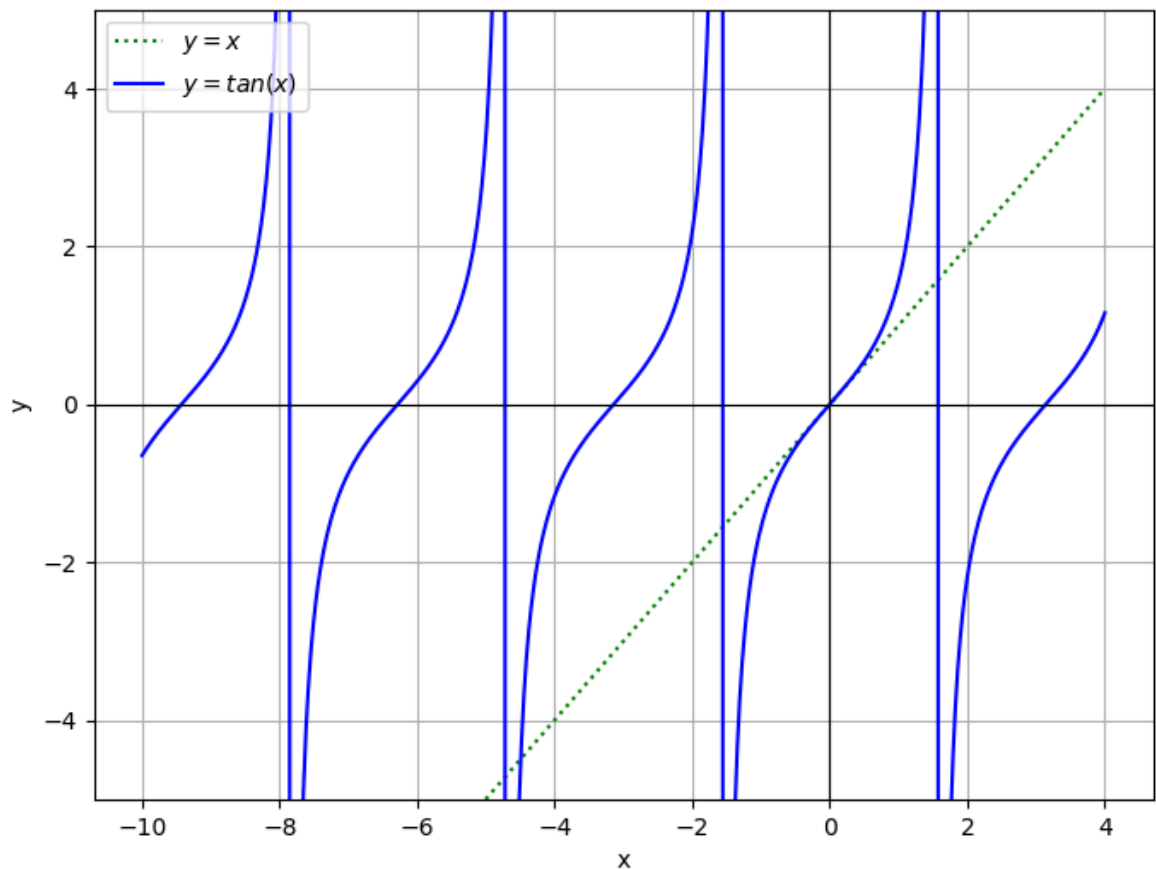
# Definir las funciones
y1 = x
y2 = np.tan(x)

# Graficar
plt.figure(figsize=(8, 6))
plt.plot(x, y1, label='$y = x$', color='green', linestyle=':')
plt.plot(x, y2, label='$y = \tan(x)$', color='blue')

plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)

# Ejes
plt.axhline(0, color='black', linewidth=0.8)
plt.axvline(0, color='black', linewidth=0.8)

plt.legend()
plt.ylim(-5, 5) # Limita el eje Y para evitar asintotas demasiado altas
plt.show()
```



```
In [15]: # Definir el rango de x y la función
x = np.linspace(-6, 6, 800)
y1 = np.tan(x) - x

# Crear la figura
plt.figure(figsize=(8, 6))
plt.plot(x, y1, label='$y = \tan(x) - x$', color='cyan')
plt.xlabel('x')
plt.ylabel('y')

# Añadir ejes y cuadrícula
plt.grid(True)
plt.axhline(0, color='black', linewidth=0.8) # Eje X
plt.axvline(0, color='black', linewidth=0.8) # Eje Y

# Encontrar las raíces (aproximadas)
# Usamos np.diff para detectar cambios de signo en y1
roots_indices = np.where(np.diff(np.sign(y1)))[0]
roots_x = x[roots_indices]
roots_y = y1[roots_indices]

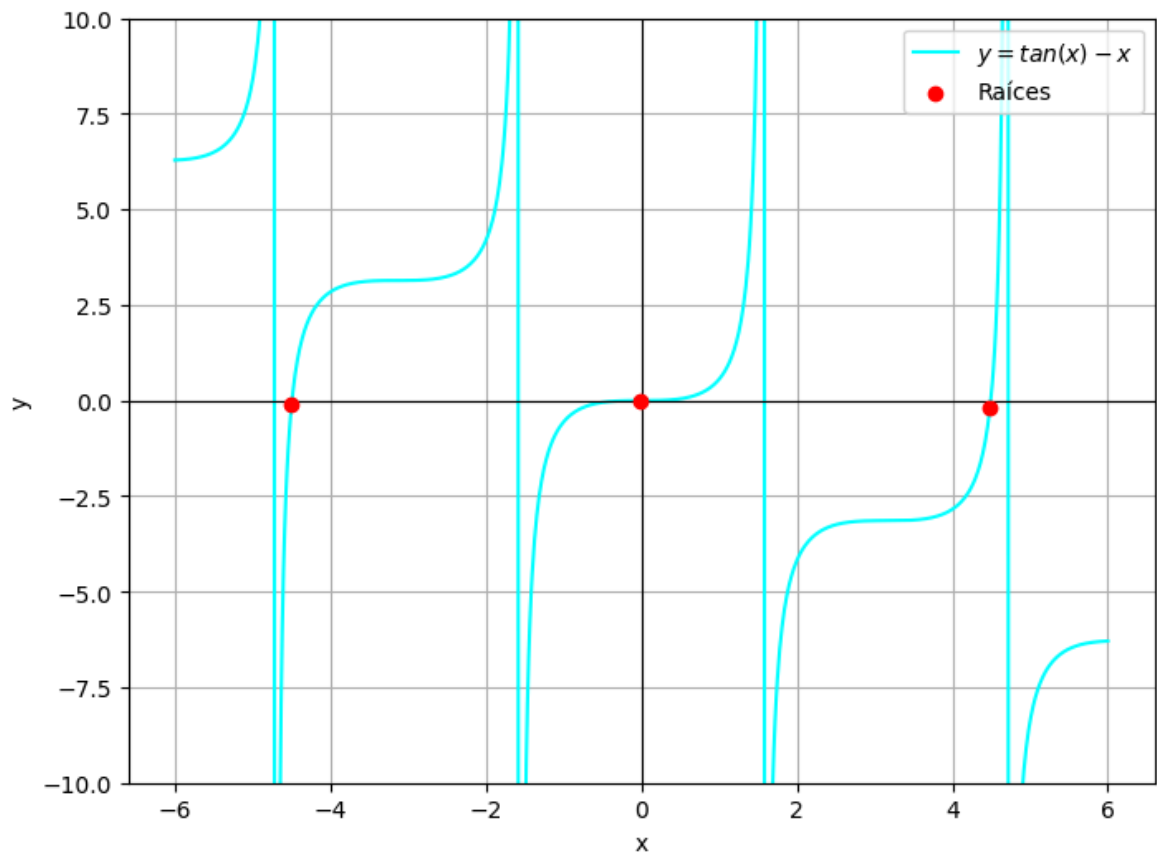
# Marcar las raíces con puntos rojos
plt.scatter(roots_x, roots_y, color='red', label='Raíces', zorder=5)

# Ajustar límites y leyenda
plt.ylim(-10, 10)
plt.legend()

plt.show()

# b)
```

```
def f3(x):
    return np.tan(x) - x
```



```
In [16]: raiz3= biseccion(f3, 4, 4.5, tol=1e-5)
```

Iteraciones del método de Biseccion:

Iter	a	b	c	f(c)

1	4.250000	4.5000000000	4.2500000000	-2.2436909721
2	4.375000	4.5000000000	4.3750000000	-1.5243878650
3	4.437500	4.5000000000	4.4375000000	-0.8917623434
4	4.468750	4.5000000000	4.4687500000	-0.4458526791
5	4.484375	4.5000000000	4.4843750000	-0.1749484129
6	4.492188	4.5000000000	4.4921875000	-0.0245308310
7	4.492188	4.4960937500	4.4960937500	0.0548924199
8	4.492188	4.4941406250	4.4941406250	0.0148138712
9	4.493164	4.4941406250	4.4931640625	-0.0049489846
10	4.493164	4.4936523438	4.4936523438	0.0049096656
11	4.493408	4.4936523438	4.4934082031	-0.0000253349
12	4.493408	4.4935302734	4.4935302734	0.0024407442
13	4.493408	4.4934692383	4.4934692383	0.0012073496
14	4.493408	4.4934387207	4.4934387207	0.0005909187
15	4.493408	4.4934234619	4.4934234619	0.0002827697
16	4.493408	4.4934158325	4.4934158325	0.0001287119

El método converge a la raíz aproximada: 4.493412017822266 después de 16 iteraciones.

El error máximo es: 3.814697265625e-06

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 $x^2 - 1 = e^{1-x^2}$.

```
In [17]: # Definir el rango de x y la función
x = np.linspace(-6, 6, 800)
y1 = np.exp(1-x**2) - x**(2) +1

# Crear la figura
plt.figure(figsize=(8, 6))
plt.plot(x, y1, label='$y = e^{1-x^2} - x^2 +1$', color='green')
plt.xlabel('x')
plt.ylabel('y')

# Añadir ejes y cuadrícula
plt.grid(True)
plt.axhline(0, color='black', linewidth=0.8) # Eje X
plt.axvline(0, color='black', linewidth=0.8) # Eje Y

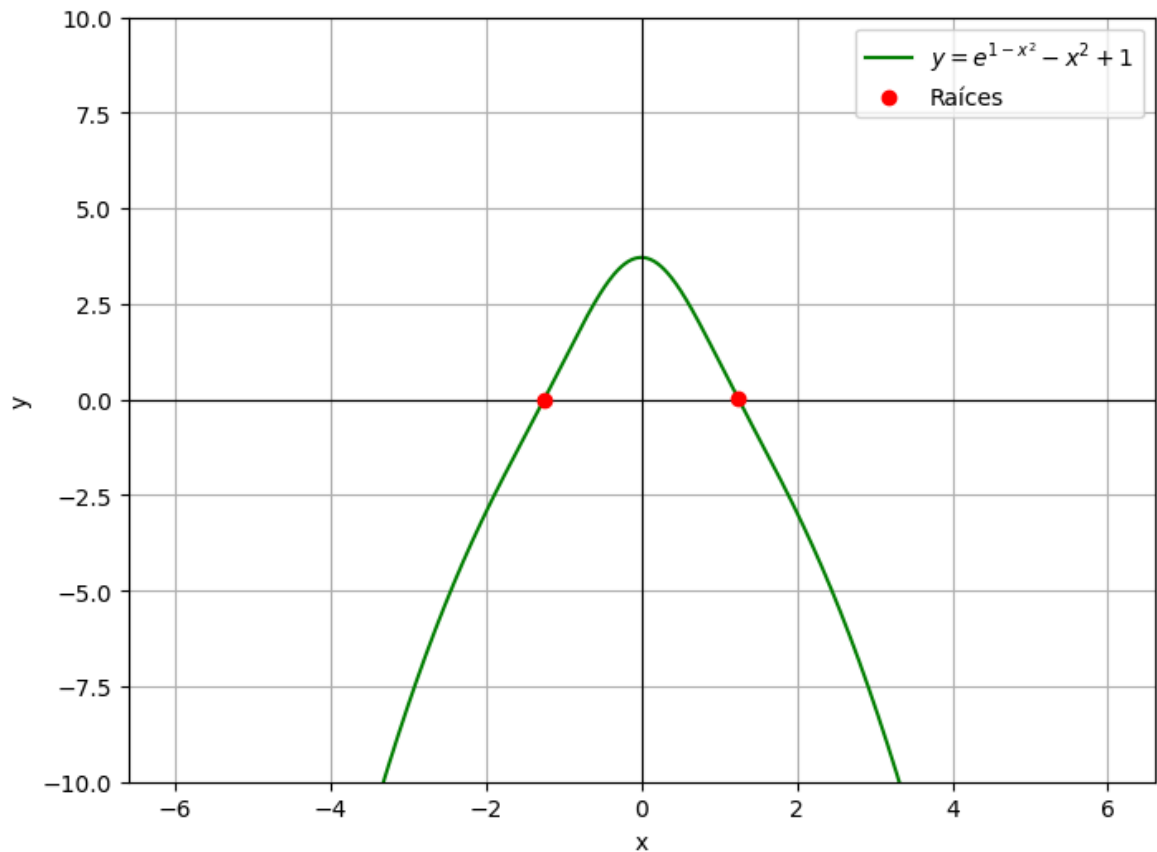
# Encontrar las raíces (aproximadas)
# Usamos np.diff para detectar cambios de signo en y1
roots_indices = np.where(np.diff(np.sign(y1)))[0]
roots_x = x[roots_indices]
roots_y = y1[roots_indices]

# Marcar las raíces con puntos rojos
plt.scatter(roots_x, roots_y, color='red', label='Raíces', zorder=5)

# Ajustar límites y leyenda
plt.ylim(-10, 10)
plt.legend()

plt.show()

def f4(x):
    return np.exp(1-x**2) - x**(2) +1
```



```
In [18]: raiz4 = biseccion(f4, -2, 0, tol=1e-5)
print(f"Raíz aproximada: {raiz4:.6f}")
```

Iteraciones del método de Biseccion:

Iter	a	b	c	f(c)

1	-2.000000	-1.0000000000	-1.0000000000	1.0000000000
2	-1.500000	-1.0000000000	-1.5000000000	-0.9634952031
3	-1.500000	-1.2500000000	-1.2500000000	0.0072828247
4	-1.375000	-1.2500000000	-1.3750000000	-0.4802258269
5	-1.312500	-1.2500000000	-1.3125000000	-0.2371952141
6	-1.281250	-1.2500000000	-1.2812500000	-0.1151529543
7	-1.265625	-1.2500000000	-1.2656250000	-0.0539856148
8	-1.257812	-1.2500000000	-1.2578125000	-0.0233641610
9	-1.253906	-1.2500000000	-1.2539062500	-0.0080438730
10	-1.251953	-1.2500000000	-1.2519531250	-0.0003813268
11	-1.251953	-1.2509765625	-1.2509765625	0.0034505481
12	-1.251953	-1.2514648438	-1.2514648438	0.0015345604
13	-1.251953	-1.2517089844	-1.2517089844	0.0005766043
14	-1.251953	-1.2518310547	-1.2518310547	0.0000976356
15	-1.251892	-1.2518310547	-1.2518920898	-0.0001418464
16	-1.251862	-1.2518310547	-1.2518615723	-0.0000221056
17	-1.251862	-1.2518463135	-1.2518463135	0.0000377649
18	-1.251862	-1.2518539429	-1.2518539429	0.0000078297

El método converge a la raíz aproximada: -1.2518577575683594 después de 18 iteraciones.

El error máximo es: 3.814697265625e-06

Raíz aproximada: -1.251858

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?

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

```
In [19]: def f5(x):  
         return (x+3)*(x+1)**2 *x*(x-1**3)**3 *(x-3)
```

```
In [20]: # Intervalos  
raiz = biseccion(f5, -1.5,2.5, 1e-5)  
print(f"\nRaíz aproximada: {raiz}")  
  
# Intervalos  
raiz = biseccion(f5, -0.5,2.4, 1e-5)  
print(f"\nRaíz aproximada: {raiz}")  
  
# Intervalos  
raiz = biseccion(f5, -0.5,3, 1e-5)  
print(f"\nRaíz aproximada: {raiz}")  
  
# Intervalos  
raiz = biseccion(f5, -0.5,3.5, 1e-5)  
print(f"\nRaíz aproximada: {raiz}")
```

Iteraciones del método de Biseccion:

Iter	a	b	c	f(c)

Error: La función no cambia de signo en el intervalo $[-1.5, 2.5]$.

Raíz aproximada: None

Iteraciones del método de Biseccion:

Iter	a	b	c	f(c)

Error: La función no cambia de signo en el intervalo $[-0.5, 2.4]$.

Raíz aproximada: None

Iteraciones del método de Biseccion:

Iter	a	b	c	f(c)

Error: La función no cambia de signo en el intervalo $[-0.5, 3]$.

Raíz aproximada: None

Iteraciones del método de Biseccion:

Iter	a	b	c	f(c)

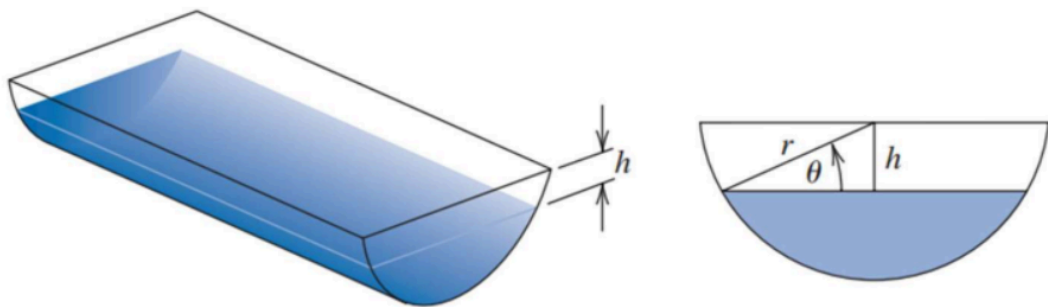
1	1.500000	3.5000000000	1.5000000000	-7.9101562500
2	2.500000	3.5000000000	2.5000000000	-284.2382812500

Raíz exacta encontrada en 3.0 en la iteración 2.

Raíz aproximada: 3.0

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

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



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

Datos:

$$L = 10 \text{ cm}, \quad r = 1 \text{ cm}, \quad V = 12.4 \text{ cm}^3$$

Ecuación del volumen:

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

Sustituyendo los valores:

$$12.4 = 10 \left[0.5\pi(1)^2 - (1)^2 \arcsin(h) - h\sqrt{1 - h^2} \right]$$

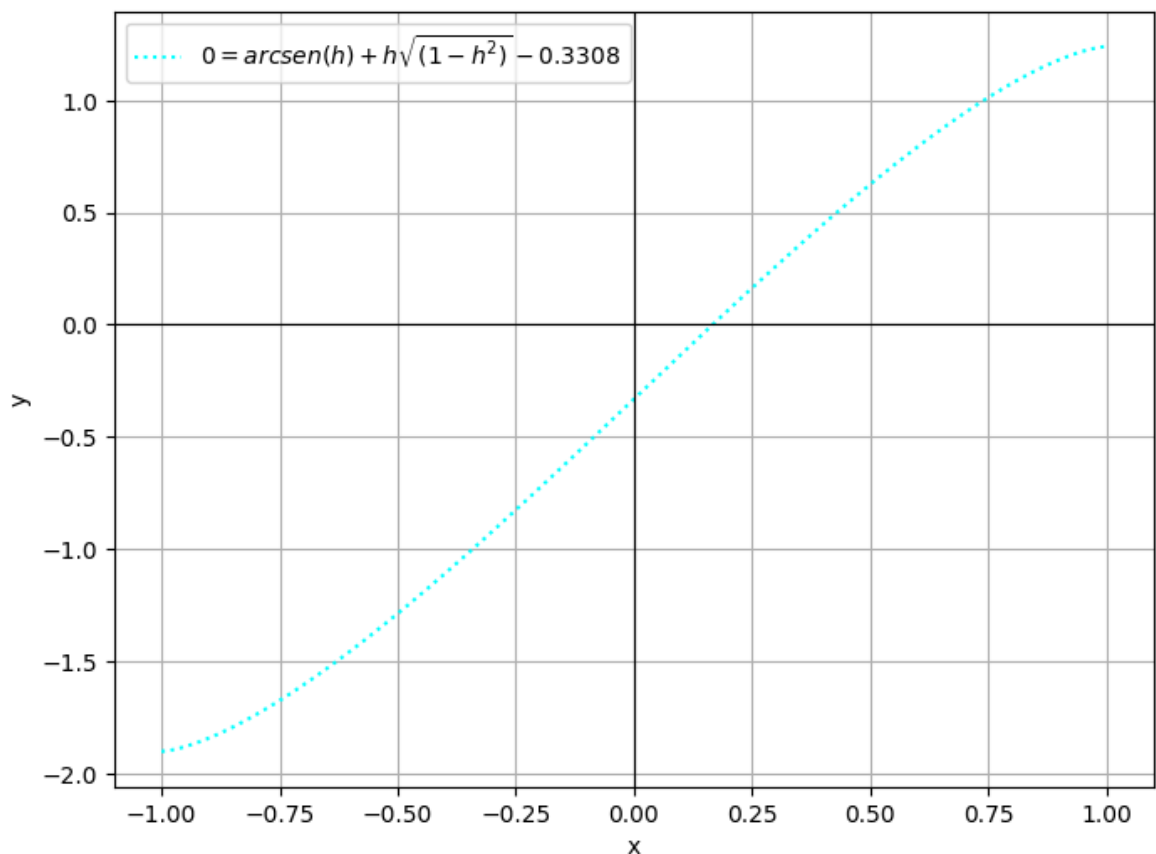
$$12.4 = 10 \left[0.5\pi - \arcsin(h) - h\sqrt{1 - h^2} \right]$$

$$1.24 = 0.5\pi - \arcsin(h) - h\sqrt{1 - h^2}$$

```
In [22]: x=np.linspace(-1, 1, 800)
y1 = np.arcsin(x) + x*(1-x**2)**(1/2) - 0.3308

plt.figure(figsize=(8, 6)) #tamaño de la figura
plt.plot(x, y1, label='$0 = \arcsen(h) + h\sqrt{1 - h^2} - 0.3308$', color='cy')
plt.xlabel('x')
plt.ylabel('y')

plt.grid(True)
plt.axhline(0, color='black', linewidth=0.8) # Eje X
plt.axvline(0, color='black', linewidth=0.8) # Eje Y
plt.legend()
plt.show()
```



```
In [24]: def f6(x):
        return np.arcsin(x) + x*(1-x**2)**(1/2) - 0.3308
```

```
In [26]: raiz6=biseccion(f6, 0, 0.5, 1e-2)
```

Iteraciones del método de Biseccion:

Iter	a	b	c	f(c)
1	0.000000	0.2500000000	0.2500000000	0.1639417143
2	0.125000	0.2500000000	0.1250000000	-0.0814525761
3	0.125000	0.1875000000	0.1875000000	0.0419909992
4	0.156250	0.1875000000	0.1562500000	-0.0195762635
5	0.156250	0.1718750000	0.1718750000	0.0112499662
6	0.164062	0.1718750000	0.1640625000	-0.0041529973

El método converge a la raíz aproximada: 0.16796875 después de 6 iteraciones.

El error máximo es: 0.00390625

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 (S_0) y que la altura del objeto después de (t) segundos es:

$$S(t) = S_0 - \frac{mg}{k} t + \frac{m^2 g}{k^2} (1 - e^{-kt/m})$$

Datos:

$$g = 9.81 \text{ m/s}^2, S_0 = 300 \text{ m}, m = 0.25 \text{ kg}, k = 0.1 \text{ N s/m}, \text{dentro de } 0.01 \text{ segundos}$$

Ecuación del volumen:

$$S(t) = S_0 - \frac{mg}{k} t + \frac{m^2 g}{k^2} (1 - e^{-kt/m})$$

Sustituyendo los valores:

$$0 = 300 - \frac{(0.25)(9.81)}{0.1} t + \frac{(0.25)^2(9.81)}{0.1^2} (1 - e^{-0.1t/0.25})$$

$$0 = 300 - \frac{2.4525}{0.1} t + \frac{0.613125}{0.01} (1 - e^{-0.4t})$$

Ecuación final a resolver:

$$0 = 300 - 24.525t + 61.3125(1 - e^{-0.4t})$$

Encuentre, con una precisión de 0.01 segundos, el tiempo que tarda un objeto de un cuarto de kilogramo en golpear el piso.

La ecuación a resolver es:

$$0 = S_0 - \frac{mg}{k} t + \frac{m^2 g}{k^2} (1 - e^{-kt/m})$$

```
In [28]: def f7(x):
         return 300 - 24.525 * x + 61.3125 * (1 - np.e**(-0.4 * x))
```

```
In [29]: raiz7 = biseccion(f7, 14, 15, 1e-2)
```

Iteraciones del método de Biseccion:

Iter	a	b	c	f(c)

1	14.500000	15.0000000000	14.5000000000	5.5143730497
2	14.500000	14.7500000000	14.7500000000	-0.5992122105
3	14.625000	14.7500000000	14.6250000000	2.4578011829
4	14.687500	14.7500000000	14.6875000000	0.9293483059
5	14.718750	14.7500000000	14.7187500000	0.1650813350
6	14.718750	14.7343750000	14.7343750000	-0.2170621366
7	14.718750	14.7265625000	14.7265625000	-0.0259895729

El método converge a la raíz aproximada: 14.72265625 después de 7 iteraciones.
El error máximo es: 0.00390625

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

Número de iteraciones en el método de bisección

El número mínimo de iteraciones necesarias (n) para alcanzar una tolerancia de error (ϵ) se calcula mediante la siguiente fórmula:

$$n \geq \frac{\log\left(\frac{b-a}{\epsilon}\right)}{\log(2)}$$

```
In [30]: import math
         a = 1
         b = 2
         tol = 1e-4

         n = math.ceil(math.log((b - a) / tol, 2))
         print("Número teórico mínimo de iteraciones:", n)
```

Número teórico mínimo de iteraciones: 14

```
In [31]: def f8(x):
         return x**3-x-1
```

```
In [32]: raiz8 = biseccion(f8, 1, 2, 1e-4)
```

Iteraciones del método de Biseccion:

Iter	a	b	c	f(c)

1	1.000000	1.5000000000	1.5000000000	0.8750000000
2	1.250000	1.5000000000	1.2500000000	-0.2968750000
3	1.250000	1.3750000000	1.3750000000	0.2246093750
4	1.312500	1.3750000000	1.3125000000	-0.0515136719
5	1.312500	1.3437500000	1.3437500000	0.0826110840
6	1.312500	1.3281250000	1.3281250000	0.0145759583
7	1.320312	1.3281250000	1.3203125000	-0.0187106133
8	1.324219	1.3281250000	1.3242187500	-0.0021279454
9	1.324219	1.3261718750	1.3261718750	0.0062088296
10	1.324219	1.3251953125	1.3251953125	0.0020366507
11	1.324707	1.3251953125	1.3247070312	-0.0000465949
12	1.324707	1.3249511719	1.3249511719	0.0009947910
13	1.324707	1.3248291016	1.3248291016	0.0004740388
14	1.324707	1.3247680664	1.3247680664	0.0002137072

El método converge a la raíz aproximada: 1.324737548828125 después de 14 iteraciones.

El error máximo es: 3.0517578125e-05

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 $a+b = 2$

```
In [33]: # --- Definir el rango de x y la función ---
x = np.linspace(-1, 3, 800)
y = np.sin(np.pi * x)

# --- Crear la figura y graficar la función ---
plt.figure(figsize=(8, 6))
plt.plot(x, y, color='cyan', label=r'$\sin(\pi x)$')

# --- Etiquetas de ejes ---
plt.xlabel('x')
plt.ylabel('y')

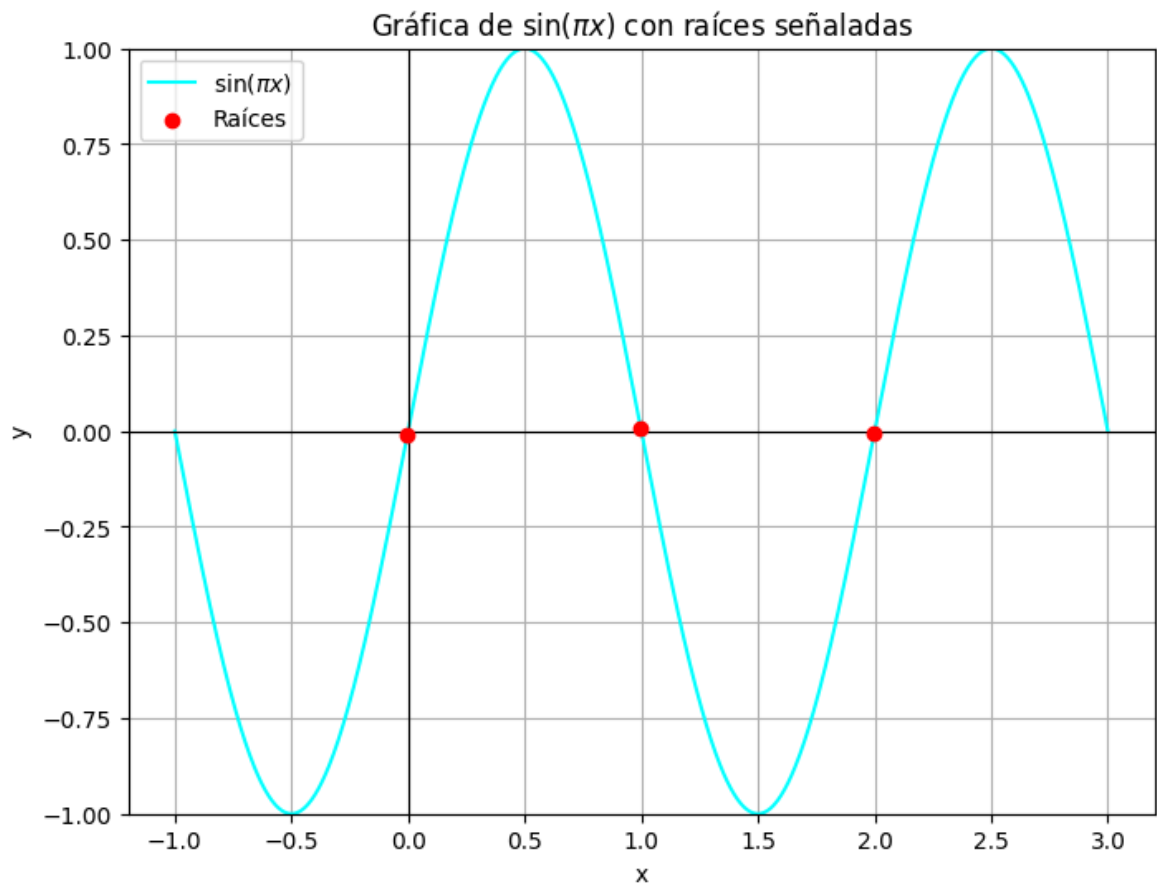
# --- Añadir ejes principales y cuadrícula ---
plt.axhline(0, color='black', linewidth=0.8) # Eje X
plt.axvline(0, color='black', linewidth=0.8) # Eje Y
plt.grid(True)

# --- Encontrar raíces aproximadas ---
roots_indices = np.where(np.diff(np.sign(y)))[0]
roots_x = x[roots_indices]
roots_y = y[roots_indices]

# --- Marcar raíces en la gráfica ---
plt.scatter(roots_x, roots_y, color='red', label='Raíces', zorder=5)
```

```
# --- Ajustes finales ---
plt.ylim(-1, 1)
plt.legend()
plt.title("Gráfica de  $\sin(\pi x)$  con raíces señaladas")

# --- Mostrar la figura ---
plt.show()
```



punto medio inicial:

$$m_o = \frac{a + b}{2}$$

Este punto medio determinará en qué subintervalo se continúa el proceso de bisección, dependiendo del signo de $f(m_0)$ que nos indica hacia dónde se dirige la convergencia.

tal que:

- si $a + b < 2$, $m_0 < 1$ - La raíz más cercana a m_0 es 0
- si $a + b > 2$, $m_0 > 1$ - La raíz más cercana a m_0 es 2
- si $a + b = 2$, $m_0 = 1$ - La raíz directa m_0 es 1

De esta forma usando el punto medio y la gráfica, logramos demostrar si la convergencia es correcta en dichos literales