📊 Métodos Numéricos

🧑‍🏫 Profesor: Jonathan Zea

👤 Estudiante: Matthew Cedeño

📅 Fecha: 16 de mayo de 2025

📝 Tema: Bisección

✨ “La matemática es el lenguaje con el que Dios escribió el universo.”

Use el método de bisección para encontrar soluciones precisas

In [2]:
```
## Metodo de bisección para encontrar raices de una funcion
import numpy as np
def f(x):
    return x**3 - 7*x**2 + 14*x - 6
def biseccion(f, a, b, tol=1e-2, max_iter=100):
    for i in range(max_iter):
        c = (a+b)/2
        if abs(f(c)) < tol:
            return c
        if f(a)*f(c) < 0:
            b = c
        else:
            a = c
    return c
raiz = biseccion(f, 0, 1)
print(f"La raiz es: {raiz}")
```

```
La raiz es: 0.5859375
```

In [3]:
```
raiz = biseccion(f, 1, 3.2)
print(f"La raiz es: {raiz}")
```

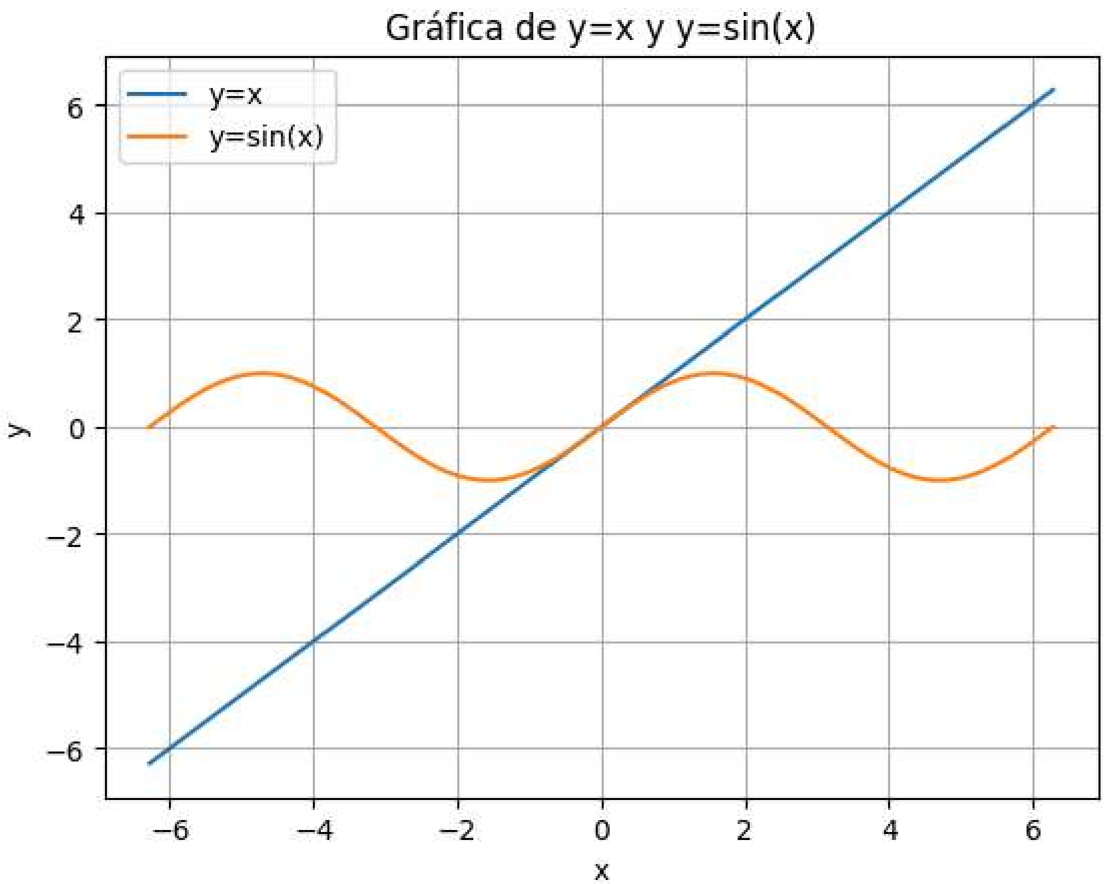
```
La raiz es: 2.9937500000000004
```

In [4]:
```
raiz = biseccion(f, 3.4, 4)
print(f"La raiz es: {raiz}")
```

```
La raiz es: 3.41875
```

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

In [7]:
```
import matplotlib.pyplot as plt
x = np.linspace(-2*np.pi, 2*np.pi, 500)
y1=x
y2=np.sin(x)
plt.plot(x, y1, label='y=x')
plt.plot(x, y2, label='y=sin(x)')
plt.title('Gráfica de y=x y y=sin(x)')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```

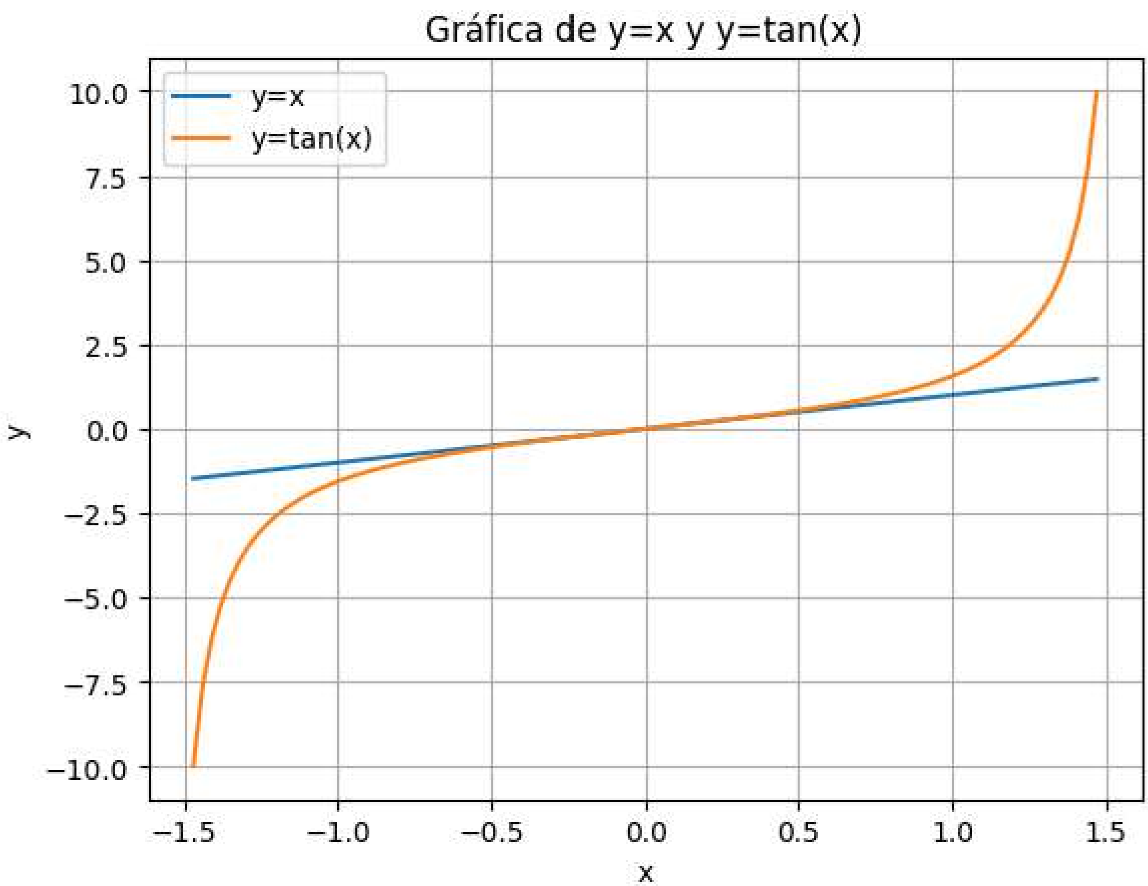


In [ ]:
```
def f(x):
    return x-2*np.sin(x)
def biseccion(f, a, b, tol=1e-6, max_iter=100):
    for i in range(max_iter):
        c = (a+b)/2
        if abs(f(c)) < tol:
            return c
        if f(a)*f(c) < 0:
            b = c
        else:
            a = c
    return c
raiz = biseccion(f, 1, 2)
print(f"La raiz es: {raiz}")
```

```
La raiz es: 1.8954944610595703
```

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

In [9]:
```
x=np.linspace(-np.pi/2 +0.1,np.pi/2 -0.1,100)
y1=x
y2=np.tan(x)
plt.plot(x, y1, label='y=x')
plt.plot(x, y2, label='y=tan(x)')
plt.title('Gráfica de y=x y y=tan(x)')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```



In [10]:
```
def f(x):
    return x-np.tan(x)
def biseccion(f, a, b, tol=1e-5, max_iter=100):
    for i in range(max_iter):
        c = (a+b)/2
        if abs(f(c)) < tol:
            return c
        if f(a)*f(c) < 0:
            b = c
        else:
            a = c
```

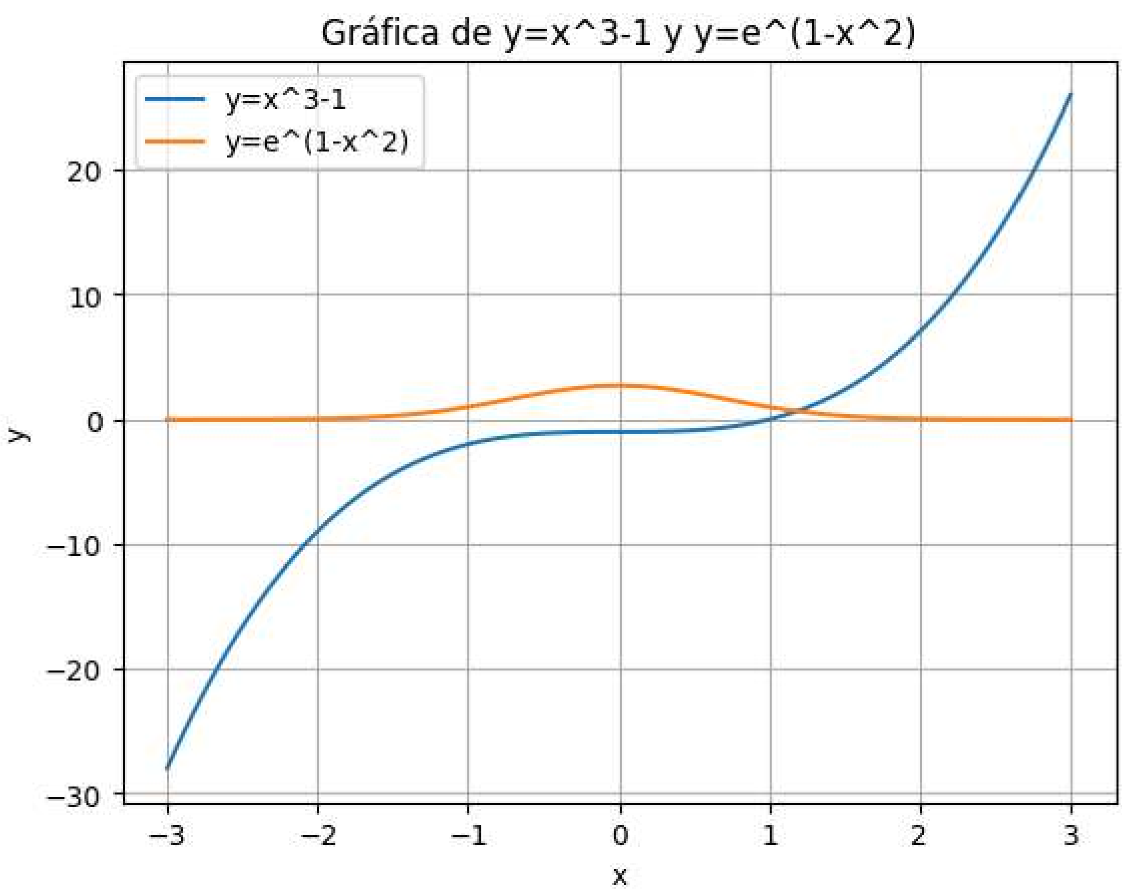
```
    return c
raiz = biseccion(f, 1, 2)
print(f"La raiz es: {raiz}")
```

La raiz es: 1.5707963267948966

Dibuje las gráficas para las funciones:

- (y = x^3 - 1)
- (y = e^{x/3})

In [12]:
```
x=np.linspace(-3,3,100)
y1=np.power(x,3)-1
y2=np.power(np.e,1-np.power(x,2))
plt.plot(x, y1, label='y=x^3-1')
plt.plot(x, y2, label='y=e^(1-x^2)')
plt.title('Gráfica de y=x^3-1 y y=e^(1-x^2)')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```



In [14]:
```
def f(x):
    return np.power(x,2)-1-np.power(np.e,1-np.power(x,2))
def biseccion(f, a, b, tol=1e-3, max_iter=100):
    for i in range(max_iter):
```

```
        c = (a+b)/2
        if abs(f(c)) < tol:
            return c
        if f(a)*f(c) < 0:
            b = c
        else:
            a = c
    return c
raiz = biseccion(f, -2, 0)
print(f"La raiz es: {raiz}")
```

```
La raiz es: -1.251953125
```

5.

In [15]:
```
def f(x):
    return (x+3)*np.power(x+1,2)*x*np.power(x-1,3)*(x-3)
def biseccion(f, a, b, tol=1e-3, max_iter=100):
    for i in range(max_iter):
        c = (a+b)/2
        if abs(f(c)) < tol:
            return c
        if f(a)*f(c) < 0:
            b = c
        else:
            a = c
    return c
raiz = biseccion(f, -1.5, 2.5)
print(f"La raiz es: {raiz}")
raiz = biseccion(f, -0.5, 2.4)
print(f"La raiz es: {raiz}")
raiz = biseccion(f, -0.5, 3)
print(f"La raiz es: {raiz}")
raiz = biseccion(f, -3, -0.5)
print(f"La raiz es: {raiz}")
```

```
La raiz es: 0.0
La raiz es: 3.0517578124984686e-05
La raiz es: 2.9999995827674866
La raiz es: -0.5
```

Ejercicios aplicados

- Abrevadero

In [18]:
```
import numpy as np

def calcular_volumen(L, r, h):
    """
    Calcula el volumen de agua en el abrevadero semicircular

    Parámetros:
    L: longitud del abrevadero (cm)
```

```
    r: radio del semicírculo (cm)
    h: profundidad del agua desde la parte superior (cm)

    Retorna:
    V: volumen de agua (cm³)
    """
    if h > r:
        raise ValueError("La profundidad h no puede ser mayor que el radio r")

    termino1 = 0.5 * np.pi * r**2
    termino2 = r**2 * np.arcsin(h/r)
    termino3 = h * np.sqrt(r**2 - h**2)

    V = L * (termino1 - termino2 - termino3)

    return V

def encontrar_profundidad(L, r, volumen_objetivo, precision=0.01):
    """
    Encuentra la profundidad del agua para un volumen dado

    Parámetros:
    L: longitud del abrevadero (cm)
    r: radio del semicírculo (cm)
    volumen_objetivo: volumen de agua conocido (cm³)
    precision: precisión requerida (cm)

    Retorna:
    h: profundidad del agua (cm)
    """
    h_min = 0
    h_max = r

    while (h_max - h_min) > precision:
        h_medio = (h_min + h_max) / 2
        volumen_calculado = calcular_volumen(L, r, h_medio)

        if volumen_calculado < volumen_objetivo:
            h_min = h_medio
        else:
            h_max = h_medio

    return round(h_min, 2)  # Redondear a 2 decimales (0.01 cm)

# Valores del problema
L = 10  # cm
r = 1   # cm
V = 12.4  # cm³

# Calcular la profundidad
profundidad = encontrar_profundidad(L, r, V)
print(f"La profundidad del agua en el abrevadero es {profundidad} cm")
```

```
La profundidad del agua en el abrevadero es 0.99 cm
```

Un objeto que cae verticalmente a través del aire está sujeto a una resistencia viscosa, así como a la fuerzade 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

In [1]:

```python
import numpy as np

# Parámetros dados
s0 = 300  # altura inicial en metros
m = 0.25  # masa en kg
k = 0.1   # coeficiente de resistencia en Ns/m
g = 9.81  # aceleración gravitacional en m/s²

# Función que describe la altura en función del tiempo
def altura(t):
    return s0 - (m*g/k)*t + (m**2*g/k**2)*(1-np.exp(-k*t/m))

# Método de bisección con mayor precisión
def biseccion(f, a, b, tol=0.00001):
    """
    Encuentra la raíz de f en el intervalo [a,b] con tolerancia tol
    """
    if f(a) * f(b) >= 0:
        print("El método de bisección requiere que f(a) y f(b) tengan signos opuest
        return None

    c = a
    iteraciones = 0

    while (b - a) >= tol:
        iteraciones += 1
        # Encontrar el punto medio
        c = (a + b) / 2

        # Verificar si el punto medio es raíz (con tolerancia numérica)
        if abs(f(c)) < 1e-10:
            break

        # Decidir qué mitad contiene la raíz
        if f(c) * f(a) < 0:
            b = c
        else:
            a = c

    print(f"Convergencia después de {iteraciones} iteraciones")
    return c

# Basado en los resultados anteriores, refinamos aún más el intervalo
a = 14.725  # tiempo inicial refinado
b = 14.73   # tiempo final refinado

print(f"Intervalo refinado: [{a}, {b}]")
print(f"Altura en t={a}: {altura(a):.10f} metros")
print(f"Altura en t={b}: {altura(b):.10f} metros")
```

```
# Aplicar bisección con mayor precisión
tiempo_impacto = biseccion(altura, a, b, tol=0.000001)
print(f"\nEl objeto golpea el suelo después de {tiempo_impacto:.6f} segundos")

# Verificación
altura_final = altura(tiempo_impacto)
print(f"Altura en ese tiempo: {altura_final:.10f} metros")

# Redondeamos a 0.01 segundos como pide el problema
tiempo_redondeado = round(tiempo_impacto, 2)
print(f"\nTiempo redondeado a 0.01 segundos: {tiempo_redondeado:.2f} segundos")
print(f"Altura a los {tiempo_redondeado:.2f} segundos: {altura(tiempo_redondeado):.

# Verificación más detallada
print("\nVerificación con valores más precisos:")
for t in np.arange(14.725, 14.728, 0.0005):
    h = altura(t)
    print(f"t = {t:.4f}, altura = {h:.10f} metros")
```

```
Intervalo refinado: [14.725, 14.73]
Altura en t=14.725: 0.0122247413 metros
Altura en t=14.73: -0.1100612973 metros
Convergencia después de 13 iteraciones

El objeto golpea el suelo después de 14.725500 segundos
Altura en ese tiempo: -0.0000008466 metros

Tiempo redondeado a 0.01 segundos: 14.73 segundos
Altura a los 14.73 segundos: -0.1100612973 metros

Verificación con valores más precisos:
t = 14.7250, altura = 0.0122247413 metros
t = 14.7255, altura = -0.0000038321 metros
t = 14.7260, altura = -0.0122324122 metros
t = 14.7265, altura = -0.0244609991 metros
t = 14.7270, altura = -0.0366895928 metros
t = 14.7275, altura = -0.0489181933 metros
t = 14.7280, altura = -0.0611468005 metros
```

Ejercicios teoricos

In [3]:
```
def f(x):
    return np.power(x,3)-x-1
def biseccion(f, a, b, tol=1e-4, max_iter=100):
    for i in range(max_iter):
        c = (a+b)/2
        if abs(f(c)) < tol:
            return c
        if f(a)*f(c) < 0:
            b = c
        else:
            a = c
    return c
```

```python
raiz = biseccion(f, 1, 2)
print(f"La raiz es: {raiz}")
```

```
La raiz es: 1.32470703125
```

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

# Definir la función f(x) = sin(pi * x)
def f(x):
    return np.sin(np.pi * x)

# Implementar el método de bisección
def biseccion(f, a, b, tol=1e-6, max_iter=100):
    if f(a) * f(b) >= 0:
        raise ValueError("No hay cambio de signo en el intervalo.")

    for i in range(max_iter):
        m = (a + b) / 2
        fm = f(m)

        if abs(fm) < tol or (b - a) / 2 < tol:
            return m

        if f(a) * fm < 0:
            b = m
        else:
            a = m

    return (a + b) / 2


casos = [
    ("Caso 1: a + b < 2", -0.5, 2.3),
    ("Caso 2: a + b = 2", -0.1, 2.1),
    ("Caso 3: a + b > 2", -0.2, 2.5)
]

for descripcion, a, b in casos:
    raiz = biseccion(f, a, b)
    print(f"{descripcion} → Raíz encontrada: {raiz:.6f} → Aproximadamente: {round(r
```

```
Caso 1: a + b < 2 → Raíz encontrada: 0.000000 → Aproximadamente: 0
Caso 2: a + b = 2 → Raíz encontrada: 1.000000 → Aproximadamente: 1
Caso 3: a + b > 2 → Raíz encontrada: 2.000000 → Aproximadamente: 2
```

Link del repositorio: https://github.com/MatthewC-20/Deber4