Participantes

Andrea Lima Blanca - 174723

Dicka Jamesina Lezama Alvarado - 176880

Alejandra Hernández Vázquez – 174685

Georgina Zerón Cabrera - 174592

Materia

LFA3082-1: Análisis Numérico

Periodo

Otoño 2023

Proyecto 1

Resolución de ecuaciones en Python

Profesora

Anabel Hernández Ramírez

Métodos para el programa

Método de Bisección

El método de bisección se aplica en funciones algebraicas y trascendentes, de las que se busca obtener solo sus raíces reales. Consiste en cortar un intervalo (a, b) por la mitad, considerándose a ese punto como una aproximación a la raíz de la función, por consiguiente, se determina si la raíz encontrada se ubica a la derecha o a la izquierda según su cercanía al límite a o b, pero siempre manteniéndose dentro del intervalo, dicho proceso denominado teorema del valor intermedio se repite las veces necesarias hasta que la aproximación sea menor a la tolerancia que se establezca (Burden et al., 2017).

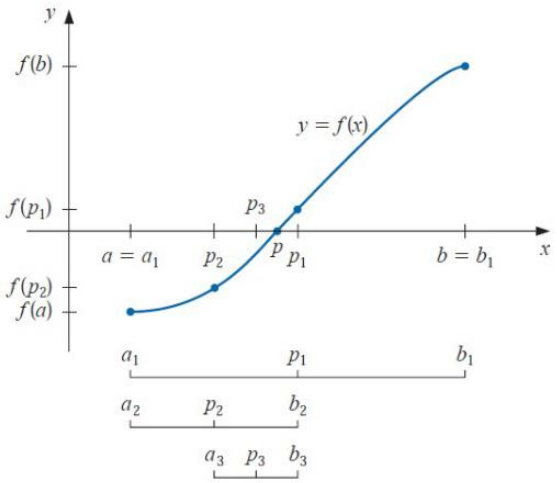


Figura 1 Gráfica del Método de Bisección

Método de Newton

El método de Newton busca encontrar las raíces de las funciones (de cualquier complejidad), partiendo del teorema de Taylor donde se establece un punto aproximado a la raíz denominado x0, del cual su pendiente coincide con la derivada de dicho punto; de tal modo

que el nuevo punto se interseca generando otro punto que lleva a repetir el procedimiento las veces necesarias hasta que se encuentra una aproximación a las raíces de la función original.

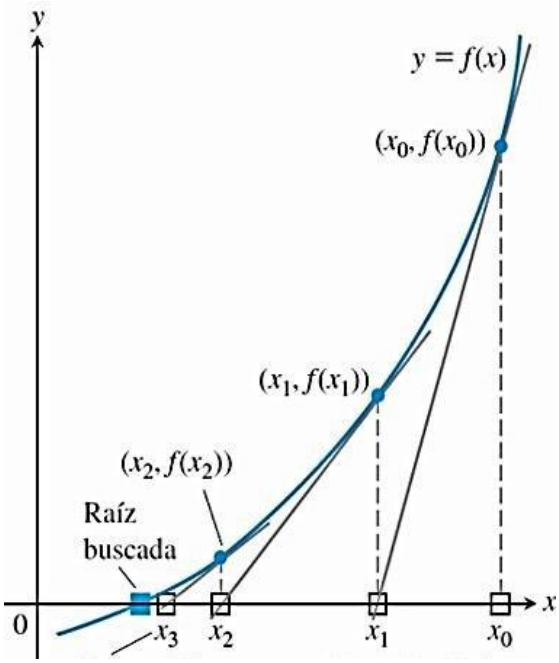


Figura 2 Gráfica del Método de Newton

Método de Newton Modificado

El método de Newton Modificado busca encontrar las raíces de una función, como dice su nombre funciona muy parecido al de Newton con la modificación de que permite resolver funciones no lineales con raíces críticas, tienen máximos, mínimos o puntos de inflexión donde corta el eje x, por lo que la función y la derivada se hacen cero (Burden et al., 2017). Cuando la raíz es un punto mínimo o máximo se repite la raíz en números pares, mientras que en los puntos de inflexión la raíz se repite en números impares, recibiéndose en ambos casos el nombre de raíces múltiples.

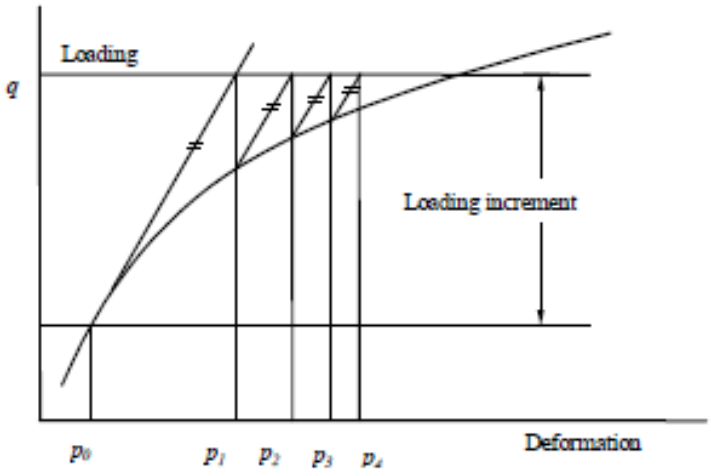


Figura 3 Gráfica del Método de Newton Modificado

Método de la Secante

El método de la secante funciona en base de la búsqueda de los ceros de una función de manera iterativa, donde se necesitan dos aproximaciones iniciales de la raíz y estima la tangente para encontrar una pendiente inicial, así creándose una serie de raíces de las líneas secantes para aproximarse mejor a la raíz de la función.

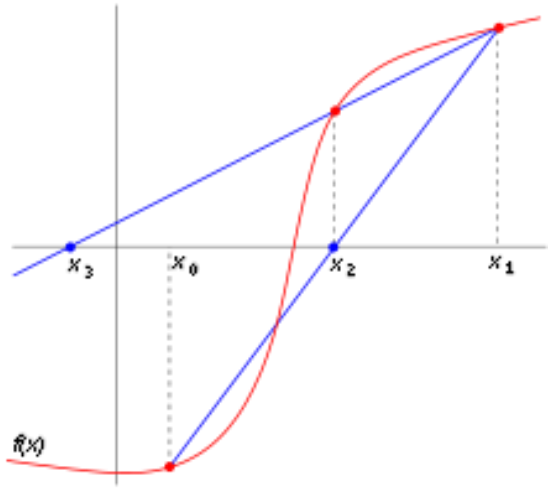


Figura 4 Gráfica del Método de la Secante

Descripción del código

El archivo del proyecto contiene 5 diferentes archivos: uno para cada uno de los métodos mencionados anteriormente (bisección, newton, newton modificado, y secante) y un archivo que contiene nuestro menú. Cada uno de ellos utiliza las librerías S*ymPy* y/o N*umPy* para poder manejar de manera correcta operaciones matemáticas.

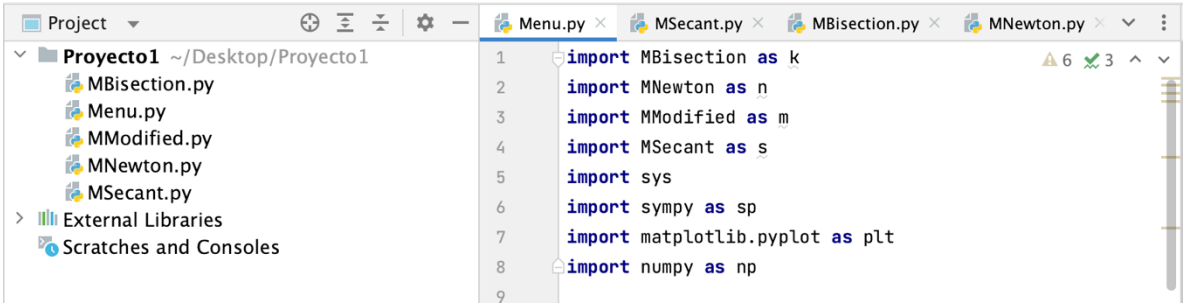


Figura 5 Importación de funciones y archivos

En el archivo del menú, que también juega el papel de nuestro main, primero se encuentra el fragmento de código donde se le solicita al usuario ingresar una función en términos de x., y posteriormente se despliegan las opciones de los métodos a través de los cuales se puede obtener la raíz de la función dada, además de la opción de salida. Una vez elegido el método, nuestro programa procederá a solicitarle al usuario los parámetros necesarios para la resolución del problema, por ejemplo, el número máximo de iteraciones, la tolerancia, etc.

```python
def get_root(choice, f, df, ddf):
    """Execute the selected root-finding method."""
    if choice == 1:
        a = float(input("Enter the starting interval a: "))
        b = float(input("Enter the ending interval b: "))
        tol = float(input("Enter the tolerance: "))
        print("\nUsing Bisection method:")
        return k.my_bisection(f, a, b, tol)
    elif choice == 2:
        x0 = float(input("Enter the initial guess x0: "))
        epsilon = float(input("Enter the tolerance: "))
        max_iter = int(input("Enter the maximum number of iterations: "))
        print("\nUsing Newton method:")
        return n.newton(f, df, x0, epsilon, max_iter)
    elif choice == 3:
        x0 = float(input("Enter the initial guess x0: "))
        epsilon = float(input("Enter the tolerance: "))
        max_iter = int(input("Enter the maximum number of iterations: "))
        print("\nUsing Modified Newton-Raphson method:")
        return m.modified(f, df, ddf, x0, epsilon, max_iter)
    elif choice == 4:
        x0 = float(input("Enter the first initial guess x0: "))
        x1 = float(input("Enter the second initial guess x1: "))
        epsilon = float(input("Enter the tolerance: "))
        max_iter = int(input("Enter the maximum number of iterations: "))
        print("\nUsing Secant method:")
        return s.secant(f, x0, x1, epsilon, max_iter)
    elif choice == 5:
        print("Exiting the program. Goodbye!")
        sys.exit()
    else:
        print("Invalid selection. Please try again.")
        return None
```

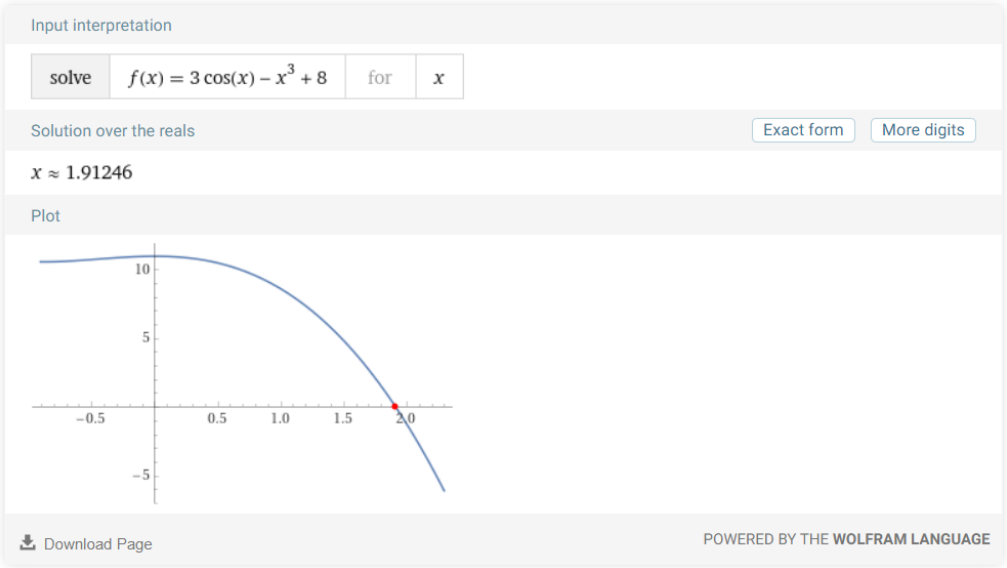
Figura 6 Menú en nuestro programa

Además de ofrecerle la respuesta al usuario, nuestro programa contiene una función para graficar los datos obtenidos utilizando la librería *matplotlib.pyplot.*

Ejemplo 1

$$f(x) = 3\cos(x) - x^3 + 8$$

Solución por WOLFRAM



Solución por Bisección

En el método de bisección se observa que los valores de a y b que determina el usuario no contienen a la solución por lo que se obtiene una excepción o error que declara “The scalars a and b do not bound the root”. Por lo que se observa que a causa de eso no se obtiene la raíz, esta es una de las limitantes de este método.

Entradas:

```
Enter your function in terms of x (e.g., x**2 - 2): 3 * cos(x)-x**3 + 8

Select a method to find the root of the function:
1 Bisection
2 Newton
3 Newton's Modified Method
4 Secant
5 Exit

Select the method number: 1
Enter the starting interval a: -10
Enter the ending interval b: 0
Enter the tolerance: 1e-3
```

Solución:

```
Using Bisection method:
Traceback (most recent call last):
  File "C:\Users\koqui\Desktop\P1E1\Menu.py", line 93, in <module>
    main()
  File "C:\Users\koqui\Desktop\P1E1\Menu.py", line 86, in main
    root = get_root(user_choice, f, df, ddf)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\koqui\Desktop\P1E1\Menu.py", line 27, in get_root
    return k.my_bisection(f, a, b, tol)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\koqui\Desktop\P1E1\MBisection.py", line 5, in my_bisection
    raise Exception ("The scalars a and b do not bound a root")
Exception: The scalars a and b do not bound a root
```

Solución por Newton

En el método de newton es necesario que la derivada de la función evaluada se diferente de 0 ya que esta en el denominador. En este caso la derivada de la función da 0 y ese es el error que marca el programa.

```
Enter your function in terms of x (e.g., x**2 - 2): 3*cos(x)-x**3+8

Select a method to find the root of the function:
1 Bisection
2 Newton
3 Newton's Modified Method
4 Secant
5 Exit

Select the method number: 2
Enter the initial guess x0: 0
Enter the tolerance: 1e-3
Enter the maximum number of iterations: 50

Using Newton method:
Zero derivate. No solution found
```

Solución por Newton Modificado

La solución por el método de Newton Modificado evade el error de la primera derivada, y en la segunda, ya que coseno de 0 es 1. Por lo tanto, converge la función. Sin embargo, el número de iteraciones proveídas con la tolerancia correspondiente y la aproximación inicial dada no fue adecuada para que se llegara a una solución.

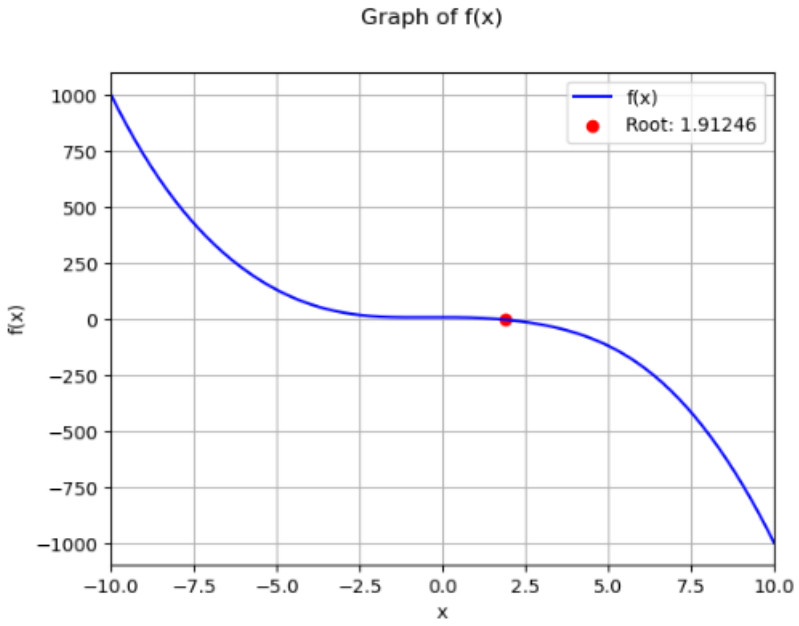
Aumentando el numero de iteraciones, haciendo mayor la tolerancia o con una mejor aproximación puede ser que el resultado pudiera ser proveído por este método.

```
Enter your function in terms of x (e.g., x**2 - 2): 3*cos(x)-x**3+8

Select a method to find the root of the function:
1 Bisection
2 Newton
3 Newton's Modified Method
4 Secant
5 Exit

Select the method number: 3
Enter the initial guess x0: 0
Enter the tolerance: 1e-3
Enter the maximum number of iterations: 50

Using Modified Newton-Raphson method:
Exceeded maximum iterations. No solution found
```

Solución por secante

Por último, la solución por medio de secante fue obtenida con el mismo numero de iteraciones y la tolerancia especificada. Por medio de secante se proveyeron dos aproximaciones, siendo la segunda más cercana al resultado. Por lo que al final el método resolvió la función.

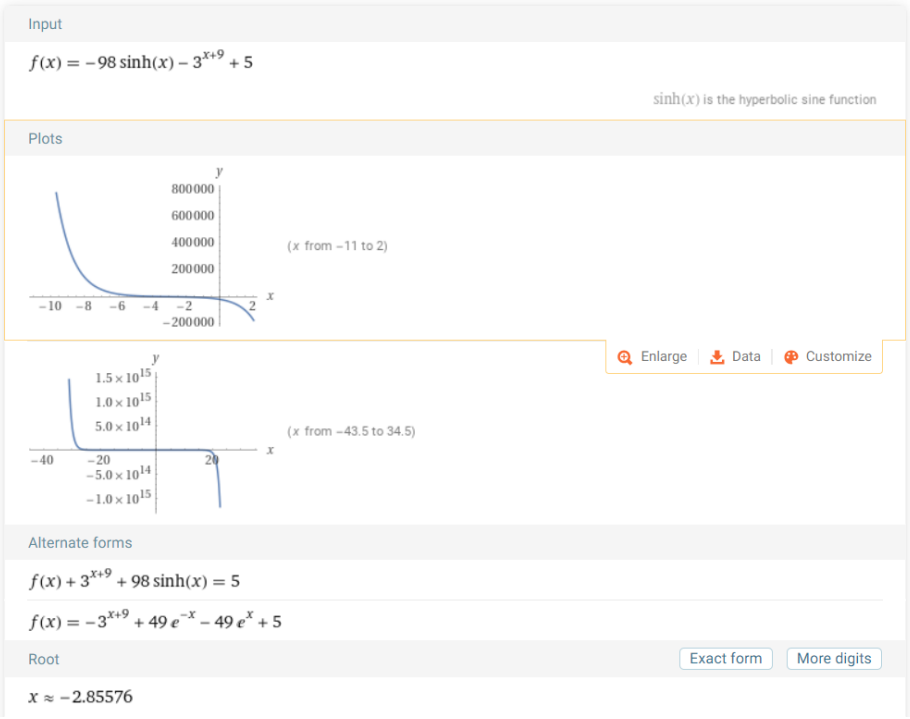
```
Enter your function in terms of x (e.g., x**2 - 2): 3*cos(x)-x**3+8

Select a method to find the root of the function:
1 Bisection
2 Newton
3 Newton's Modified Method
4 Secant
5 Exit

Select the method number: 4
Enter the first initial guess x0: 0
Enter the second initial guess x1: 1
Enter the tolerance: 1e-3
Enter the maximum number of iterations: 50

Using Secant method:
The root of the function 3*cos(x)-x**3+8 is approximately: 1.91247
```



Ejemplo 2

$$f(x) == -98senh(x) - 3^{x+9} + 5$$

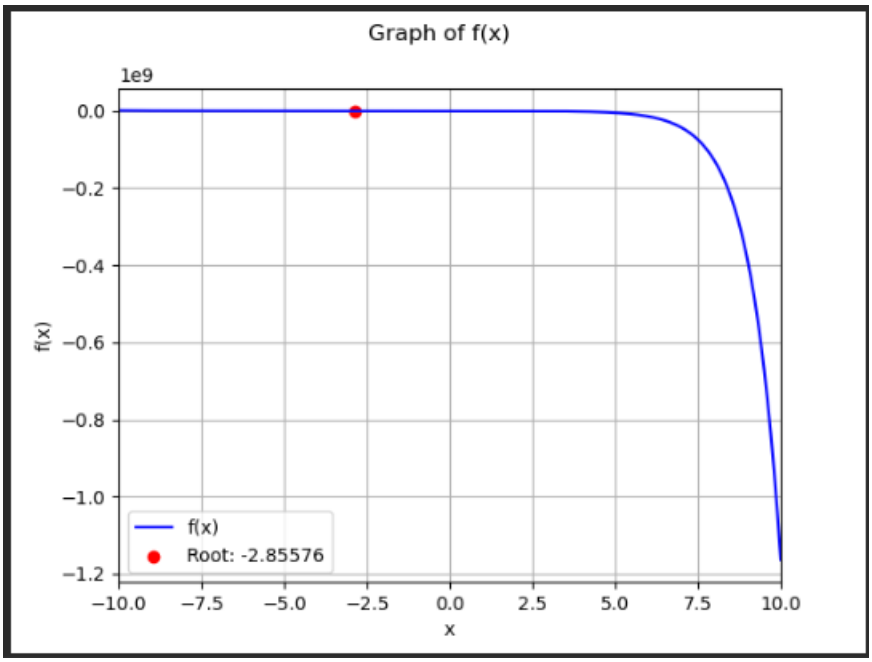
Solución por WOLFRAM



Solución por Bisección

Se observa que el método resuelve la ecuación al dar a y b correctamente y con una tolerancia aun mas alta de la esperada.

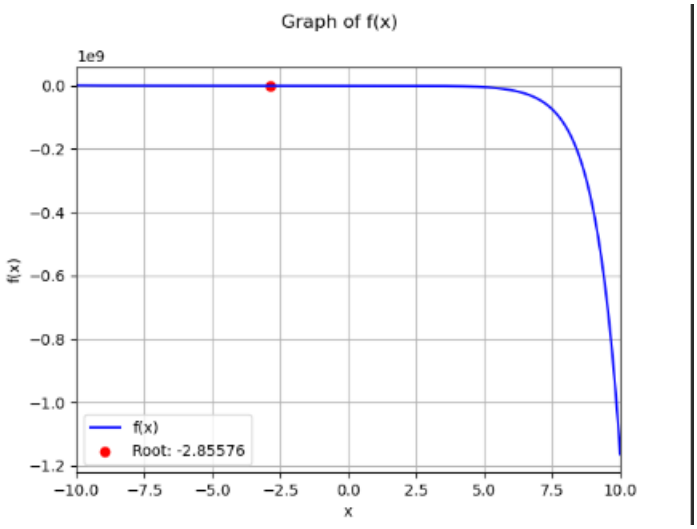
```
1 Bisection
2 Newton
3 Newton's Modified Method
4 Secant
5 Exit

Select the method number: 1
Enter the starting interval a: -20
Enter the ending interval b: 20
Enter the tolerance: 1e-3

Using Bisection method:
The root of the function -98*sinh(x) - 3 ** (x+9) +5 is approximately: -2.85576
```



Solución por Newton

Newton soluciona el problema y de igual que el anterior problema se grafica la solución.

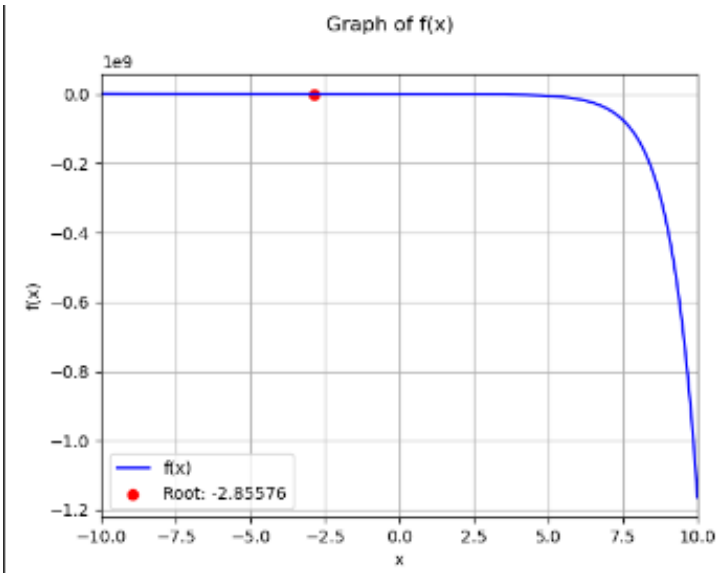
```
Select a method to find the root of the function:
1 Bisection
2 Newton
3 Newton's Modified Method
4 Secant
5 Exit

Select the method number: 2
Enter the initial guess x0: -1
Enter the tolerance: 1e-5
Enter the maximum number of iterations: 25

Using Newton method:
Found solution after n iterations
The root of the function  -98*sinh(x) - 3 ** (x+9) +5 is approximately: -2.85576
```



Solución por Newton Modificado

Este método pudo obtener la raíz con las especificaciones de error e iteraciones proporcionadas.

```
Enter your function in terms of x (e.g., x**2 - 2): -98*sinh(x) - 3 ** (x+9) +5

Select a method to find the root of the function:
1 Bisection
2 Newton
3 Newton's Modified Method
4 Secant
5 Exit

Select the method number: 4
Enter the first initial guess x0: -5
Enter the second initial guess x1: 1e-1
Enter the tolerance: 1e-2
Enter the maximum number of iterations: 200

Using Secant method:
The root of the function -98*sinh(x) - 3 ** (x+9) +5 is approximately: -2.85576
```



Solución por Secante

En este caso el método genera la búsqueda de las raíces por medio de las raíces de las aproximaciones iniciales. En este caso las raíces generaron el error de una división con cero y por lo tanto se incita al usuario usar un valor inicial de aproximación diferente.

```
Enter your function in terms of x (e.g., x**2 - 2): -98*sinh(x) - 3 ** (x+9) +5

Select a method to find the root of the function:
1 Bisection
2 Newton
3 Newton's Modified Method
4 Secant
5 Exit

Select the method number: 4
Enter the first initial guess x0: -100
Enter the second initial guess x1: 20
Enter the tolerance: 1e-3
Enter the maximum number of iterations: 50

Using Secant method:
Division by zero encountered. Try different initial guesses.
```

Referencias

Burden, R. L., Faires, J., & Burden, A. (2017). *Análisis numérico (10a. ed.).*