Tarea4Lenin_Amangandi

November 5, 2025

Tarea N4

Métodos Númericos

Nombre: Lenin Amangandi

Enlace GitHub Tarea 4

0.1 Pregunta 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
import os

def f(x):
    return x**3 - 7*x**2 + 14*x - 6

def biseccion(f, a, b, tol=1e-2, max_iter=100):
    if f(a)*f(b) >= 0:
        print("El método no se puede aplicar en este intervalo.")
        return None

    iter_count = 0
    while (b - a)/2 > tol and iter_count < max_iter:
        c = (a + b)/2
        if f(c) == 0:
            return c
        elif f(a)*f(c) < 0:
            b = c
        else:
            a = c
        iter_count += 1
    return (a + b)/2
```

```
intervalos = [(0, 1), (1, 3.2), (3.2, 4)]

raices = []
for i, (a, b) in enumerate(intervalos):
    raiz = biseccion(f, a, b, tol=1e-2)
    raices.append(raiz)
    print(f"Raíz en el intervalo {a}-{b}: {raiz:.4f}")

x = np.linspace(0, 4, 400)
y = f(x)

plt.figure(figsize=(8,5))
plt.plot(x, y, label='f(x) = x^3 - 7x^2 + 14x - 6')
plt.axhline(0, color='black', linewidth=0.8)
plt.scatter(raices, f(np.array(raices)), color='red', label='Raíces␣
 ↪aproximadas')
plt.title("Pregunta 1: Raíces usando Bisección")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.legend()
plt.grid(True)

ruta_carpeta = r"C:
 ↪\Workspace-Metodos-Numericos-2k25A-\Tareas\Tarea4\ImagenesTarea4"
os.makedirs(ruta_carpeta, exist_ok=True)
ruta_imagen = os.path.join(ruta_carpeta, "pregunta1Grafica.png")

plt.savefig(ruta_imagen)
plt.show()
```

1 Pregunta 2

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

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

x = np.linspace(0, 4, 400)
y = x

plt.figure(figsize=(6,4))
plt.plot(x, y, label='y = x', color='blue')
plt.axhline(0, color='black', linewidth=0.8)
plt.title("Pregunta 2a: y = x")
plt.xlabel("x")
```

```
plt.ylabel("y")
plt.legend()
plt.grid(True)

ruta_carpeta = r"C:
 ↪\Workspace-Metodos-Numericos-2k25A-\Tareas\Tarea4\ImagenesTarea4"
os.makedirs(ruta_carpeta, exist_ok=True)
ruta_imagen = os.path.join(ruta_carpeta, "pregunta2a_y_equals_x.png")
plt.savefig(ruta_imagen)
plt.show()
```

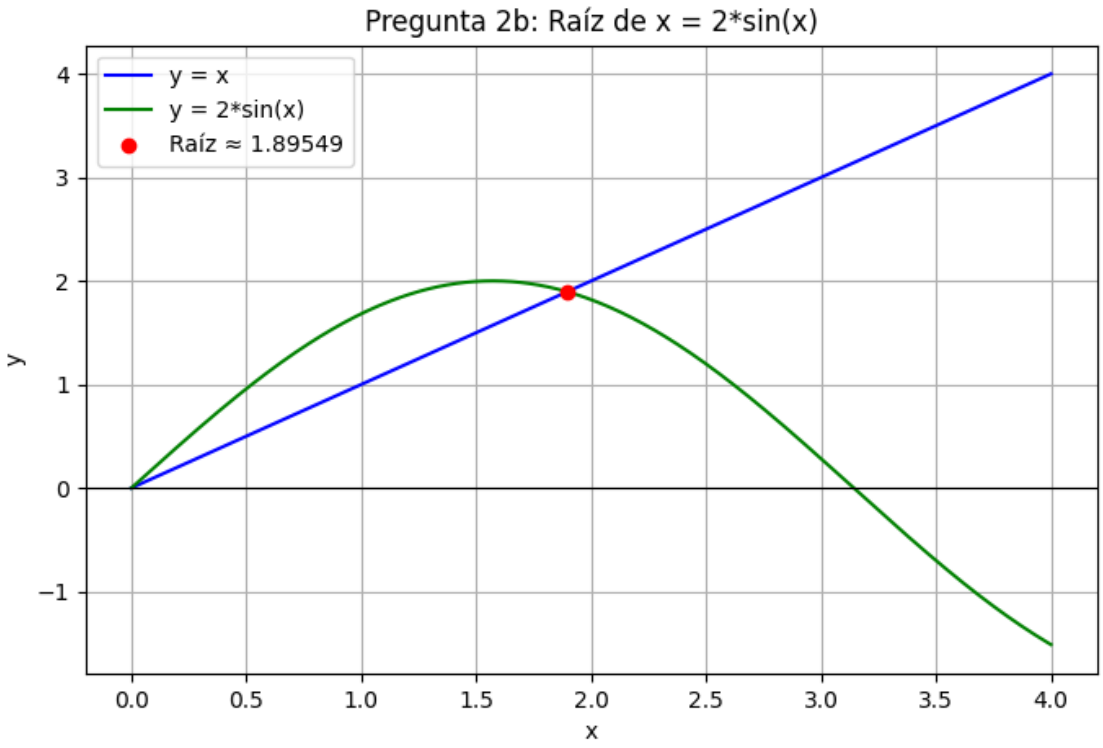
[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 4, 400)
y = np.sin(x)

plt.figure(figsize=(6,4))
plt.plot(x, y, label='y = sin(x)', color='green')
plt.axhline(0, color='black', linewidth=0.8)
plt.title("Pregunta 2a: y = sin(x)")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid(True)

ruta_imagen = os.path.join(ruta_carpeta, "pregunta2a_y_equals_sin.png")
plt.savefig(ruta_imagen)
plt.show()
```

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

[16]:
```
import numpy as np
import matplotlib.pyplot as plt
import os

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

def biseccion(f, a, b, tol=1e-5, max_iter=1000):
    if f(a)*f(b) > 0:
        print("El método no se puede aplicar en este intervalo.")
        return None

    iter_count = 0
    while (b - a)/2 > tol and iter_count < max_iter:
```

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

a, b = 1e-6, np.pi
raiz = biseccion(g, a, b, tol=1e-5)

if raiz is not None:
    print(f"Primera raíz positiva de x = 2*sin(x): {raiz:.5f}")
else:
    print("No se encontró raíz en el intervalo.")

x_vals = np.linspace(0, 4, 400)
y1 = x_vals
y2 = 2*np.sin(x_vals)

plt.figure(figsize=(8,5))
plt.plot(x_vals, y1, label='y = x', color='blue')
plt.plot(x_vals, y2, label='y = 2*sin(x)', color='green')
plt.axhline(0, color='black', linewidth=0.8)
if raiz is not None:
    plt.scatter(raiz, raiz, color='red', zorder=5, label=f'Raíz  {raiz:.5f}')
plt.title("Pregunta 2b: Raíz de x = 2*sin(x)")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid(True)

ruta_carpeta = r"C:
 ↪\Workspace-Metodos-Numericos-2k25A-\Tareas\Tarea4\ImagenesTarea4"
os.makedirs(ruta_carpeta, exist_ok=True)

ruta_imagen = os.path.join(ruta_carpeta, "pregunta2bGrafica.png")

plt.savefig(ruta_imagen)
plt.show()
```

```
Primera raíz positiva de x = 2*sin(x): 1.89549
```

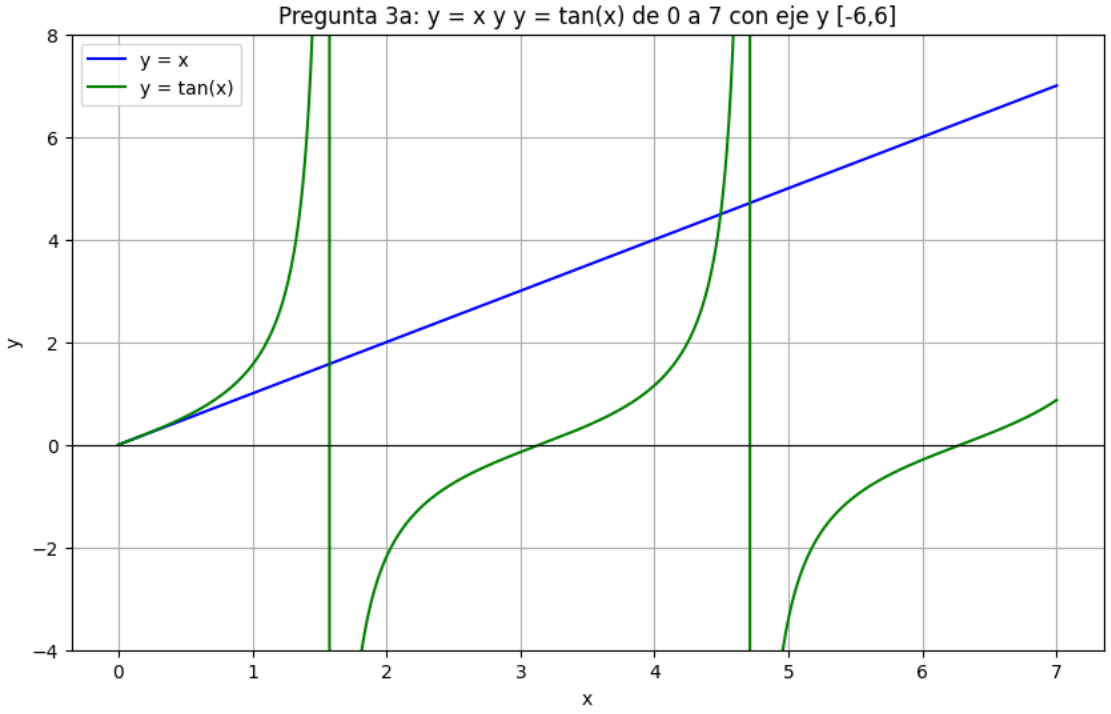


1.1 Pregunta 3

a. Dibuje las gráficas para $y = x$ y $y = \tan x$.

[22]:
```
import numpy as np
import matplotlib.pyplot as plt
import os

x = np.linspace(0, 7, 1000)
y1 = x
y2 = np.tan(x)

plt.figure(figsize=(10,6))
plt.plot(x, y1, label='y = x', color='blue')
plt.plot(x, y2, label='y = tan(x)', color='green')
plt.axhline(0, color='black', linewidth=0.8)

plt.ylim(-4, 8)

plt.title("Pregunta 3a: y = x y y = tan(x) de 0 a 7 con eje y [-6,6]")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
```

```
plt.grid(True)

ruta_carpeta = r"C:
 ↪\Workspace-Metodos-Numericos-2k25A-\Tareas\Tarea4\ImagenesTarea4"
os.makedirs(ruta_carpeta, exist_ok=True)
ruta_imagen = os.path.join(ruta_carpeta, "pregunta3aGrafica_0_a_7_y_-6_6.png")

plt.savefig(ruta_imagen)
plt.show()
```

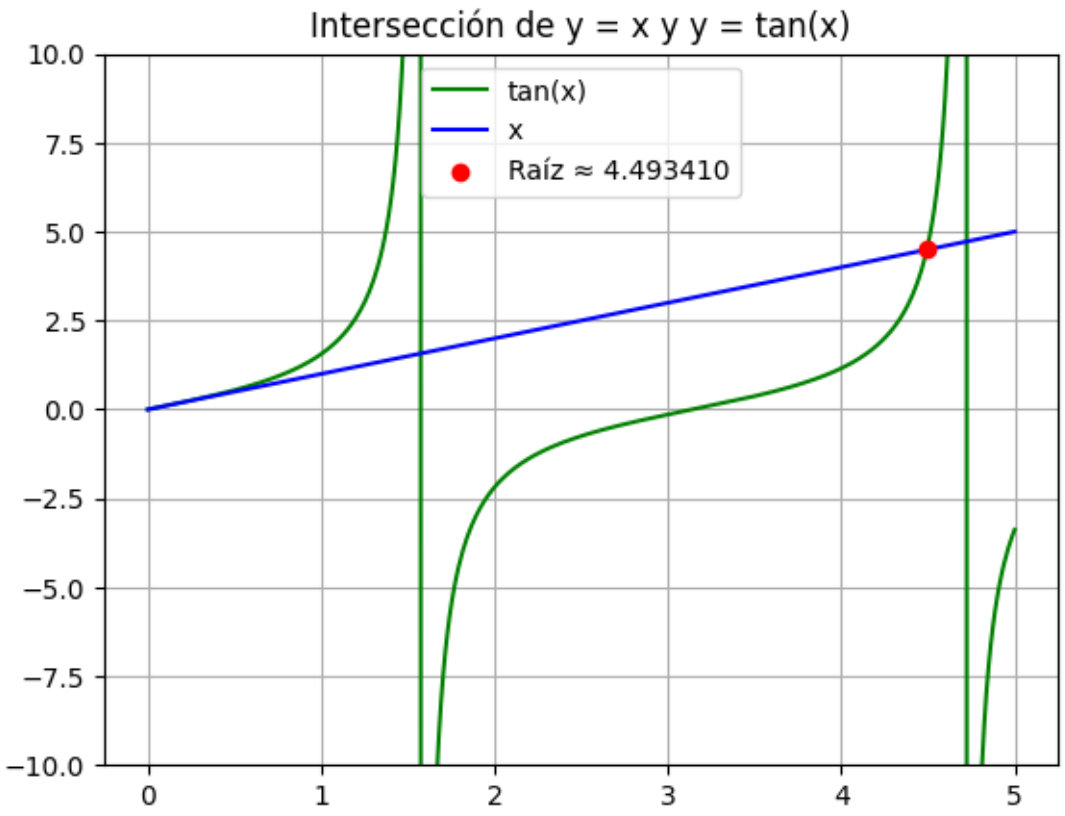


b. Use el método de bisección para encontrar una aproximación dentro de 10^{-5} para el primer valor positivo de x con $x = \tan x$.

[23]:
```
import numpy as np
import matplotlib.pyplot as plt
import os

def biseccion(f, a, b, tol=1e-6, max_iter=1000):
    if f(a)*f(b) > 0:
        print("El método no se puede aplicar en este intervalo.")
        return None

    iter_count = 0
    while (b - a)/2 > tol and iter_count < max_iter:
```

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

f = lambda x: x - np.tan(x)

a, b = 4, 4.7
raiz = biseccion(f, a, b, tol=1e-6)

if raiz is not None:
    print(f"Raíz  {raiz:.6f}")
else:
    print("No se encontró raíz en el intervalo.")

x_vals = np.linspace(0, 5, 400)
plt.plot(x_vals, np.tan(x_vals), label='tan(x)', color='green')
plt.plot(x_vals, x_vals, label='x', color='blue')
plt.ylim(-10, 10)
if raiz is not None:
    plt.scatter(raiz, raiz, color='red', zorder=5, label=f'Raíz  {raiz:.6f}')
plt.legend()
plt.grid(True)
plt.title("Intersección de y = x y y = tan(x)")

ruta_carpeta = r"C:
 ↪\Workspace-Metodos-Numericos-2k25A-\Tareas\Tarea4\ImagenesTarea4"
os.makedirs(ruta_carpeta, exist_ok=True)
ruta_imagen = os.path.join(ruta_carpeta, "pregunta3bGrafica_adaptada.png")
plt.savefig(ruta_imagen)

plt.show()
```

```
Raíz  4.493410
```

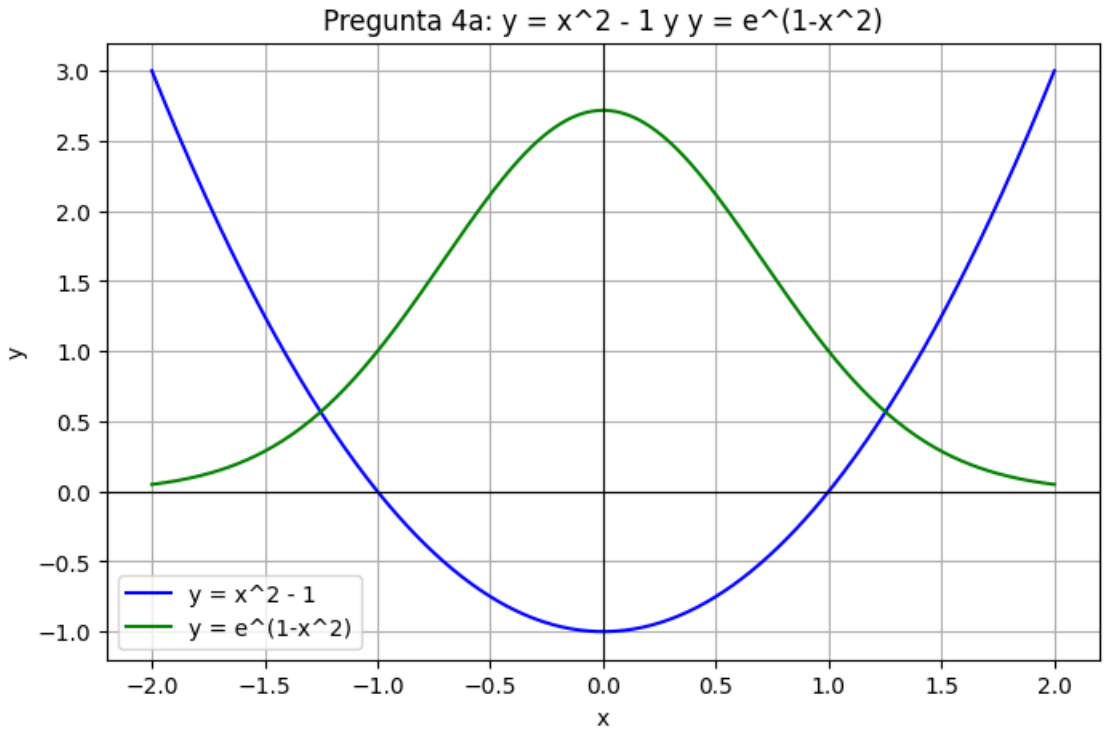


2 Pregunta 4

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

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

x = np.linspace(-2, 2, 400)

y1 = x**2 - 1
y2 = np.exp(1 - x**2)

plt.figure(figsize=(8,5))
plt.plot(x, y1, label='y = x^2 - 1', color='blue')
plt.plot(x, y2, label='y = e^(1-x^2)', color='green')
plt.axhline(0, color='black', linewidth=0.8)
plt.axvline(0, color='black', linewidth=0.8)
plt.title("Pregunta 4a: y = x^2 - 1 y y = e^(1-x^2)")
plt.xlabel("x")
```

```
plt.ylabel("y")
plt.legend()
plt.grid(True)

ruta_carpeta = r"C:
 ↪\Workspace-Metodos-Numericos-2k25A-\Tareas\Tarea4\ImagenesTarea4"
os.makedirs(ruta_carpeta, exist_ok=True)
ruta_imagen = os.path.join(ruta_carpeta, "pregunta4aGrafica.png")
plt.savefig(ruta_imagen)

plt.show()
```

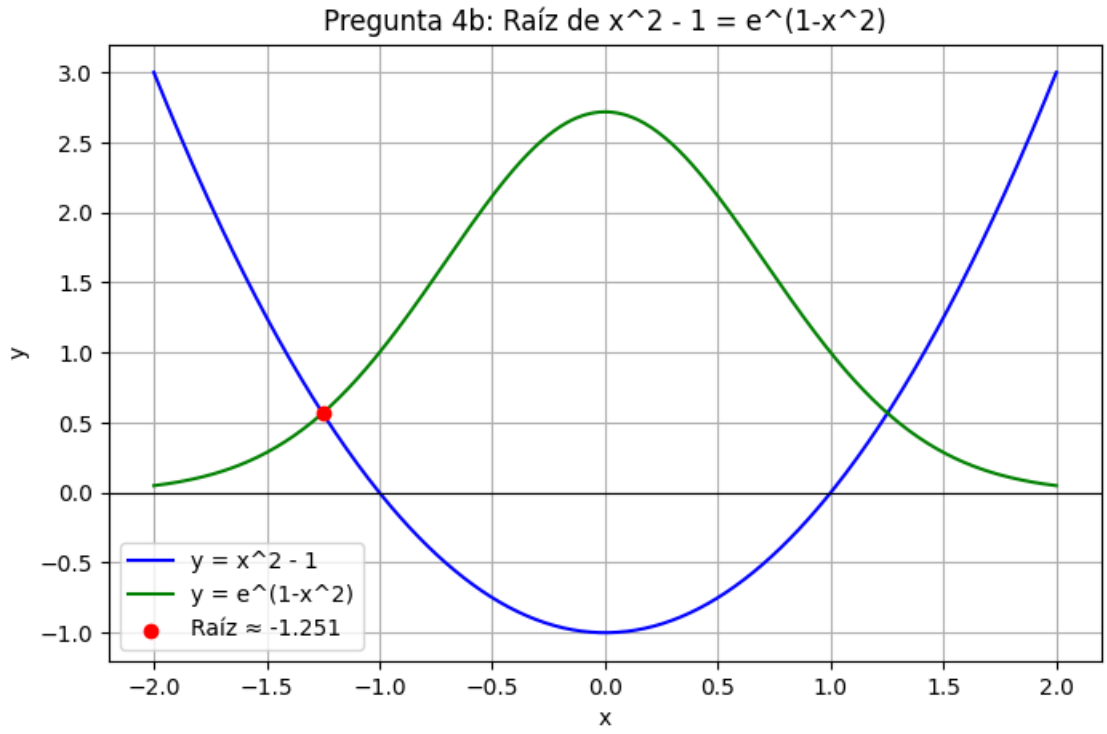


b. Use el método de bisección para encontrar una aproximación dentro de 10^{-3} para un valor en $[-2, 0]$ con $x^2 - 1 = e^{1-x^2}$.

[27]:
```
def biseccion(f, a, b, tol=1e-3, max_iter=1000):
    if f(a)*f(b) > 0:
        print("El método no se puede aplicar en este intervalo.")
        return None
    iter_count = 0
    while (b - a)/2 > tol and iter_count < max_iter:
        c = (a + b)/2
        if f(c) == 0:
```

```
            return c
        elif f(a)*f(c) < 0:
            b = c
        else:
            a = c
        iter_count += 1
    return (a + b)/2

f = lambda x: x**2 - 1 - np.exp(1 - x**2)

a, b = -2, 0
raiz = biseccion(f, a, b, tol=1e-3)

if raiz is not None:
    print(f"Raíz aproximada en [-2,0]: {raiz:.3f}")
else:
    print("No se encontró raíz en el intervalo.")

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

plt.figure(figsize=(8,5))
plt.plot(x_vals, y1, label='y = x^2 - 1', color='blue')
plt.plot(x_vals, y2, label='y = e^(1-x^2)', color='green')
plt.axhline(0, color='black', linewidth=0.8)
if raiz is not None:
    plt.scatter(raiz, raiz**2 - 1, color='red', zorder=5, label=f'Raíz  {raiz:.
 ↪3f}')
plt.title("Pregunta 4b: Raíz de x^2 - 1 = e^(1-x^2)")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid(True)

ruta_imagen = os.path.join(ruta_carpeta, "pregunta4bGrafica.png")
plt.savefig(ruta_imagen)
plt.show()
```

```
Raíz aproximada en [-2,0]: -1.251
```



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]

d. [-3, -0.5]

[1]:
```python
f = lambda x: (x + 3) * (x + 1)**2 * x * (x - 1)**3 * (x - 3)

def biseccion(a, b, tol=1e-6, max_iter=100):
    if f(a) * f(b) > 0:
        print(f"No hay cambio de signo en [{a}, {b}]")
        return None

    for i in range(max_iter):
        c = (a + b) / 2
        if abs(f(c)) < tol or abs(b - a) / 2 < tol:
            print(f"Iteraciones: {i+1}")
            return c
```

```
        if f(a) * f(c) < 0:
            b = c
        else:
            a = c
    print("No convergió")
    return (a + b) / 2

for intervalo in [(-1.5, 2.5), (-0.5, 2.4), (-0.5, 3), (-0.5, 3.5)]:
    raiz = biseccion(*intervalo)
    if raiz is not None:
        print(f"Raíz aproximada en {intervalo}: {raiz}")
```

```
No hay cambio de signo en [-1.5, 2.5]
No hay cambio de signo en [-0.5, 2.4]
Iteraciones: 22
Raíz aproximada en (-0.5, 3): 2.999999165534973
Iteraciones: 3
Raíz aproximada en (-0.5, 3.5): 3.0
```

2.1 EJERCICIOS APLICADOS

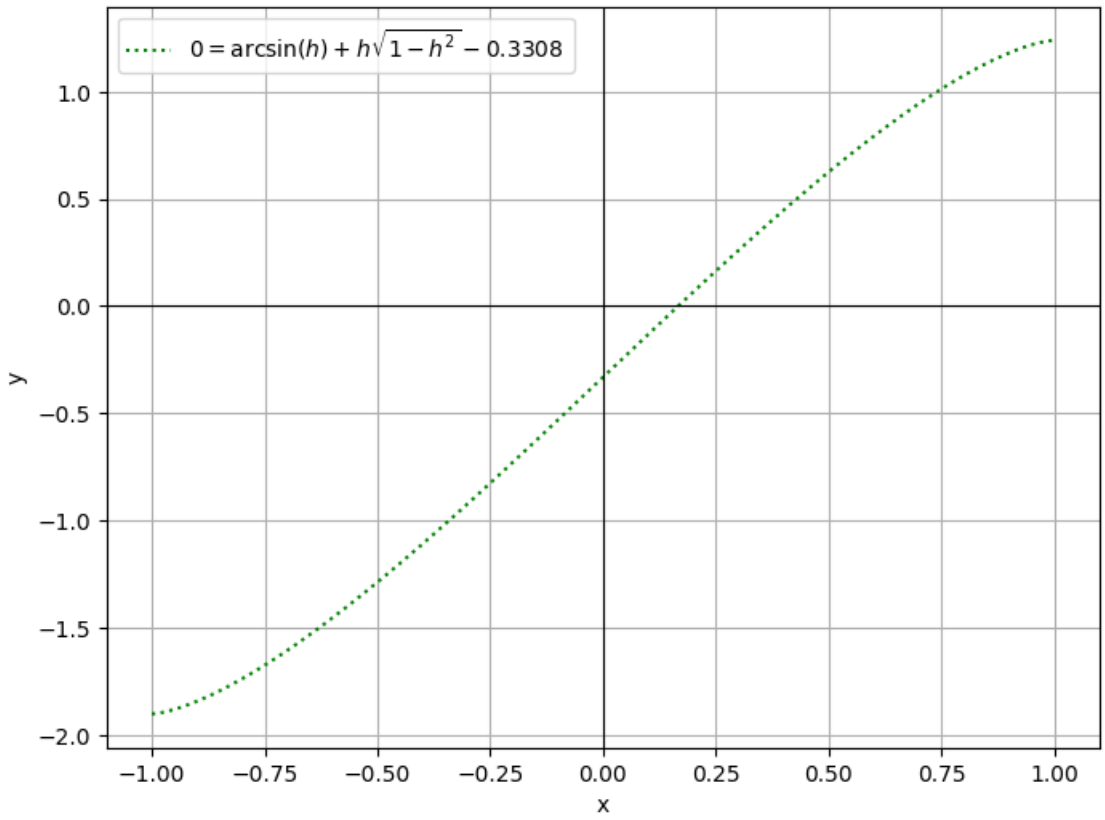
1. Un abrevadero de longitud tiene una sección transversal en forma de semicírculo con radio . (Consulte la figura adjunta.) Cuando se llena con agua hasta una distancia a partir de la parte superior, el volumen de agua es

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

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

[5]:
```
import numpy as np
import matplotlib.pyplot as plt
import os

ruta_guardado = r"C:
 ↪\Workspace-Metodos-Numericos-2k25A-\Tareas\Tarea4\ImagenesTarea4"
os.makedirs(ruta_guardado, exist_ok=True)

x = np.linspace(-1, 1, 800)
y = np.arcsin(x) + x * np.sqrt(1 - x**2) - 0.3308

plt.figure(figsize=(8, 6))
plt.plot(x, y, label=r'$0 = \arcsin(h) + h\sqrt{1 - h^2} - 0.3308$',␣
 ↪color='green', linestyle=':')
plt.axhline(0, color='black', linewidth=0.8)
plt.axvline(0, color='black', linewidth=0.8)
plt.xlabel('x')
```

```
plt.ylabel('y')
plt.legend()
plt.grid(True)

# Guardar imagen
nombre_imagen = "Grafica_Biseccion_Tarea4.png"
plt.savefig(os.path.join(ruta_guardado, nombre_imagen), dpi=300,␣
 ↪bbox_inches='tight')
plt.show()

f = lambda x: np.arcsin(x) + x * np.sqrt(1 - x**2) - 0.3308

def biseccion(a, b, tol=1e-2, max_iter=50):
    if f(a) * f(b) > 0:
        print("No hay cambio de signo en los extremos del intervalo.")
        return None

    for i in range(1, max_iter + 1):
        c = (a + b) / 2
        if abs(f(c)) < tol or abs(b - a) < tol:
            print(f"Iteraciones totales: {i}")
            return c
        if f(a) * f(c) < 0:
            b = c
        else:
            a = c
    print("No convergió")
    return c

raiz = biseccion(0, 0.5)
print(f"Raíz aproximada: h = {raiz:.6f}")
```




```
Iteraciones totales: 6
Raíz aproximada: h = 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 m cae desde una altura s_0 y que la altura del objeto después de t segundos es:

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

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

[3]:
```
import numpy as np

f = lambda x: 300 - 24.525*x + 61.3125*(1 - np.exp(-0.4*x))

def biseccion(a, b, tol=1e-2, max_iter=50):
    print(f"\nIntervalo inicial: [{a}, {b}]")

    if f(a) * f(b) > 0:
```

```
        print("No hay cambio de signo en los extremos del intervalo.")
        return None

    for i in range(1, max_iter + 1):
        c = (a + b) / 2
        fc = f(c)
        print(f"Iteración {i:2d} = {c:.6f}, f(c) = {fc:.6f}")

        if abs(fc) < tol:
            break
        if f(a) * fc < 0:
            b = c
        else:
            a = c

    return c

raiz = biseccion(14, 15)
print(f"\nRaíz aproximada: t = {raiz:.6f}")
```

```
Intervalo inicial: [14, 15]
Iteración  1 = 14.500000, f(c) = 5.514373
Iteración  2 = 14.750000, f(c) = -0.599212
Iteración  3 = 14.625000, f(c) = 2.457801
Iteración  4 = 14.687500, f(c) = 0.929348
Iteración  5 = 14.718750, f(c) = 0.165081
Iteración  6 = 14.734375, f(c) = -0.217062
Iteración  7 = 14.726562, f(c) = -0.025990
Iteración  8 = 14.722656, f(c) = 0.069546
Iteración  9 = 14.724609, f(c) = 0.021778
Iteración 10 = 14.725586, f(c) = -0.002106

Raíz aproximada: t = 14.725586
```

2.1.1 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.

2.2 Cálculo del número de iteraciones necesarias

Queremos determinar cuántas iteraciones (n) son necesarias para que el error del método de bisección sea menor que (10^{-4}).

La fórmula del error en el método de bisección es:

$$\frac{b-a}{2^n} \geq 10^{-4}$$

Dado que el intervalo inicial es ([1, 2]), tenemos:

$$b - a = 1$$

Sustituyendo en la fórmula:

$$\frac{1}{2^n} \geq 10^{-4}$$

Aplicamos logaritmos naturales para despejar (n):

$$\ln(2^n) \leq \ln(10^4)$$

Simplificamos:

$$n \ln(2) \leq 4 \ln(10)$$

Despejamos (n):

$$n \leq \frac{4 \ln(10)}{\ln(2)}$$

Calculando:

$$n \leq \frac{4(2.30258)}{0.69314} \approx 13.29$$

Por lo tanto:

$$n \approx 13 \text{ a } 14$$

[7]:
```
def f(x):
    return x**3 - x - 1

def biseccion(a, b, tol=1e-4, max_iter=50):
    if f(a) * f(b) >= 0:
        print("El método de bisección no se puede aplicar: f(a) y f(b) deben␣
 ↪tener signos opuestos.")
        return None, 0

    iteracion = 0
    print(f"\nIntervalo inicial: [{a}, {b}]")
    print("Iter |        c        |       f(c)")
```

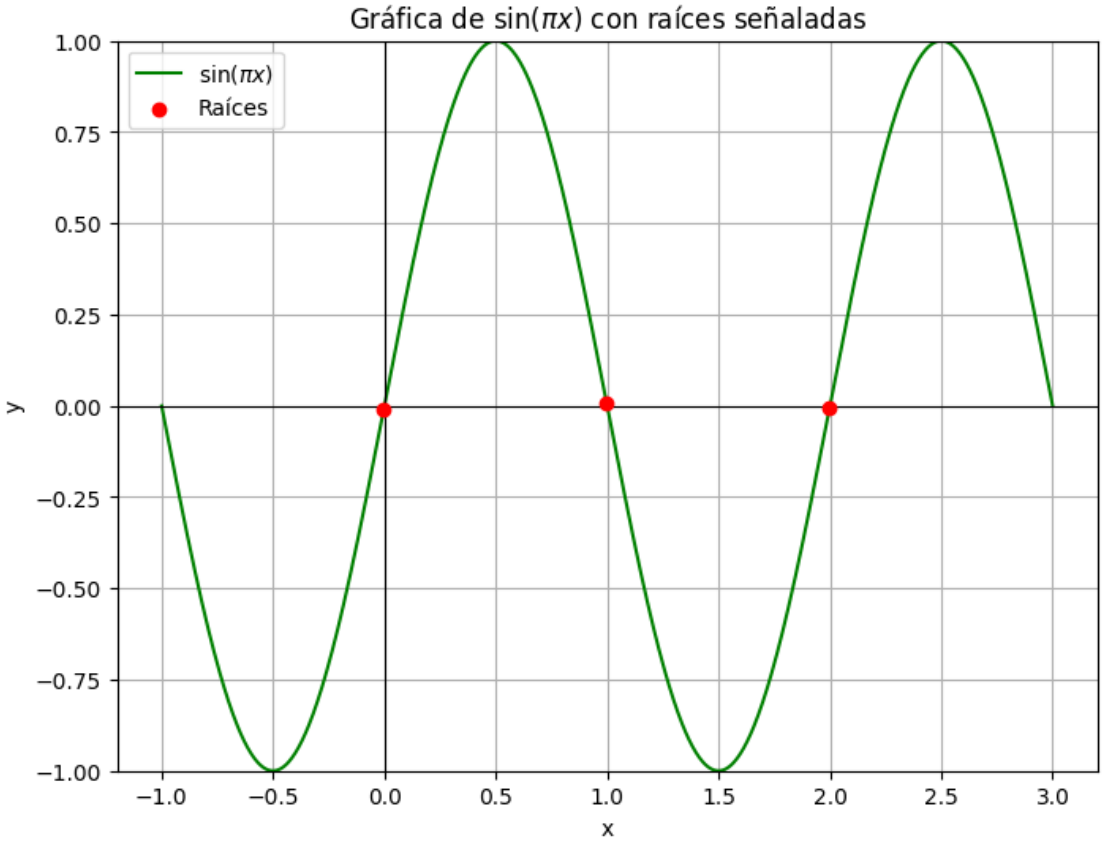
```
    print("-" * 35)

    while abs(b - a) >= tol and iteracion < max_iter:
        c = (a + b) / 2.0
        fc = f(c)
        print(f"{iteracion+1:4d} | {c:14.8f} | {fc:14.8f}")

        if fc == 0:
            break
        elif f(a) * fc < 0:
            b = c
        else:
            a = c
        iteracion += 1

    return (a + b) / 2.0, iteracion

raiz, iteraciones = biseccion(1, 2)
print(f"\nNúmero total de iteraciones: {iteraciones}")
print(f"Raíz aproximada: {raiz:.6f}")
```

```
Intervalo inicial: [1, 2]
Iter |        c        |       f(c)
-----------------------------------
   1 |     1.50000000 |     0.87500000
   2 |     1.25000000 |    -0.29687500
   3 |     1.37500000 |     0.22460938
   4 |     1.31250000 |    -0.05151367
   5 |     1.34375000 |     0.08261108
   6 |     1.32812500 |     0.01457596
   7 |     1.32031250 |    -0.01871061
   8 |     1.32421875 |    -0.00212795
   9 |     1.32617188 |     0.00620883
  10 |     1.32519531 |     0.00203665
  11 |     1.32470703 |    -0.00004659
  12 |     1.32495117 |     0.00099479
  13 |     1.32482910 |     0.00047404
  14 |     1.32476807 |     0.00021371

Número total de iteraciones: 14
Raíz aproximada: 1.324738
```

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 $a + b = 2$

[17]:
```python
import numpy as np
import matplotlib.pyplot as plt
import os

save_path = r"C:
 ↪\Workspace-Metodos-Numericos-2k25A-\Tareas\Tarea4\ImagenesTarea4"
os.makedirs(save_path, exist_ok=True)

x = np.linspace(-1, 3, 800)
y = np.sin(np.pi * x)

plt.figure(figsize=(8, 6))
plt.plot(x, y, color='green', label=r'$\sin(\pi x)$')

roots_indices = np.where(np.diff(np.sign(y)))[0]
roots_x = x[roots_indices]
roots_y = y[roots_indices]

plt.scatter(roots_x, roots_y, color='red', label='Raíces', zorder=5)

print("Raíces encontradas en el intervalo [-1, 3]:")
for i, rx in enumerate(roots_x, 1):
    print(f"  Raíz {i}: x  {rx:.6f}")

plt.xlabel('x')
plt.ylabel('y')
plt.title("Gráfica de $\\sin(\\pi x)$ con raíces señaladas")
plt.axhline(0, color='black', linewidth=0.8)
plt.axvline(0, color='black', linewidth=0.8)
plt.ylim(-1, 1)
plt.grid(True)
plt.legend()

image_path = os.path.join(save_path, "grafica_sin_pi_x.png")
plt.savefig(image_path, dpi=300, bbox_inches='tight')
plt.show()
```

```
Raíces encontradas en el intervalo [-1, 3]:
  Raíz 1: x  -0.003755
  Raíz 2: x  0.997497
  Raíz 3: x  1.998748
```


Gráfica de sin(πx) con raíces señaladas
sin(πx)
Raíces
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
y
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x