Manual de Usuario SolNE - Python

Kenneth Hernández, Marco Herrera, and Jasson Rodríguez

Instituto Tecnológico de Costa Rica

CE - 3102 Análisis Numérico para Ingeniería

Agosto 2019

Índice

1. ¿Que es SolNe?

SolNe es un paquete computacional de python, que como su nombre indica, se encuentra orientado a la aproximación de una solución para ecuaciones no lineales; mediante métodos iterativos. El paquete cuenta con 12 métodos iterativos; los cuales se dividen en aquellos que hacen uso de derivadas y aquellos que no. Los métodos permiten al usuario especificar el error de la solución que esperan obtener, esto mediante un parámetro de tolerancia. Todos los

métodos utilizan el error absoluto de la función; para su condición de parada. Además, para cada método el usuario tiene la elección de generar una gráfica de errores en función del número de iteraciones; lo cual permite apreciar la convergencia de los diferentes métodos implementados en el paquete y como se comportan estos ante distintas funciones de entrada.

2. Instalación SolNe

2.1. Prerrequisitos

Dado que este paquete es computacional es desarrollado para Python; se requieren conocimientos básicos sobre este, además de una instalación de las versiones más actualizadas. Lo anterior puede encontrarse en la página oficial de Python. Además se requiere el manejador de paquetes PIP. En adición a lo anterior; el paquete hace uso de los paquetes Sympy y Matplotlib, los cuales serán instalados por el paquete SolNE en la sección siguiente. Además se utiliza la biblioteca Math; sin embargo, en la mayoría de los casos, esta se encuentra instalada por defecto.

2.2. Instalación

Si bien se recomienda tener los requisitos de la sección anterior preinstalados; en caso de que este no sea el caso, SolNE intentará instalarlos mediante el manejador de paquetes pip. Para instalar SolNE, basta con descomprimir el paquete SolNE.zip, y una vez ubicado dentro de este en la terminal; al mismo nivel que el archivo setup.py debe ejecutar el comando mediante la consola: **pip install .**, no olvide el punto al final. Y requerde que debe estar en la misma carpeta que el archivo setup.py. Una vez sea instalado verá un mensaje en la conpesola diciendo, **Successfully installed SolNE-1.** para comprobar la correcta instalación muevase dentro de la siguiente carpeta SolNE y ejecute el archivo llamado test.py, el cual le mostrará una gráfica seguida de otra, conforme las cierre.

3. Uso de SolNe

En esta sección se explicara el como usar las diferentes funciones integradas en el paquete computacional SolNe, tanto sus parámetros como los diferentes resultados que estas calculan.

3.1. Parámetros

El conjunto de métodos iterativos implementados sigue en su mayoría una estructura similar, cuyos parámetros son el valor inicial, la tolerancia, la función y la gráfica. Algunas otras funciones requieren de parámetros extras que se explicaran en cada una de estas. A continuación se presenta una breve explicación de cada uno de los parámetros principales.

3.1.1. funcion

El parámetro función, como lo sugiere el propio nombre del paquete, es una ecuación no lineal escrita en forma de String, a la cual se le quiere aproximar una solución,la ecuación es dependiente del parámetro x y cumple con la siguiente característica:

$$f(x) = 0 \tag{1}$$

Para este paquete se utilizo el lector de funciones que pertenece a Sympy, por lo cual es recomendable que se consulte su documentación, en todo caso la siguiente tabla contendrá operaciones básicas aceptadas por el lector de Sympy:

Operación	Equivalente
x^a	$x**a$
ax	$a*x$
$x+a$	$x+a$
$\frac{x}{a}$	a/x
$sen(x)$	$sin(x)$
$cos(x)$	$cos(x)$
$tan(x)$	$tan(x)$
e^x	$E**x$
$a+x$	$a+x$
$a-x$	$a-x$
$ln(x)$	$ln(x)$

El siguiente es un ejemplo de como se vería una función matemática y su representación aceptada por el lector de funciones:

$$\frac{ln(x)x - e^x}{x+1-sen(x)} -> \text{“}(ln(x)*x - E**x)/(x+1-sin(x))\text{”} \tag{2}$$

También se presentan ciertos casos en los cuales la función puede generar errores, como lo puede ser una ecuación se encuentre mal escrita como en la figura 1 o que su derivada sea 0 en alguna iteración, como en la figura 2 (este es exclusivo de las funciones que involucran derivadas), si se presentan estos casos en la consola se podrá apreciar lo siguiente:

Figura 1: Excepción en caso de función mal escrita

Figura 2: Excepción en caso de derivada igual a 0

3.1.2. graf

El parámetro graf esta relacionado con la capacidad de las funciones en generar un gráfico de $\mid f(x) \mid$ contra las iteraciones realizadas para la aproximación. Este gráfico se mostrara solo si el usuario no introduce el argumento o lo define como 1, si no se quiere mostrar se define como 0, en cualquier otro caso se mostrara un mensaje.En la figura 3 se puede apreciar un mensaje en el caso de que se introduzca un argumento diferente a 1 y 0, mientras que en la figura 4 se muestra el caso en que no se introduce el argumento o se define como 1.

```
>>> ud.sne_ud_1(3, 1, 0.0001, "x**2-4", 4)
WARNING: graf has two possible values, 1 or 0

(2.0000000000262146, 2)

>>>
```

Figura 3: Excepción en caso de graf diferente de 0 y 1

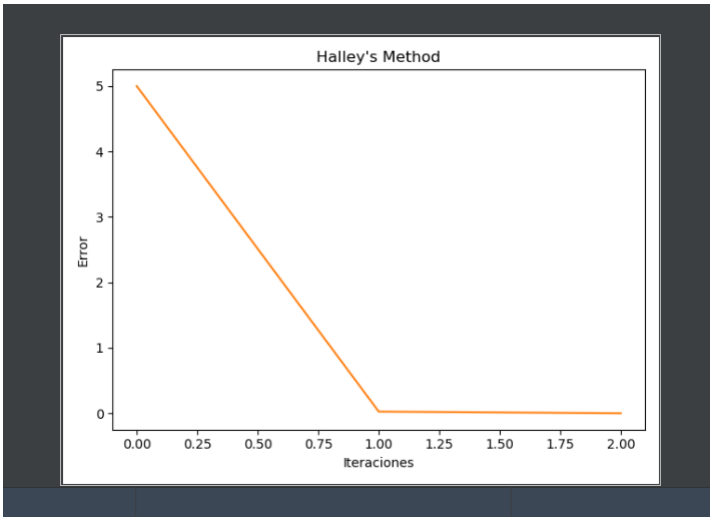

Figura 4: Gráfico generado sin introducir graf o con graf igual a 1

3.1.3. x0

Este parámetro es el valor inicial con el cual se empezaran los cálculos en el método iterativo, puede ser tanto un numero entero como un double, en particular este parámetro no genera problemas a la hora de realizar los cálculos a excepción en los cuales un divisor se convierta en 0 o se indefina una función; en dichos casos el método se detendrá y se mostrará el error generado por sympy, además de los resultados obtenidos hasta el momento. El valor inicial también puede causar que según el método este diverja, lo cual no genera ningún error sino que muestra el valor aproximado luego de que se alcance el límite de iteraciones, el método también llega a un máximo de iteraciones cuando la función no tiene solución. En algunas funciones se pedirá un segundo valor inicial para realizar los cálculos, pero esto se aclarara en la definición de cada función. En la figura 5 se aprecia el caso de que un valor indefina una función, y en la figura 6 se encuentra el caso de que un método sobre una función sin solución real.

```
>>> from SolNE import ud
>>> ud.sne_ud_1(2, 1, 0.0001, "ln(x) - x -1", 1)
<string>:2: RuntimeWarning: invalid value encountered in log
(-1.4806123590663098, 2)

>>>
```

Figura 5: Excepción generada por una función indefinida

Figura 6: Método sin encontrar solución

3.1.4. tol

La tolerancia es el parámetro que indica cuando terminar la ejecución de la función, ya que todas las funciones detienen el ciclo de iteraciones y retornan resultados cuando se cumple que $\mid f(x) \mid$ es menor que la tolerancia dada. Este parámetro puede ser tanto un entero como un double o float positivo, y el único error asociado que se tiene es cuando su valor es negativo, de tal manera que se genera un mensaje, como se puede ver en la figura 7

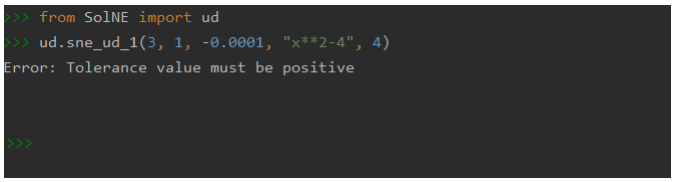

Figura 7: Excepción de tolerancia negativa

3.2. Funciones

En esta sección se demostraran las diferentes funciones junto a un par de ejemplos de ejecución por cada uno, esto con el objetivo de que los parámetros que hemos discutido anteriormente queden lo suficientemente claros junto a que hace cada función.Un aspecto general es que todas las funciones retornan la aproximación a la función no lineal y la cantidad de iteraciones necesarias para alcanzar dicha aproximación.

3.2.1. set_max_iter

Esta función es la utilizada para definir la cantidad máxima de iteraciones que pueden realizar las demás funciones que realizan cálculos iterativos, su objetivo es evitar que las funciones se ejecuten de forma indefinida, su sintaxis es la siguiente:

```
>>set_max_iter(max_iter)
```

Listing 1: Ejemplo de uso de set_max_iter

Donde max_iter es la cantidad máxima de iteraciones que se quiere ejecuten las funciones.

3.2.2. help

La función help es la utilizada para conseguir información, esta tiene como argumento el nombre de una función, dando como resultado toda la información sobre ella, su sintaxis es help(**nombrefuncion**) como se puede apreciar en el siguiente ejemplo:

```
>>> help(ud.sne_ud_1)
Help on function sne_ud_1 in module SolNE.ud:

sne_ud_1(x0, alpha, tol, funcion, graf=1)
    Implementación del método de Halley y variaciones
    Entradas:
    -x0: valor inicial de iteración (tipo: numérico real)
    -alpha: selecciona entre variaciones del metodo, 0 para Chebyshev's, 1/2 para Halley's y 1 para Super-Halley's
    -tol: tolerancia mínima del error (tipo: numérico real positivo)
    -funcion: función sobre la cual iterar, siguiendo lineamientos de sympy (tipo: cadena de caracteres)
    -graf: bandera para graficar o no el error de la función (tipo: numérico 1 o 0)
    Salidas:
    -Valor aproximado de la solución según tolerancia indicada o hasta que se indefina el procedimiento
    -Cantidad de iteraciones realizadas según tolerancia indicada o hasta que se indefina el procedimiento
    -Gráfica del error en función del número de iteraciones


>>>
```

Figura 8: Ejemplo de uso de la funcion help

3.2.3. sne_ud_1

Esta función esta basada en el método de Halley y algunas variaciones de este, la cual se ve representada en la función 1 de la pagina 939 del articulo«A stable class of improved second-derivative free Chebyshev-Halley type methods with optimal eighth order convergence»[1] , su sintaxis es sne_ud_1(x0, alpha, tol, funcion, graf), donde el parámetro alpha sirve para seleccionar distintas variaciones del método, 0 para Chebyshev's, 1/2 para Halley's y 1 para Super-Halley's. Al utilizar este método la primera derivada de la función, rechaza toda función en la cual $f'(x) = 0$. A continuación se presentan 2 ejemplos de ejecución de la función:

```
>> sne_ud_1(100, 0, 0.0001, "E**x+3*x-sin(x)", 1)
>> (-0.34912703619836616, 68)
```

Listing 2: Ejemplo de uso de la función

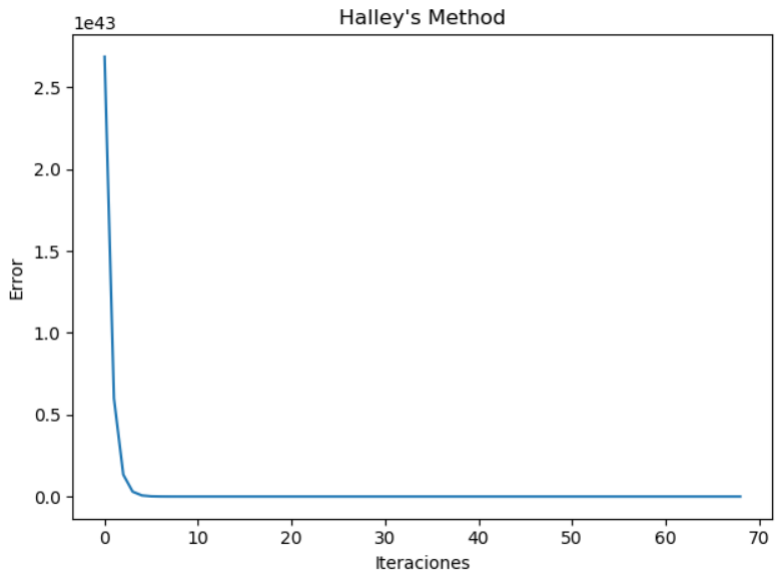

Figura 9: Ejemplo 1 de sne_ud_1

```
>>> sne_ud_1(5, 1/2, 0.0001, "x**x + tan(x)", 1)
```

```
(1.8710873027560853, 6)
```

Listing 3: Ejemplo de uso de la función

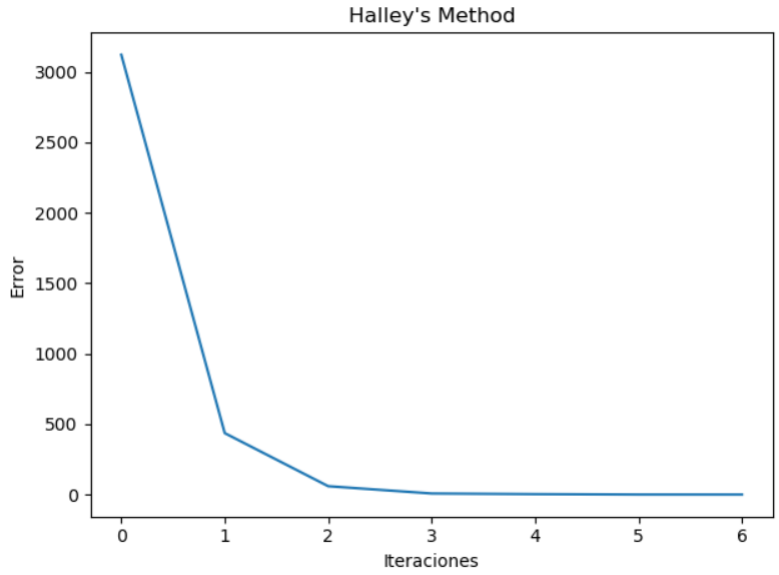

Figura 10: Ejemplo 2 de sne_ud_1

3.2.4. sne_ud_2

Esta función esta basada en el método de Chun y Neta, la cual se ve representada en la función 1 de la pagina 154 del articulo «On improved three-step schemes with high efficiency index and their dynamics»[2], su sintaxis es sne_ud_2(x0, tol,funcion,graf), al utilizar este método la primera derivada de la función, rechaza toda función en la cual $f'(x) = 0$. A continuación se presentan 2 ejemplos de ejecución de la función:

```
>>>sne_ud_2(100, 0.0001, "E**x+3*x-sin(x)", 1)
(-0.34912767940523237, 62)
```

Listing 4: Ejemplo de uso de la función

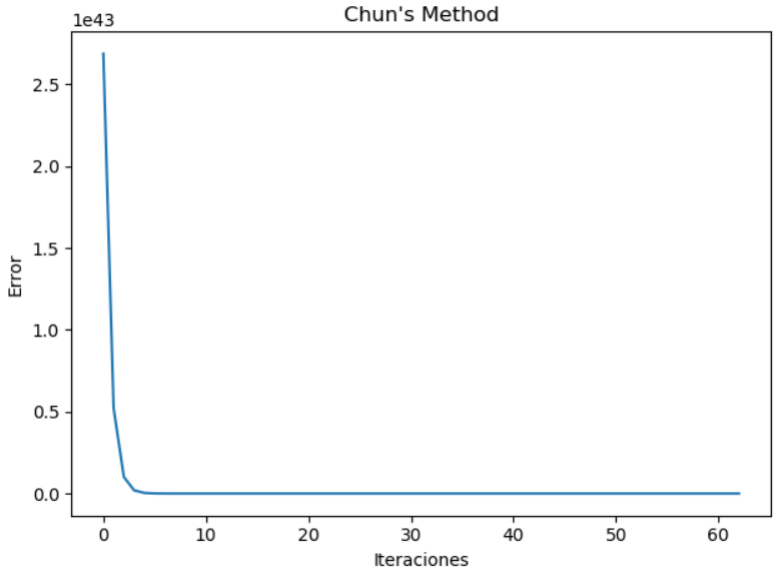

Figura 11: Ejemplo 1 de sne_ud_2

```
>>>sne_ud_2(5, 0.0001, "x**x + tan(x)", 1)
(1.8710917786642938, 6)
```

Listing 5: Ejemplo de uso de la función

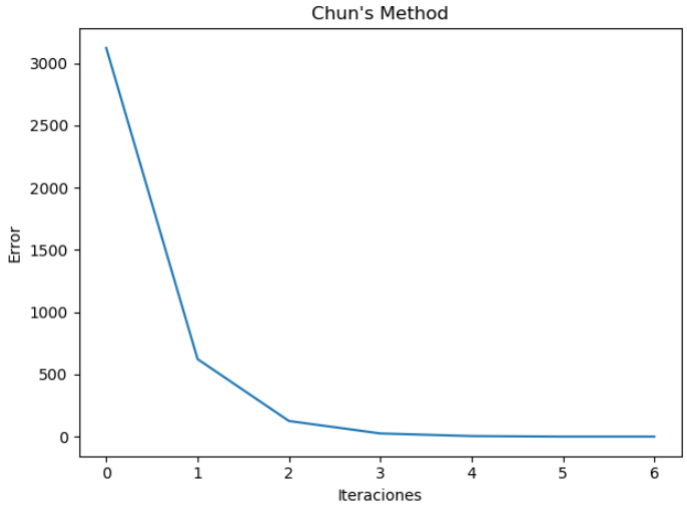

Figura 12: Ejemplo 2 de sne_ud_2

3.2.5. sne_ud_3

Esta función esta basada en el método de Traub, la cual se ve representada en la función 1 de la pagina 1065 del articulo «Multidimensional generalization of iterative methods for solving nonlinear problems by means of weight-function procedure»[3], su sintaxis es sne_ud_3(x0, tol,funcion,graf), al utilizar este método la primera derivada de la función, rechaza toda función en la cual $f'(x) = 0$. A continuación se presentan 2 ejemplos de ejecución de la función:

```
>>>sne_ud_3(100, 0.0001, "E**x+3*x-sin(x)", 1)
(-0.34912705624245977, 75)
```

Listing 6: Ejemplo de uso de la función

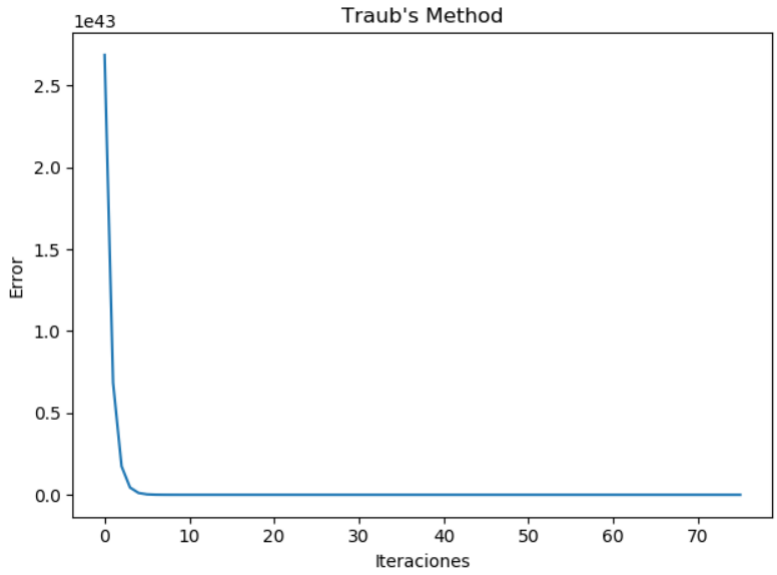

Figura 13: Ejemplo 1 de sne_ud_3

```
>>>sne_ud_3(5, 0.0001, "x**x + tan(x)", 1)
(1.8710964183986993, 7)
```

Listing 7: Ejemplo de uso de la función

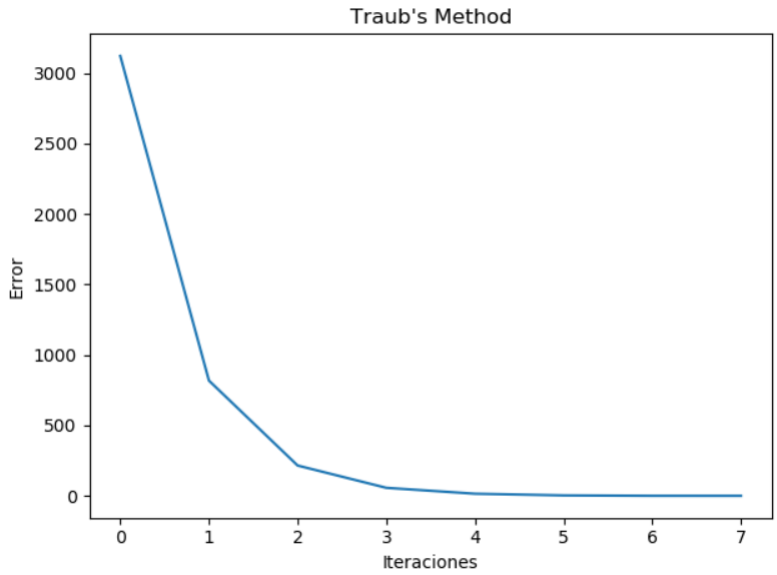

Figura 14: Ejemplo 2 de sne_ud_3

3.2.6. sne_ud_4

Esta función esta basada en el método de Frontini y Sormani , la cual se ve representada en la función 7 de la pagina 85 del articulo «Performance of cubic convergent methods for implementing nonlinear constitutive models»[4], su sintaxis es sne_ud_4(x0, tol,funcion,graf), al utilizar este método la primera derivada de la función, rechaza toda función en la cual $f'(x) = 0$. A continuación se presentan 2 ejemplos de ejecución de la función:

```
>>>sne_ud_4(100, 0.0001, "E**x+3*x-sin(x)", 1)
(-0.3491270583142914, 62)
```

Listing 8: Ejemplo de uso de la función

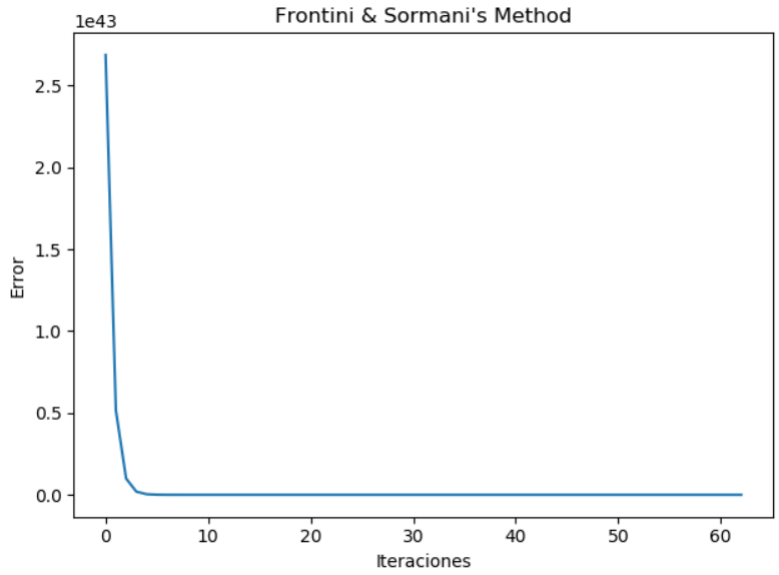

Figura 15: Ejemplo 1 de sne_ud_4

```
>>>sne_ud_4(5, 0.0001, "x**x + tan(x)", 1)
(1.8710917786859795, 7)
```

Listing 9: Ejemplo de uso de la función

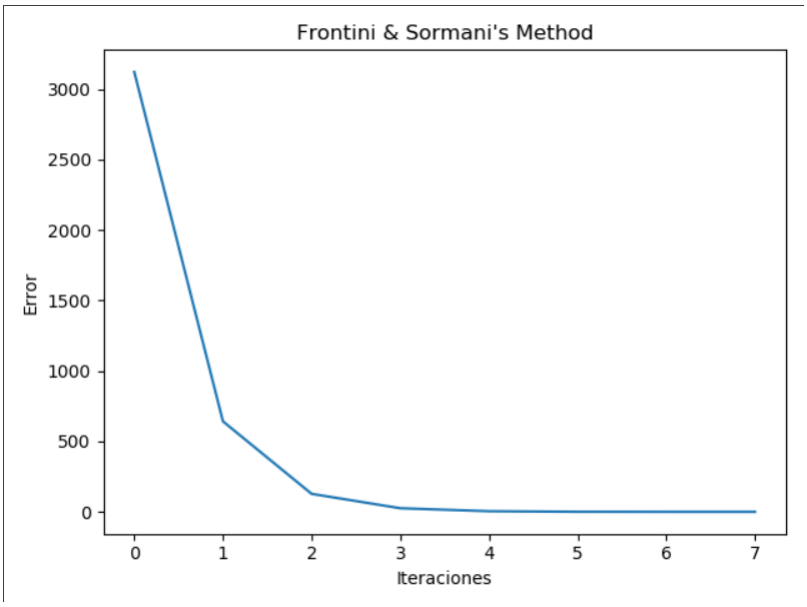

Figura 16: Ejemplo 2 de sne_ud_4

3.2.7. sne_ud_5

Esta función esta basada en el método de Weerakoon y Fernando, la cual se ve representada en la función 5 de la pagina 84 del articulo «Performance of cubic convergent methods for implementing nonlinear constitutive models»[4], su sintaxis es sne_ud_5(x0, tol,funcion,graf), al utilizar este método la primera derivada de la función, rechaza toda función en la cual $f'(x) = 0$ A continuación se presentan 2 ejemplos de ejecución de la función:

```
>>>sne_ud_5(100, 0.0001, "E**x+3*x-sin(x)", 1)
(-0.3491270501609614, 70)
```

Listing 10: Ejemplo de uso de la función

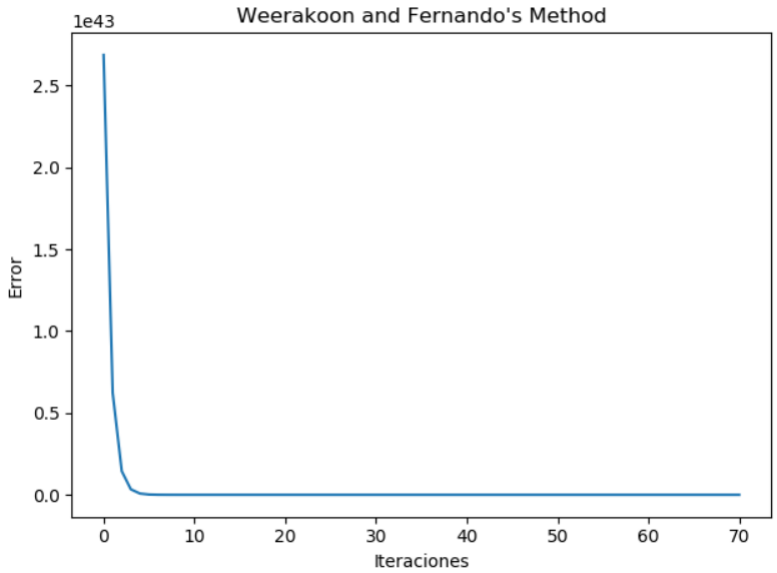

Figura 17: Ejemplo 1 de sne_ud_5

```
>>>sne_ud_5(5, 0.0001, "x**x + tan(x)", 1)
(1.8710924163806841, 7)
```

Listing 11: Ejemplo de uso de la función

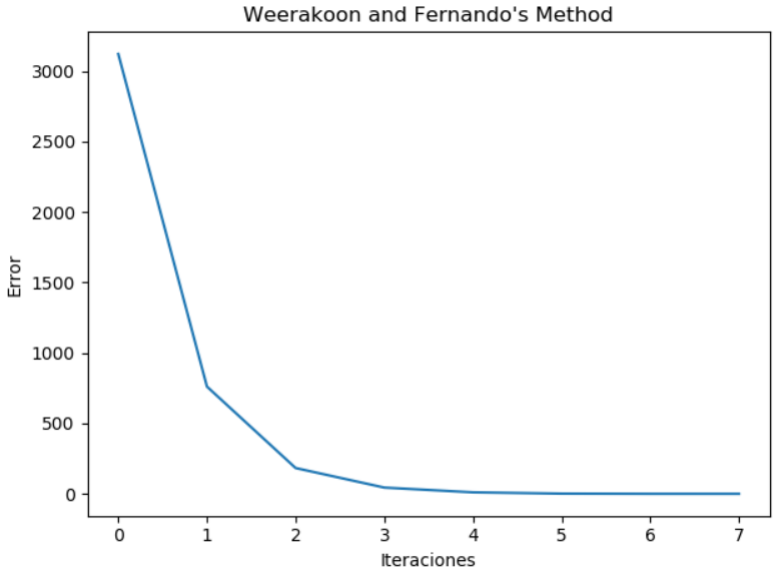

Figura 18: Ejemplo 2 de sne_ud_5

3.2.8. sne_ud_6

Esta función esta basada en el método de Dong, la cual se ve representada en la ecuación X de la pagina X del articulo XXXXXXX, su sintaxis es sne_ud_6(x0, m, tol,funcion,graf), donde m representa la multiplicidad de la raíz de la función. Al utilizar este método la primera derivada de la función, rechaza toda función en la cual $f'(x) = 0$. A continuación se presentan 2 ejemplos de ejecución de la función, uno de estos ejemplos es diferente a los anteriores pues este método se aplica para funciones con raíces de distinta multiplicidad, este parámetro se especifica en la ejecución:

```
>>>sne_ud_6(20, 2, 0.0001, "(x-2)**2*(x+3)*E**x", 1)
(2.000011384224617, 10)
```

Listing 12: Ejemplo de uso de la función

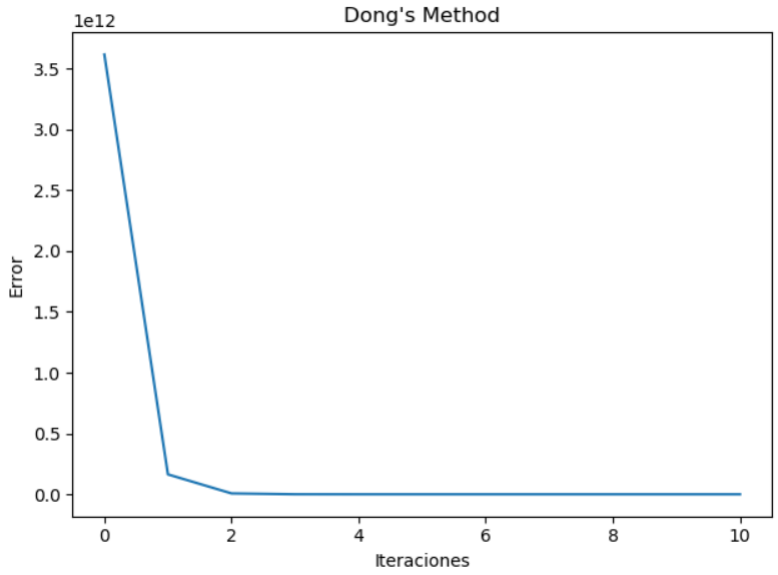

Figura 19: Ejemplo 1 de sne_ud_6

```
>>>sne_ud_6(5, 1, 0.0001, "x**x + tan(x)", 1)
(1.8710964183986993, 7)
```

Listing 13: Ejemplo de uso de la función

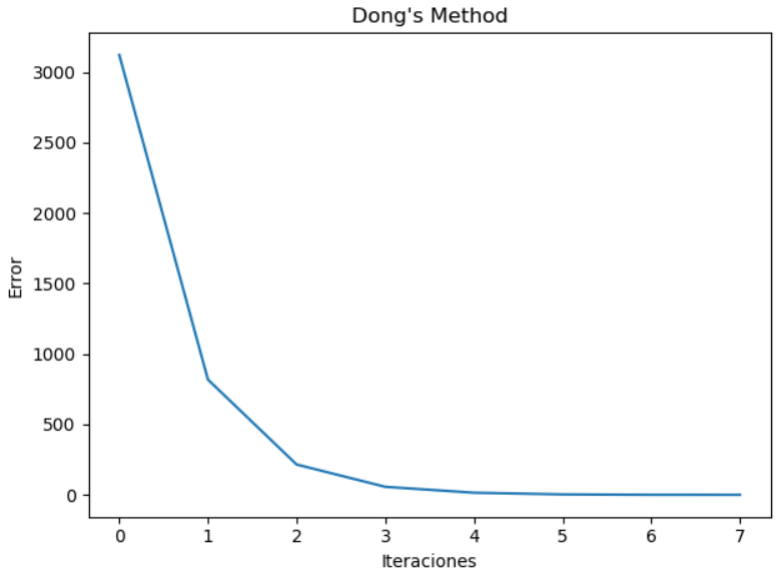

Figura 20: Ejemplo 2 de sne_ud_6

3.2.9. sne_fd_1

Esta función esta basada en el método de Ostrowski, la cual se ve representada en la ecuación 3 de la pagina 3059 del articulo «Steffensen type methods for solving nonlinear equations»[5], su sintaxis es sne_fd_1(x0, tol,funcion,graf), al no utilizar este método la primera derivada de la función, acepta funciones en la cual $f'(x) = 0$. A continuación se presentan 2 ejemplos de ejecución de la función:

```
>>>sne_fd_1(100, 0.000000001, "x**2+3*x-10", 1)
(2.0, 4)
```

Listing 14: Ejemplo de uso de la función

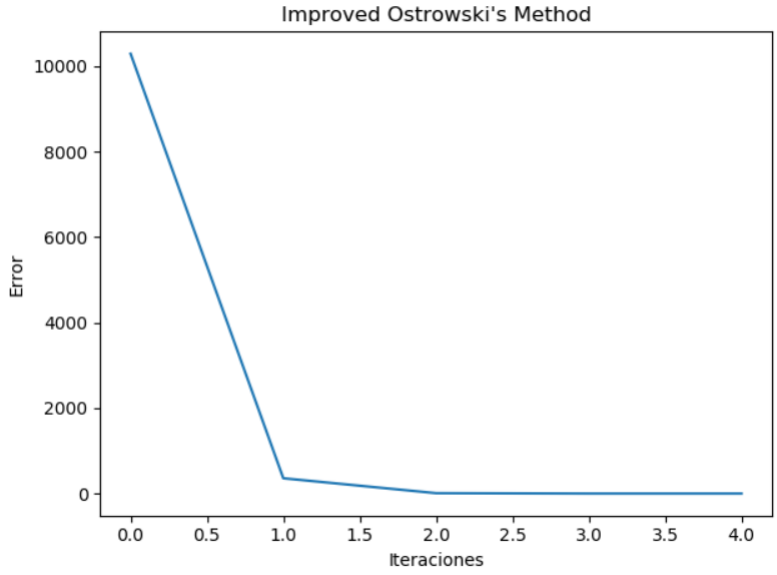

Figura 21: Ejemplo 1 de sne_fd_1

```
>>>sne_fd_1(1, 0.000000001, "x**4+sin(pi/x**2)-5", 1)
(1.414213562373158, 13)
```

Listing 15: Ejemplo de uso de la función

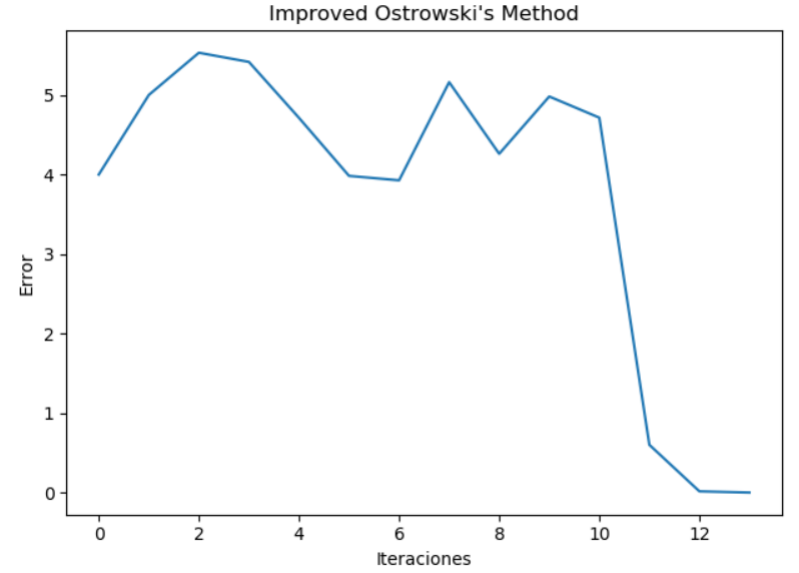

Figura 22: Ejemplo 2 de sne_fd_1

3.2.10. sne_fd_2

Esta función esta basada en el método de Steffensen, la cual se ve representada en la ecuación 1.2 de la pagina 3059 del articulo «Steffensen type methods for solving nonlinear equations»[5], su sintaxis es sne_fd_1(x0, tol,funcion,graf), al no utilizar este método la primera derivada de la función, acepta funciones en la cuales $f'(x) = 0$. A continuación se presentan 2 ejemplos de ejecución de la función:

```
>>>sne_fd_2(100, 0.000000001, "x**2+3*x-10", 1)
(2.000000000000181, 110)
```

Listing 16: Ejemplo de uso de la función

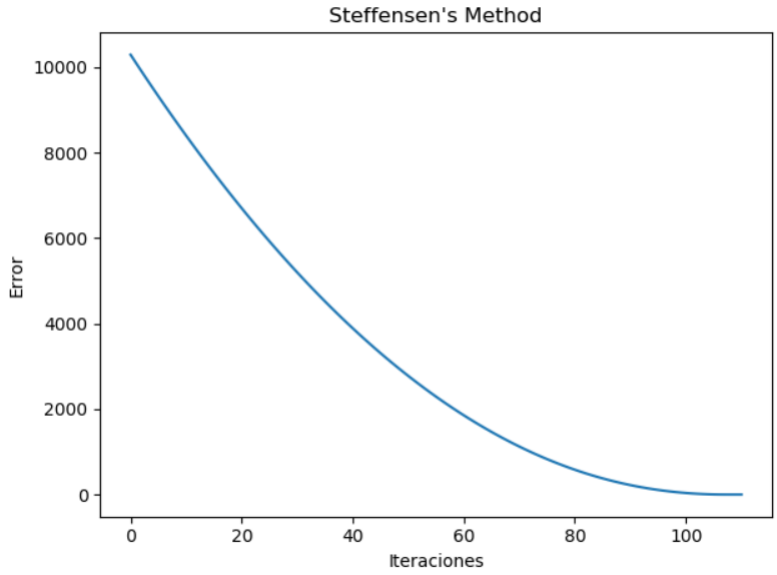

Figura 23: Ejemplo 1 de sne_fd_2

```
>>>sne_fd_2(1, 0.000000001, "x**4+sin(pi/x**2)-5", 1)
(-1.4142135623457257, 65)
```

Listing 17: Ejemplo de uso de la función

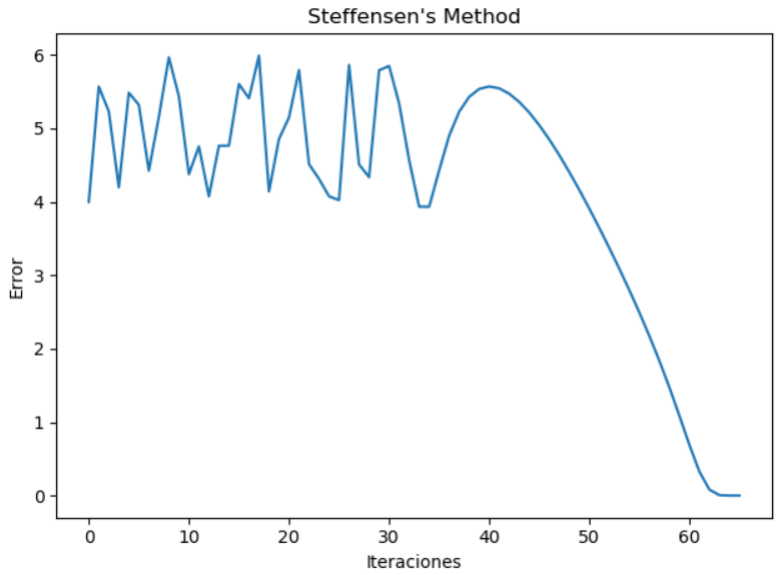

Figura 24: Ejemplo 2 de sne_fd_2

3.2.11. sne_fd_3

Esta función esta basada en el método mejorado de Ren, la cual se ve representada en la ecuación 3 de la pagina 7664 del articulo «A class of Steffensen type methods with optimal order of convergence»[6], su sintaxis es sne_fd_3(x0, tol,funcion,graf), al no utilizar este método la primera derivada de la función, acepta funciones en la cuales $f'(x) = 0$. A continuación se presentan 2 ejemplos de ejecución de la función:

```
>>>sne_fd_3(100, 1, 0.000000001, "x**2+3*x-10", 1)
(2.0, 6)
```

Listing 18: Ejemplo de uso de la función

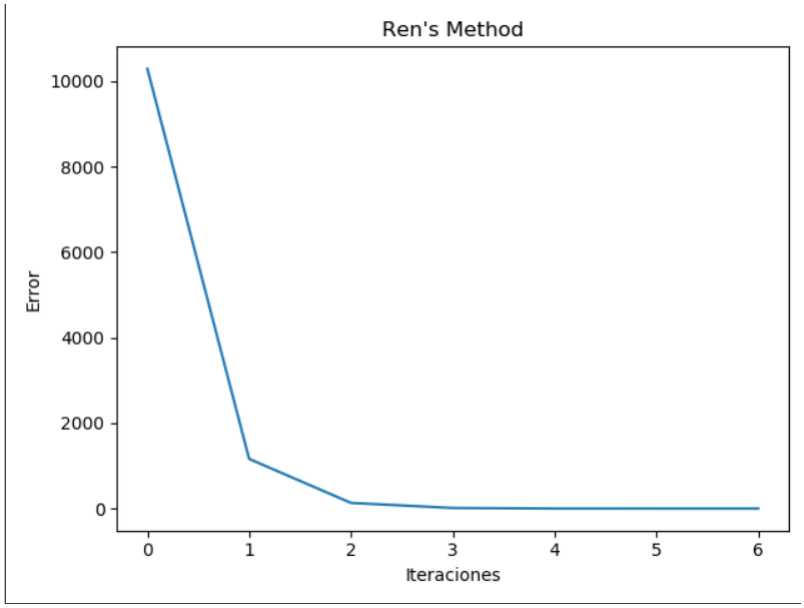

Figura 25: Ejemplo 1 de sne_fd_3

```
>>>sne_fd_3(1, 1, 0.000000001, "x**4+sin(pi/x**2)-5", 1)
(1.4142135623799903, 4)
```

Listing 19: Ejemplo de uso de la función

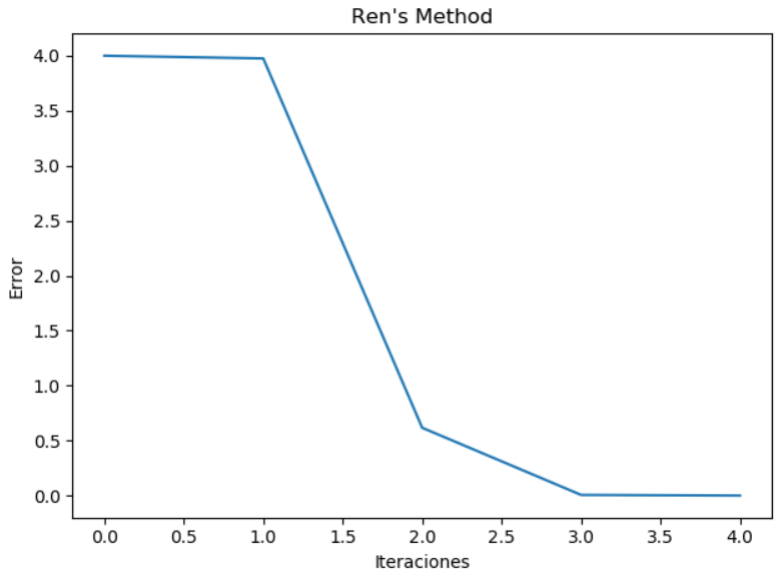

Figura 26: Ejemplo 2 de sne_fd_3

3.2.12. sne_fd_4

Esta función esta basada en el método de Kurchatov , la cual se ve representada en la ecuación 4 de la pagina 365 del articulo «A family of Kurchatov-type methods and its stability»[7] , su sintaxis es sne_fd_4(x0,prev, tol,funcion,graf),esta función tiene un parametro extra llamado prev, el cual es otro valor para calcular la solucion, al no utilizar este método la primera derivada de la función, acepta funciones en la cuales $f'(x) = 0$. A continuación se presentan 2 ejemplos de ejecución de la función:

```
>>>sne_fd_4(100, 0, 0.000000001, "x**2+3*x-10", 1)
(2.000000000000003, 9)
```

Listing 20: Ejemplo de uso de la función

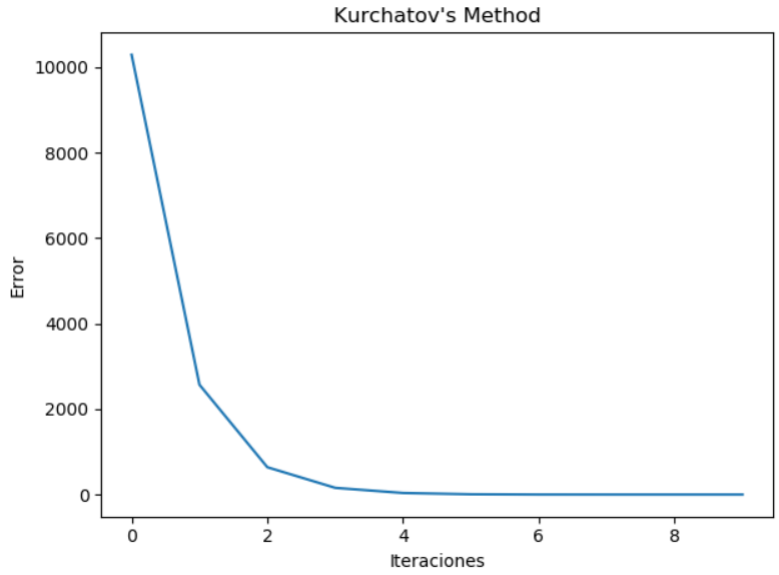

Figura 27: Ejemplo 1 de sne_fd_4

```
>>>sne_fd_4(2, 1, 0.000000001, "x**4+sin(pi/x**2)-5", 1)
(1.4142135623730951, 6)
```

Listing 21: Ejemplo de uso de la función

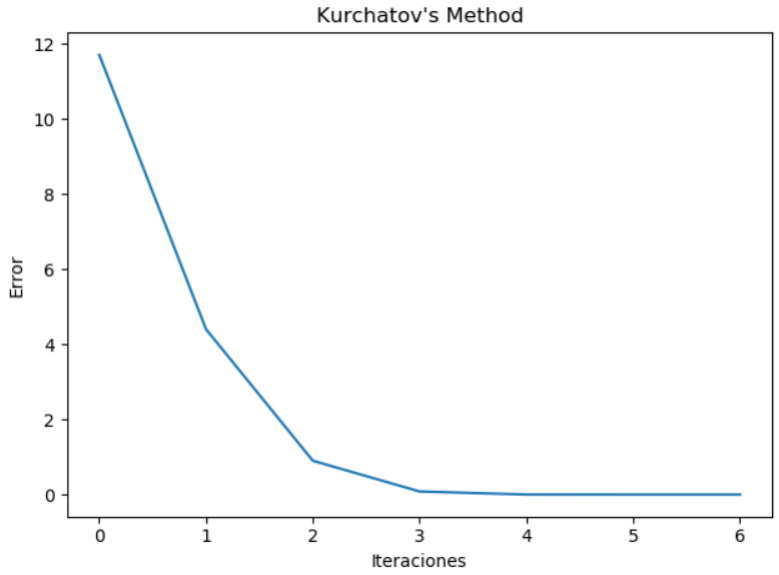

Figura 28: Ejemplo 2 de sne_fd_4

3.2.13. sne_fd_5

Esta función esta basada en el método de Jain, la cual se ve representada en la ecuación 1 de la pagina 7654 del articulo «A class of Steffensen type methods with optimal order of convergence»[6], su sintaxis es sne_fd_5(x0, tol,funcion,graf), al no utilizar este método la primera derivada de la función, acepta funciones en la cuales $f'(x) = 0$. A continuación se presentan 2 ejemplos de ejecución de la función:

```
>>>sne_fd_5(100, 0.000000001, "x**2+3*x-10", 1)
(2.0, 8)
```

Listing 22: Ejemplo de uso de la función

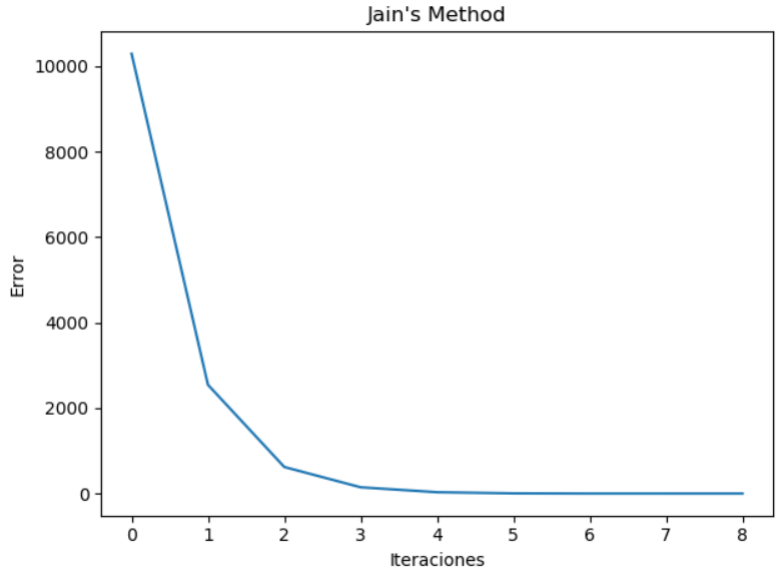

Figura 29: Ejemplo 1 de sne_fd_5

```
>>>sne_fd_5(1, 0.000000001, "x**4+sin(pi/x**2)-5", 1)
(1.4142135623730951, 4)
```

Listing 23: Ejemplo de uso de la función

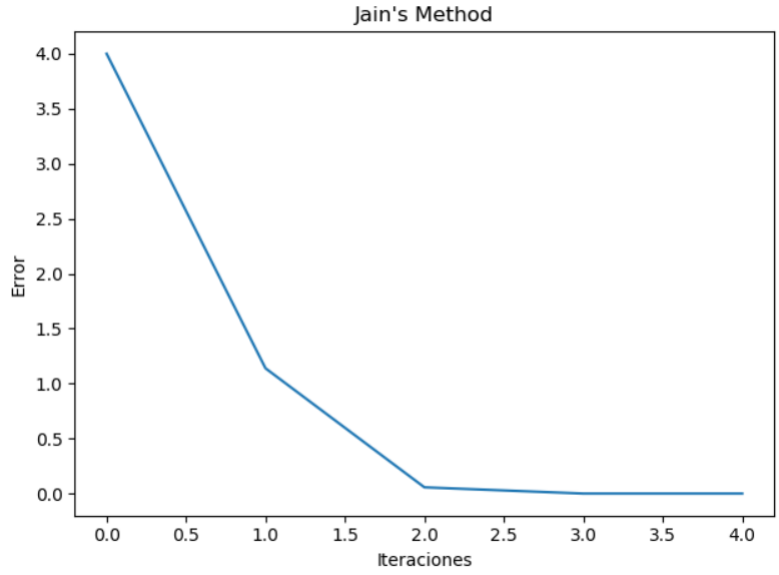

Figura 30: Ejemplo 2 de sne_fd_5

3.2.14. sne_fd_6

Esta función esta basada en el método de Zheng, la cual se ve representada en la ecuación 20 de la pagina 329 del articulo «An efficient two-parametric family with memory for nonlinear equations»[?], su sintaxis es sne_fd_6(x0, gamma, tol,funcion,graf), al no utilizar este método la primera derivada de la función, acepta funciones en la cuales $f'(x) = 0$. El valor de gamma es una constante numérica, la cual modifica el comportamiento, el valor de esta depende de la función que se esté utilizando, lo cual hace que converja a distintas velocidades; los valores más comunes son 1 y 2. A continuación se presentan 2 ejemplos de ejecución de la función:

```
>>>sne_fd_6(100, 1, 0.000000001, "x**2+3*x-10", 1)
(2.00000000000162, 7)
```

Listing 24: Ejemplo de uso de la función

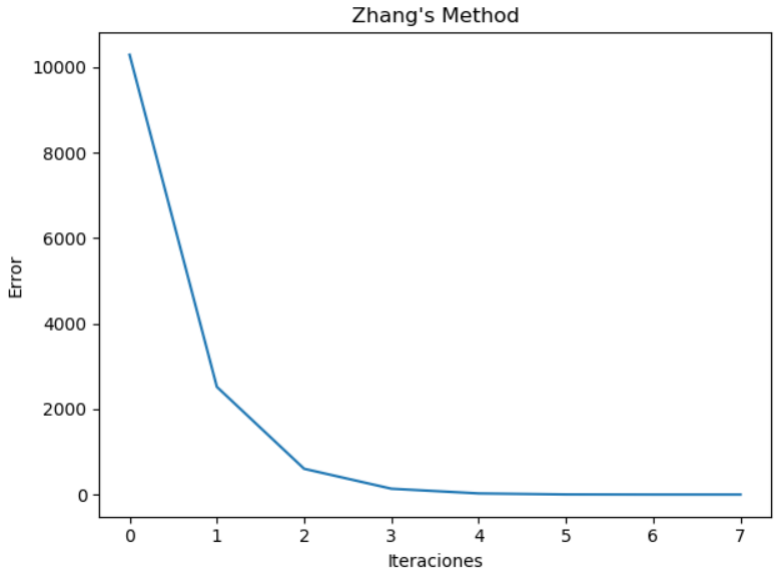

Figura 31: Ejemplo 1 de sne_fd_6

```
>>>sne_fd_6(1, 1, 0.000000001, "x**4+sin(pi/x**2)-5", 1)
(1.414213562373095, 4)
```

Listing 25: Ejemplo de uso de la función

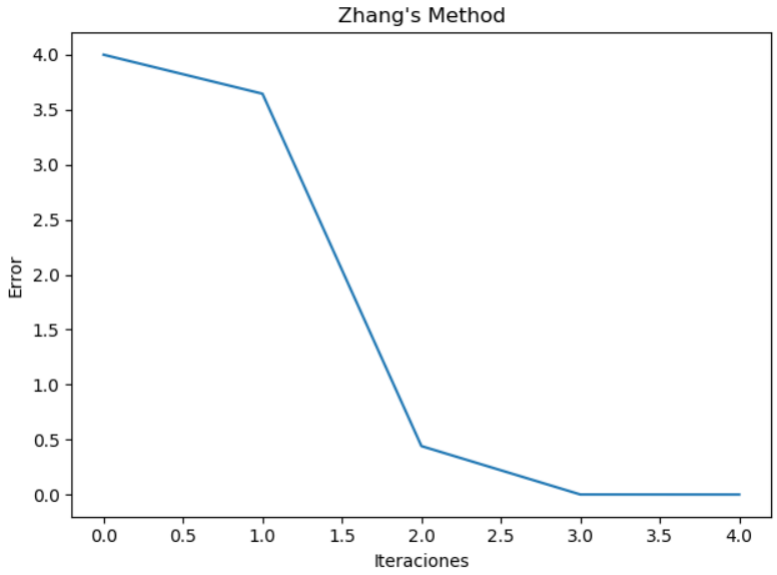

Figura 32: Ejemplo 2 de sne_fd_6

4. Bibliografía

Referencias

[1] A. Cordero, M Kansal V. Kanwar y R. Torregrosa. A stable class of improved second-derivative free Chebyshev-Halley type methods with optimal eighth order convergence.2015

[2] K. Diyashvir, R. Babajee, A Cordero, F. Soleymani, J. R. Torregrosa. On improved three-step schemes with high efficiency index and their dynamics. 2013

[3] S. Artidielloa, A. Cordero , R. Torregrosa , P. Vassilevaa. Multidimensional generalization of iterative methods for solving nonlinear problems by means of weight-function procedure.2015

[4] R. Kiran, L. Li, K. Performance of cubic convergent methods for implementing nonlinear constitutive models.2015

[5] A Cordero, J. Hueso, E. Martínez , J. R. Torregrosa. Steffensen type methods for solving nonlinear equations. 2010

[6] A. Cordero, J. R. Torregrosa. A class of Steffensen type methods with optimal order of convergence.2011

[7] A. Corderoa,F. Soleymani, R. Torregrosa and F. Haghani. A family of Kurchatov-type methods and its stability.2015

[8] A class of Steffensen type methods with optimal order of convergence