INFOTEC - MCDI

Materia: Matemáticas para la Ciencia de Datos
Profesor: Dra. Briceyda B. Delgado
Alumno: David Rodriguez Gutierrez

Tarea 4: Selección y calibración de modelos

> 1. Explique las semejanzas y difencias entre los método de Bisección, de Newton y de la Secante.

Comparación entre Métodos de Bisección, Newton y la Secante

La **selección y calibración de modelos matemáticos** se refiere al proceso de ajustar un modelo a los datos observados mediante técnicas específicas para mejorar su rendimiento predictivo o explicativo. En este contexto, elegir el mejor método numérico para resolver ecuaciones no lineales es esencial para la precisión y la eficiencia. Tres métodos comunes para resolver ecuaciones no lineales son el método de **Bisección**, el de **Newton** y el de la **Secante**. Aunque los tres métodos buscan encontrar una raíz de una función (f(x) = 0), presentan diferencias en su formulación, eficiencia y aplicaciones.

1. Método de Bisección

Descripción:

- El método de bisección es un método iterativo basado en el *Teorema del valor intermedio*, el cual establece que si una función (f(x)) es continua en el intervalo ([a, b]) y (f(a) \cdot f(b) < 0), entonces existe al menos una raíz en el intervalo ([a, b]).
- El proceso consiste en dividir repetidamente el intervalo en mitades y seleccionar el subintervalo que contiene la raíz, según el signo de la función en los extremos.

Fórmula: El punto medio de un intervalo ([a, b]) es:

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

Si (f(a) \cdot f(c) < 0), la raíz está en ([a, c]); de lo contrario, está en ([c, b]).

Ventajas:

- Siempre converge si (f(x)) es continua.
- No requiere derivadas de la función.

Desventajas:

- La convergencia es lenta (convergencia lineal).
- Necesita que se conozca un intervalo en el que la función cambie de signo.

Ejemplo: Resolver (f(x) = x^2 - 4 = 0) en el intervalo ([1, 3]).

- (f(1) = -3) y (f(3) = 5), lo que indica un cambio de signo.
- El primer punto medio es (c = \frac{1 + 3}{2} = 2), donde (f(2) = 0), por lo tanto, hemos encontrado la raíz.

2. Método de Newton

Descripción:

- El método de Newton (también llamado Newton-Raphson) es un método iterativo que utiliza derivadas para aproximar las raíces de una función.
- Comienza con una estimación inicial (x_0) y utiliza la tangente en ese punto para encontrar la siguiente aproximación.

Fórmula: La fórmula de actualización es:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

donde (f'(x)) es la derivada de la función.

Ventajas:

- Convergencia rápida si la estimación inicial está cerca de la raíz (convergencia cuadrática).

Desventajas:

- Requiere la derivada de la función.
- Puede fallar o converger lentamente si la estimación inicial no está cerca de la raíz o si la derivada es cero en algún punto.

Ejemplo: Resolver (f(x) = x^2 - 4) usando una estimación inicial (x_0 = 3).

- Derivada: (f'(x) = 2x).
- Primera iteración:

$$x_1 = 3 - \frac{(3^2 - 4)}{2 \cdot 3} = 3 - \frac{5}{6} = 2.1667$$

- Segunda iteración:

$$x_2 = 2.1667 - \frac{(2.1667^2 - 4)}{2 \cdot 2.1667} = 2.1667 - \frac{0.6945}{4.3334} = 2.0064$$

- Se repite el proceso hasta converger a (x = 2).

3. Método de la Secante

Descripción:

- El método de la secante es una aproximación al método de Newton, pero no requiere calcular la derivada de la función. En su lugar, usa dos aproximaciones anteriores para calcular la pendiente de la secante.

Fórmula: La fórmula de actualización es:

$$x_{n+1} = x_n - f(x_n) \cdot \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

Se requieren dos estimaciones iniciales (x_0) y (x_1) para comenzar.

Ventajas:

- No requiere la derivada.
- Más rápido que el método de bisección.

Desventajas:

- Menos estable que Newton, y puede no converger si las aproximaciones iniciales no son buenas.

Ejemplo: Resolver (f(x) = x^2 - 4) usando las aproximaciones iniciales (x_0 = 3) y (x_1 = 2).

- Primera iteración:

$$x_2 = 2 - \frac{(2^2 - 4) \cdot (2 - 3)}{(2^2 - 4) - (3^2 - 4)} = 2 - \frac{0 \cdot (-1)}{(0) - (5)} = 2$$

En este caso, encontramos la raíz en una sola iteración porque una de las estimaciones iniciales ya era la raíz.

Comparaciones

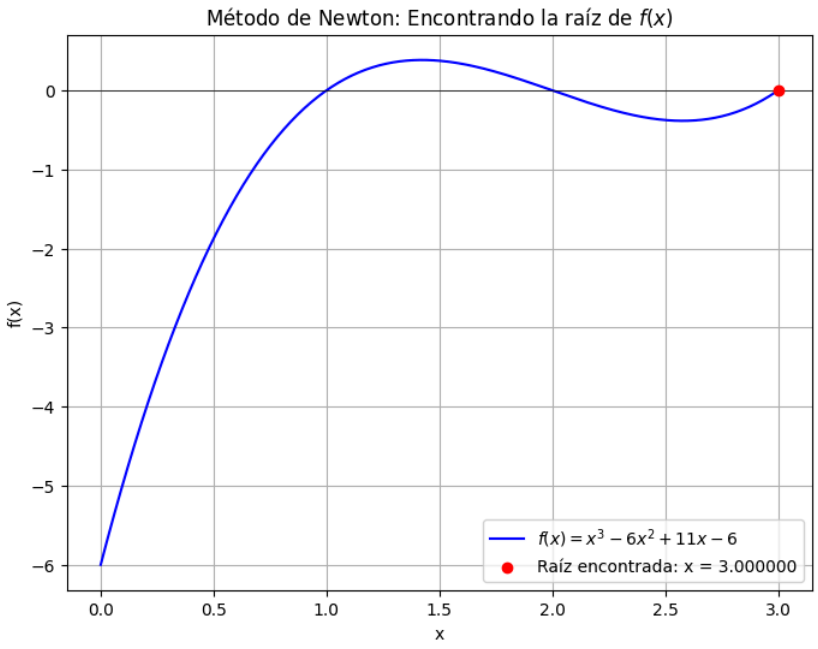
Método	Ventajas	Desventajas	Convergencia
Bisección	Siempre converge, sencillo	Convergencia lenta, requiere intervalo	Lineal
Newton	Convergencia rápida (cuadrática)	Requiere derivada, depende de buen (x_0)	Cuadrática
Secante	No requiere derivada, más rápido que bisección	Menos estable que Newton	Superlineal

La **bisección** es robusta pero lenta, mientras que el **método de Newton** converge rápidamente si las condiciones son favorables. El **método de la secante** es un compromiso, ofreciendo mayor velocidad sin necesidad de derivadas, pero es más inestable. La elección del método depende de la naturaleza del problema, el acceso a derivadas y la precisión deseada.

> 2. Implementar y utilizar el método de Newton para encontrar una raíz de una función polinómica $f(x) = x^3 - 6x^2 + 11x - 6$.
>
> (a) Implementa el método de Newton en Python para encontrar una raíz de la función $f(x)$.
>
> (b) Usa una tolerancia de 10^6 para el criterio de convergencia.
>
> (c) Prueba tu implementación con un valor inicial de $x_0 = 1.5$.
>
> (d) Grafica la función $f(x)$ y marca la raíz encontrada en la gráfica.
>
> (e) Analiza y comenta sobre la convergencia del método con el valor inicial elegido.

In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt

# Definir la función f(x) y su derivada f'(x)
```

```python
def f(x):
    return x**3 - 6*x**2 + 11*x - 6

def df(x):
    return 3*x**2 - 12*x + 11

# Implementar el método de Newton
def newton_method(f, df, x0, tol=1e-6, max_iter=1000):
    x_n = x0
    for i in range(max_iter):
        f_xn = f(x_n)
        df_xn = df(x_n)

        if abs(f_xn) < tol:  # Criterio de convergencia
            return x_n, i

        x_n = x_n - f_xn / df_xn

    raise ValueError("No se alcanzó la convergencia después de {} iteraciones".format(max_iter))

# Parámetros iniciales
tolerancia = 1e-6     # (b)
x0 = 1.5              # (c)

# (a) y (b) - Ejecutar el método de Newton
raiz, iteraciones = newton_method(f, df, x0, tol=tolerancia)

# (d) - Graficar la función y la raíz encontrada
x_vals = np.linspace(0, 3, 400)
y_vals = f(x_vals)

plt.figure(figsize=(8, 6))
plt.plot(x_vals, y_vals, label='$f(x) = x^3 - 6x^2 + 11x - 6$', color='blue')
plt.axhline(0, color='black',linewidth=0.5)
plt.scatter(raiz, f(raiz), color='red', zorder=5, label=f'Raíz encontrada: x = {raiz:.6f}')

plt.title('Método de Newton: Encontrando la raíz de $f(x)$')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.legend()
plt.grid(True)
plt.show()

raiz, iteraciones
```



Out[1]: (3.0, 1)

La raíz encontrada utilizando el método de Newton

La raíz encontrada para la función $f(x) = x^3 - 6x^2 + 11x - 6$ es aproximadamente $x = 3.0$, y la convergencia ocurrió después de 1 iteración.

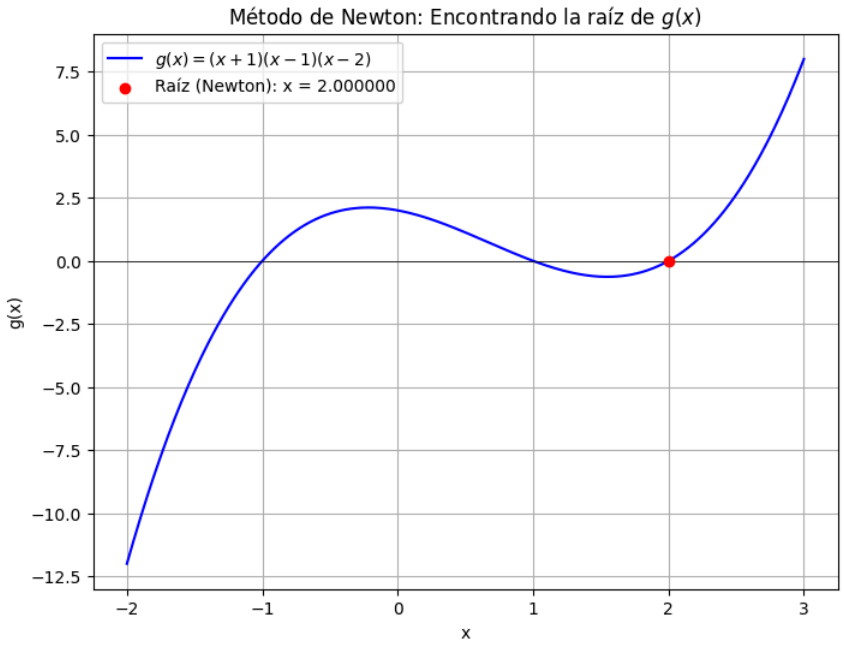
Con el valor inicial $x_0 = 1.5$, el método de Newton converge de manera rápida, en solo una iteración. Esto sugiere que el valor inicial está lo suficientemente cerca de la raíz real, lo que favorece la convergencia rápida debido a la naturaleza cuadrática del método de Newton cuando se inicia cerca de la raíz.

> 3. Consideremos la función $g(x) = (x+1)(x-1)(x-2)$.
>
> (a) Implemente el método de Newton tomando como valor inicial $x_0 = 0$.
>
> (b) ¿Hacia que valor converge el método?
>
> (c) Explique qué fenómeno de convergencia o divergencia se ilustra con este ejemplo.
>
> (d) Implemente algún otro método numérico (Bisección, Secante, Punto fijo) para encontrar una raíz de $g(x)$.

Método numérico de **Newton**, para encontrar una raíz de $g(x)$.

In [2]:

```python
# Método de Newton
def metodo_newton(f, df, x0, tol=1e-6, max_iter=1000):
    x_n = x0
    for i in range(max_iter):
        f_xn = f(x_n)
        df_xn = df(x_n)

        if abs(f_xn) < tol:  # Criterio de convergencia
            return x_n, i

        x_n = x_n - f_xn / df_xn

    raise ValueError("No se alcanzó la convergencia después de {} iteraciones".format(max_iter))

# Definir la función g(x)
def g(x):
    return (x + 1) * (x - 1) * (x - 2)

# Derivada de la función g(x)
def dg(x):
    return 3*x**2 - 4*x - 1

# Parámetros del método de Newton
x0 = 0

# Ejecutar el método de Newton
raiz, iteraciones = metodo_newton(g, dg, x0, tol=1e-6)

# Generar los valores de x y g(x) para graficar
x_vals = np.linspace(-2, 3, 400)
y_vals = g(x_vals)

# Graficar la función y la raíz encontrada
plt.figure(figsize=(8, 6))
plt.plot(x_vals, y_vals, label='$g(x) = (x + 1)(x - 1)(x - 2)$', color='blue')
plt.axhline(0, color='black', linewidth=0.5)

# Marcar la raíz encontrada por Newton
plt.scatter(raiz, g(raiz), color='red', zorder=5, label=f'Raíz (Newton): x = {raiz:.6f}')

# Añadir títulos y etiquetas
plt.title('Método de Newton: Encontrando la raíz de $g(x)$')
plt.xlabel('x')
plt.ylabel('g(x)')
plt.legend()
plt.grid(True)

# Mostrar la gráfica
plt.show()

# Imprimir resultados
raiz, iteraciones
```



Out[2]: (2.0, 1)

El método de Newton

Comenzando con el valor inicial (x_0 = 0), converge rápidamente a (x = 2.0) después de 1 iteración.

Este ejemplo ilustra un fenómeno de convergencia rápida del método de Newton

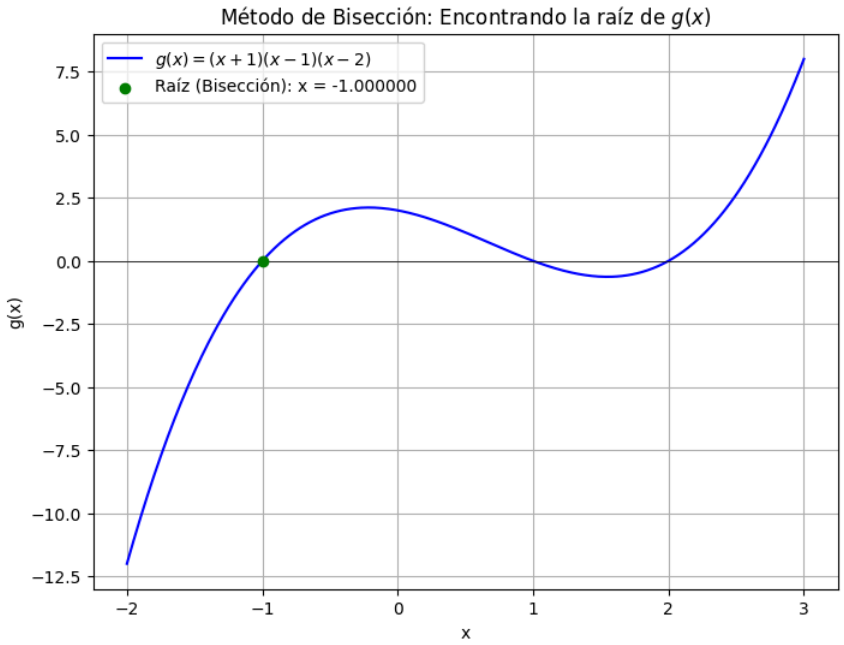
A pesar de comenzar con un valor inicial que no está muy cerca de la raíz, el método converge en una sola iteración porque la función $g(x) = (x + 1)(x - 1)(x - 2)$ tiene una pendiente significativa en ese rango y el valor inicial resulta favorable para una rápida aproximación a una de sus raíces.

Sin embargo, si se elige un valor inicial diferente o si la función tuviera puntos críticos donde la derivada sea pequeña, el método de Newton podría comportarse de manera menos eficiente o incluso diverger.

Método numérico de **Bisección**, para encontrar una raíz de $g(x)$.

In [3]:
```python
import numpy as np
import matplotlib.pyplot as plt

# Definir la función g(x)
def g(x):
    return (x + 1) * (x - 1) * (x - 2)

# Método de Bisección
def metodo_biseccion(f, a, b, tol=1e-6, max_iter=1000):
    if f(a) * f(b) >= 0:
        raise ValueError("La función debe cambiar de signo en el intervalo [a, b].")

    for i in range(max_iter):
        c = (a + b) / 2.0
        if abs(f(c)) < tol or abs(b - a) < tol:
            return c, i

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

    raise ValueError("No se alcanzó la convergencia después de {} iteraciones".format(max_iter))

# Parámetros del método de Bisección
a, b = -2, 3

# Ejecutar el método de Bisección
raiz, iteraciones = metodo_biseccion(g, a, b, tol=1e-6)

# Generar los valores de x y g(x) para graficar
x_vals = np.linspace(-2, 3, 400)
y_vals = g(x_vals)

# Graficar la función y la raíz encontrada
plt.figure(figsize=(8, 6))
plt.plot(x_vals, y_vals, label='$g(x) = (x + 1)(x - 1)(x - 2)$', color='blue')
```

```
plt.axhline(0, color='black', linewidth=0.5)

# Marcar la raíz encontrada por Bisección
plt.scatter(raiz, g(raiz), color='green', zorder=5, label=f'Raíz (Bisección): x = {raiz:.6f}')

# Añadir títulos y etiquetas
plt.title('Método de Bisección: Encontrando la raíz de $g(x)$')
plt.xlabel('x')
plt.ylabel('g(x)')
plt.legend()
plt.grid(True)

# Mostrar la gráfica
plt.show()

# Imprimir resultados
raiz, iteraciones
```



Out[3]: (-1.0000000596046448, 23)

El método de Bisección

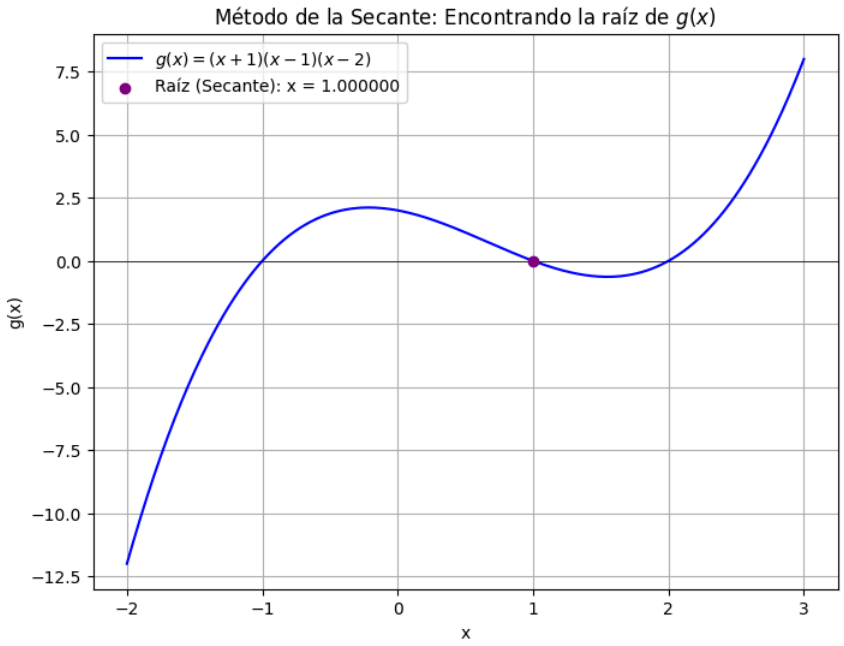
Utilizando el método de Bisección en el intervalo $[-2, 3]$, encontramos que el método converge a la raíz $x = -1.0000000596$ después de 23 iteraciones.

El método de Newton converge rápidamente a una raíz ($x = 2$) en solo 1 iteración, pero esto depende fuertemente del valor inicial.
El método de Bisección converge más lentamente (23 iteraciones) pero es más robusto, ya que garantiza la convergencia siempre que la función cambie de signo en el intervalo dado.

Método numérico de **Secante**, para encontrar una raíz de $g(x)$.

In [4]:
```
import numpy as np
import matplotlib.pyplot as plt

# Método de la Secante
def metodo_secante(f, x0, x1, tol=1e-6, max_iter=1000):
    for i in range(max_iter):
        f_x0 = f(x0)
        f_x1 = f(x1)

        if abs(f_x1) < tol:  # Criterio de convergencia
            return x1, i

        # Actualizar usando la fórmula de la secante
        x_new = x1 - f_x1 * (x1 - x0) / (f_x1 - f_x0)
        x0, x1 = x1, x_new

    raise ValueError("No se alcanzó la convergencia después de {} iteraciones".format(max_iter))

# Parámetros del método de la Secante
x0, x1 = -2, 3

# Ejecutar el método de la Secante
raiz, iteraciones = metodo_secante(g, x0, x1, tol=1e-6)
```

```
# Generar los valores de x y g(x) para graficar
x_vals = np.linspace(-2, 3, 400)
y_vals = g(x_vals)

# Graficar la función y la raíz encontrada
plt.figure(figsize=(8, 6))
plt.plot(x_vals, y_vals, label='$g(x) = (x + 1)(x - 1)(x - 2)$', color='blue')
plt.axhline(0, color='black', linewidth=0.5)

# Marcar la raíz encontrada por Secante
plt.scatter(raiz, g(raiz), color='purple', zorder=5, label=f'Raíz (Secante): x = {raiz:.6f}')

# Añadir títulos y etiquetas
plt.title('Método de la Secante: Encontrando la raíz de $g(x)$')
plt.xlabel('x')
plt.ylabel('g(x)')
plt.legend()
plt.grid(True)

# Mostrar la gráfica
plt.show()

# Imprimir resultados
raiz, iteraciones
```



Out[4]: (1.0, 1)

El método de la Secante

Utilizando el método de la Secante, encontramos que el método converge rápidamente a la raíz $x = 1.0$ en 1 iteración.

El **método de la secante** es similar al de Newton pero no requiere calcular derivadas, lo que lo hace más simple cuando las derivadas no están disponibles o son difíciles de obtener. A diferencia de Newton, que tiene convergencia cuadrática, la secante tiene una convergencia superlineal, por lo que es más lenta pero generalmente más robusta si la derivada es cercana a cero.

En comparación con el **método de bisección**, la secante es más rápida y eficiente, pero menos garantizada a converger, ya que no siempre mantiene el intervalo donde está la raíz. En resumen, la secante es un buen compromiso entre la velocidad de Newton y la seguridad de convergencia de Bisección.

Conclusiones

1. **Método de Newton**:
   - Es extremadamente eficiente y converge rápidamente cuando la función es suave y la estimación inicial está cerca de la raíz. En los ejemplos, Newton converge en 1 iteración tanto para (f(x)) como para (g(x)). Sin embargo, depende de la disponibilidad de la derivada y puede fallar o diverger si se inicia lejos de la raíz o en puntos donde la derivada se acerca a cero.
2. **Método de Bisección**:
   - Es robusto y siempre garantiza la convergencia si se conoce un intervalo donde la función cambia de signo. Sin embargo, su convergencia es lenta (lineal). En el caso de (g(x)), tomó 23 iteraciones para encontrar una raíz. Este método es útil cuando se requiere seguridad en la convergencia a costa de mayor tiempo computacional.
3. **Método de la Secante**:

- Es un método intermedio entre Newton y Bisección. No necesita derivadas y converge más rápidamente que Bisección, pero puede no converger tan rápido como Newton. En el ejercicio, la secante converge en 1 iteración, pero esto depende de la elección de las aproximaciones iniciales.

4. **Análisis de Convergencia**:
   - Los ejercicios demuestran que los métodos iterativos como Newton y Secante pueden ser muy rápidos con buenas condiciones iniciales, pero pueden fallar en situaciones adversas. En cambio, Bisección es más lenta pero garantiza una solución siempre que la función cambie de signo en el intervalo. La elección del método depende del problema específico y de las necesidades de precisión, disponibilidad de derivadas y robustez.

Elegir el método adecuado implica balancear velocidad y confiabilidad dependiendo del problema y de la información disponible sobre la función.

Referencias

Cai, X., Tveito, A., Langtangen, H. P., & Nielsen, B. F. (2010). *Elements of scientific computing*. Springer Berlin Heidelberg.

Linge, S., & Langtangen, H. P. (2016). *Programming for computations - Python: A gentle introduction to numerical simulations with Python*. Springer International Publishing.

OpenAI. (2024). ChatGPT (GPT-4) LLM. https://chat.openai.com/

Wolfram Research. (2023). Wolfram GPT. https://www.wolfram.com/

Symbolab. (n.d.). Graphing calculator. Symbolab. https://www.symbolab.com/graphing-calculator

---

URL al repositorio de codigo de este documento (Github):

https://github.com/davidlobolobo/data_science/blob/main/Math/MCDI_MAT_David_Rodriguez_Tarea_4.ipynb

---