TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica

Análisis Numérico para Ingeniería

Manual SolNE Python

Mauricio Álvarez Varela - 2016116475
Jung Hwan Bak - 2016193299
José Carlos Núñez Valverde - 2016062809
Álvaro Quesada Ramírez - 2016005829

25 – 08 – 2019

Tabla de Contenidos

1. Introducción

El paquete computacional explicado en el presente documento, cumple como función principal la resolución de ecuaciones no lineales, utilizando 12 diferentes métodos matemáticos implementados en el Lenguaje de programación de Python. Es importante mencionas, que de los métodos mencionados anteriormente, la mitad hacen uso de derivadas para calcular la solución de la ecuación, y la otra mitad no requieren hacer uso de derivadas, por ende los métodos tienen orden de convergencia distintos, y algunos utilizan más iteraciones que otros para determinar la solución. Además, si el usuario lo desea, es posible visualizar una gráfica asociada las iteraciones (k) vs errores $|f(x_k)|$.

2. Instalación

A la hora de instalar el paquete de Python se deben de seguir estos pasos:

- Descargar el archivo *tarea_ani_2.zip* y extráigalo.
- Una vez extraído el archivo, ingresar a la carpeta llamada : *Python*
- Dentro de la carpeta *Python*, extraer el archivo *SolNE.zip*
- En este punto, encontrara una carpeta SolNe la cual se debe importar a su IDE de Python.
- Abrir la terminal en la ubicación actual de la carpeta SolNE o bien una vez abierto el IDE de su preferencia ubicarse en la ruta *Python* $\backslash SolNE$
- Insertar el comando 'pip install .'.

Con estos pasos ya tendrá el paquete instalado globalmente en su computadora, para accesar a los comandos se debe de importar la librería de la siguiente forma 'import SolNE' en la consola de Python.

2.1. Requisitos

Para su uso óptimo se requerirá utilizar Python 3.7 con los módulos de matplotlib que es el encargado de realizar la gráficas, y sympy utilizado para pasar de texto a simbólico.

3. Funciones Implementadas

A continuación se explica el uso de las funciones implementadas en este paquete.

3.1. No Requieren Derivar

Las funciones que no requieren uso de derivadas son las siguientes

3.1.1. sne_fd_1

Para demostrar el uso de esta función, se estará resolviendo la siguiente ecuación no lineal:

$$f(x) = x^2 - e^x - 3x + 2 \tag{1}$$

El primer paso a realizar es definir la función que se desea resolver en Python, en este caso será la función anterior. Por lo que se escribiría en la consola tal y como se muestra en la Figura 1

```
import SolNE
f =  '(x^2)-(E^x)-(3*x)+2'
```

Figura 1: Definición de la función a resolver.

Seguidamente, es necesario llamar la función con los argumentos necesarios que son la función f, un valor inicial x_0, la tolerancia tol y un parámetro graf para graficar. Es importante rescatar que la gráfica sólo se mostrará si el parámetro graf es igual a 1 o se omite. En caso de poner algún otro valor no se realizará la gráfica.

Para el ejemplo se utilizarán valores de:

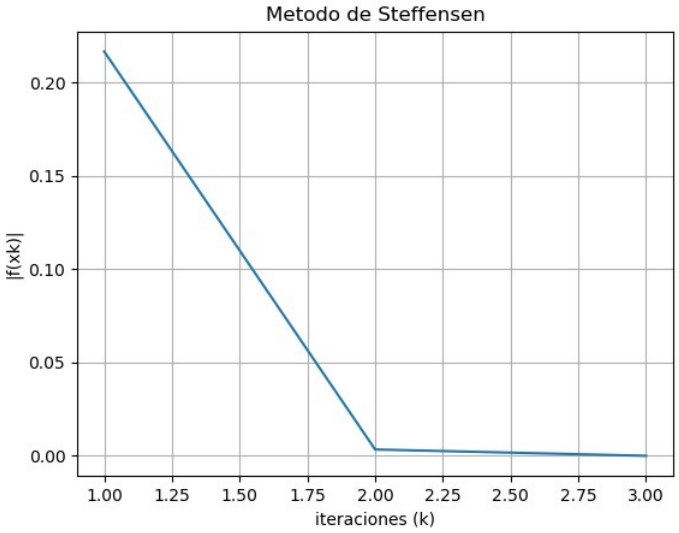
$$x_0 = 0{,}7 \tag{2}$$
$$tol = 0{,}00001 \tag{3}$$
$$graf = 1 \tag{4}$$
$$\tag{5}$$

Por lo que la llamada a la función sería tal y como se muestra en la Figura 2

```
import SolNE
f =  '(x^2)-(E^x)-(3*x)+2'
SolNE.sne_fd_1(f, 0.7, 0.00001, graf = 1)
```

Figura 2: Llamado a la función sne_fd_1

Una vez ejecutado el comando anterior, se obtendrá primero la gráfica y luego el resultado, igual al de la Figura 3. En caso de no requerir la gráfica, solo se obtendrá el resultado de la derecha de la Figura 3.




```
>>> import SolNE
>>> f =  '(x^2)-(E^x)-(3*x)+2'
>>> SolNE.sne_fd_1(f, 0.7, 0.00001, graf = 1)
xaprox: 0.2575304949085537
iter: 3
```

Figura 3: Resultado de la función sne_fd_1

Finalmente, si es necesario, se puede hacer uso del llamado de ayuda help('SolNE.sne_fd_1')

3.1.2. sne_fd_2

Para demostrar el uso de esta función, se estará resolviendo la siguiente ecuación no lineal:

$$f(x) = x^2 - e^x - 3x + 2 \tag{6}$$

El primer paso a realizar es definir la función que se desea resolver en Python, en este caso será la función anterior. Por lo que se escribiría en la consola tal y como se muestra en la Figura 4

```
import SolNE
f =  '(x^2)-(E^x)-(3*x)+2'
```

Figura 4: Definición de la función a resolver.

Seguidamente, es necesario llamar la función con los argumentos necesarios que son la función f, un valor inicial x_0, la tolerancia tol y un parámetro graf para graficar. Es importante rescatar que la gráfica sólo se mostrará si el parámetro graf es igual a 1 o se omite. En caso de poner algún otro valor no se realizará la gráfica.

Para el ejemplo se utilizarán valores de:

$$x_0 = 0{,}7 \tag{7}$$
$$tol = 0{,}00001 \tag{8}$$
$$\tag{9}$$

Nótese que para ejemplificar, en este caso se omitirá el parámetro graf, por lo que la llamada a la función sería tal y como se muestra en la Figura 5

```
import SolNE
f =  '(x^2)-(E^x)-(3*x)+2'
SolNE.sne_fd_2(f, 0.7, 0.00001)
```

Figura 5: Llamado a la función sne_fd_2

Una vez ejecutado el comando anterior, se obtendrá primero la gráfica y luego el resultado, igual al de la Figura 6.

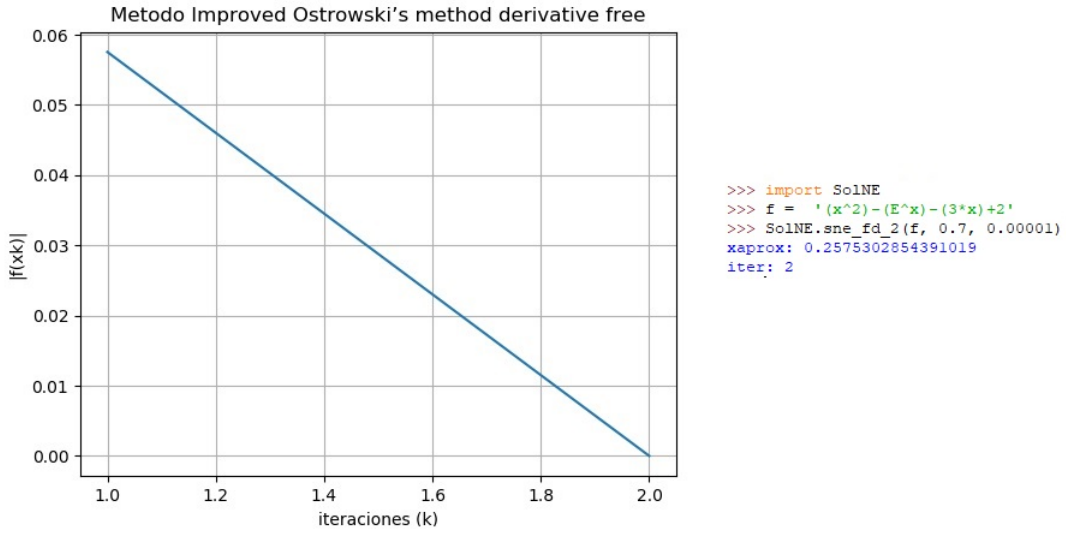


Figura 6: Resultado de la función sne_fd_2

Finalmente, si es necesario, se puede hacer uso del llamado de ayuda help('SolNE.sne_fd_2')

3.1.3. sne_fd_3

Para demostrar el uso de esta función, se estará resolviendo la siguiente ecuación no lineal:

$$f(x) = x^2 - e^x - 3x + 2 \tag{10}$$

El primer paso a realizar es definir la función que se desea resolver en Python, en este caso será la función anterior. Por lo que se escribiría en la consola tal y como se muestra en la Figura 22

```
import SolNE
f =  '(x^2)-(E^x)-(3*x)+2'
```

Figura 7: Definición de la función a resolver.

Seguidamente, es necesario llamar la función con los argumentos necesarios que son la función f, un valor inicial x_0, la tolerancia tol y un parámetro graf para graficar. Es importante rescatar que la gráfica sólo se mostrará si el parámetro graf es igual a 1 o se omite. En caso de poner algún otro valor no se realizará la gráfica.

Para el ejemplo se utilizarán valores de:

$$x_0 = 0{,}7 \tag{11}$$
$$tol = 0{,}00001 \tag{12}$$
$$graf = 0 \tag{13}$$

Nótese que para ejemplificar, en este caso se define graf igual a 0, por lo que la llamada a la función sería tal y como se muestra en la Figura 8.

```
import SolNE
f =  '(x^2)-(E^x)-(3*x)+2'
SolNE.sne_fd_3(f, 0.7, 0.00001, graf = 0)
```

Figura 8: Llamado a la función sne_fd_3

Una vez ejecutado el comando anterior, se retorna solamente el resultado, ya que anteriormente se definió graf igual a 0. Ejemplo en la Figura 9.

```
>>> import SolNE
>>> f =  '(x^2)-(E^x)-(3*x)+2'
>>> SolNE.sne_fd_3(f, 0.7, 0.00001, graf = 0)
Sin grafica
xaprox: 0.25753034776459727
iter: 2
```

Figura 9: Resultado de la función sne_fd_3

Finalmente, si es necesario, se puede hacer uso del llamado de ayuda help('SolNE.sne_fd_3')

3.1.4. sne_fd_4

Para demostrar el uso de esta función, se estará resolviendo la siguiente ecuación no lineal:

$$f(x) = x^2 - 18 \tag{14}$$

El primer paso a realizar es definir la función que se desea resolver en Python, en este caso será la función anterior. Por lo que se escribiría en la consola tal y como se muestra en la Figura 10

```
import SolNE
f = "x^2 - 18"
```

Figura 10: Definición de la función a resolver.

Seguidamente, es necesario llamar la función con los argumentos necesarios que son la función f, un valor inicial x_0, la tolerancia tol y un parámetro graf para graficar. Es importante rescatar que la gráfica sólo se mostrará si el parámetro graf es igual a 1 o se omite. En caso de poner algún otro valor no se realizará la gráfica.

Para el ejemplo se utilizarán valores de:

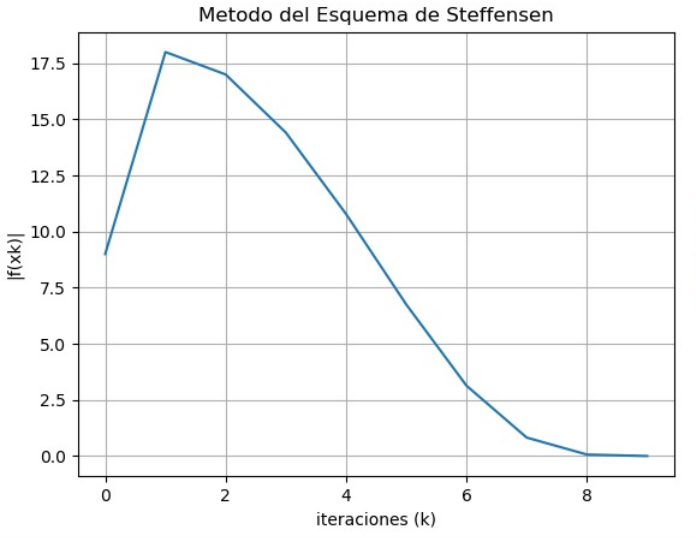
$$x_0 = 3 \tag{15}$$
$$tol = 0{,}00001 \tag{16}$$
$$graf = 1 \tag{17}$$

Nótese que para ejemplificar, en este caso se define graf igual a 1, por lo que la llamada a la función sería tal y como se muestra en la Figura 11.

```
import SolNE
f = "x**2 - 18"
SolNE.sne_fd_4("x**2 - 18", 3, 0.00001 ,1)
```

Figura 11: Llamado a la función sne_fd_4

Una vez ejecutado el comando anterior, se obtendrá primero la gráfica y luego el resultado, igual al de la Figura 12.




```
>>> import SolNE
>>> f = "x**2 - 18"
>>> SolNE.sne_fd_4("x**2 - 18", 3, 0.00001 ,1)
xaprox: -4.242640684617487
iter: 10
```

Figura 12: Resultado de la función sne_fd_4

Finalmente, si es necesario, se puede hacer uso del llamado de ayuda help('SolNE.sne_fd_4')

3.1.5. sne_fd_5

Para demostrar el uso de esta función, se estará resolviendo la siguiente ecuación no lineal:

$$f(x) = (1 - x)^2 \tag{18}$$

El primer paso a realizar es definir la función que se desea resolver en Python, en este caso será la función anterior. Por lo que se escribiría en la consola tal y como se muestra en la Figura 13

```
import SolNE
f = "(1-x)^2"
```

Figura 13: Definición de la función a resolver.

Seguidamente, es necesario llamar la función con los argumentos necesarios que son la función f, un valor inicial x_0, la tolerancia tol y un parámetro graf para graficar. Es importante rescatar que la gráfica sólo se mostrará si el parámetro graf es igual a 1 o se omite. En caso de poner algún otro valor no se realizará la gráfica.

Para el ejemplo se utilizarán valores de:

$$x_0 = 3 \tag{19}$$
$$tol = 0{,}00001 \tag{20}$$
$$graf = 0 \tag{21}$$

Nótese que para ejemplificar, en este caso se define graf igual a 0, por lo que la llamada a la función sería tal y como se muestra en la Figura 14.

```
import SolNE
f = "(1-x)^2"
SolNE.sne_fd_5(f, 3, 0.00001, 0)
```

Figura 14: Llamado a la función sne_fd_5

Una vez ejecutado el comando anterior, se obtiene solamente el resultado, ya que anteriormente se definió graf igual a 0. Ejemplo en la Figura 15.

```
>>> import SolNE
>>> f = "(1-x)^2"
>>> SolNE.sne_fd_5(f, 3, 0.00001, 0)
Sin grafica
xaprox: 1.0016260537943023
iter: 15
```

Figura 15: Resultado de la función sne_fd_5

Finalmente, si es necesario, se puede hacer uso del llamado de ayuda help('SolNE.sne_fd_5')

3.1.6. sne_fd_6

Para demostrar el uso de esta función, se estará resolviendo la siguiente ecuación no lineal:

$$f(x) = x^2 - 18 \tag{22}$$

El primer paso a realizar es definir la función que se desea resolver en Python, en este caso será la función anterior. Por lo que se escribiría en la consola tal y como se muestra en la Figura 16

```
import SolNE
f = "x^2 - 18"
```

Figura 16: Definición de la función a resolver.

Seguidamente, es necesario llamar la función con los argumentos necesarios que son la función f, un valor inicial x_0, la tolerancia tol y un parámetro graf para graficar. Es importante rescatar que la gráfica sólo se mostrará si el parámetro graf es igual a 1 o se omite. En caso de poner algún otro valor no se realizará la gráfica.

Para el ejemplo se utilizarán valores de:

$$x_0 = 3 \tag{23}$$

$$tol = 0{,}00001 \tag{24}$$

$$graf = 0 \tag{25}$$

Nótese que para ejemplificar, en este caso se define graf igual a 0, por lo que la llamada a la función sería tal y como se muestra en la Figura 17.

```
import SolNE
f = "x^2 - 18"
SolNE.sne_fd_6(f, 3, 0.00001, 0)
```

Figura 17: Llamado a la función sne_fd_6

Una vez ejecutado el comando anterior, se obtiene solamente el resultado, ya que anteriormente se definió graf igual a 0. Ejemplo en la Figura 18.

```
>>> SolNE.sne_fd_6(f, 3, 0.00001, 0)
Sin grafica
xaprox: 4.242640687112676
iter: 6
```

Figura 18: Resultado de la función sne_fd_6

Finalmente, para ejecutar el comando de ayuda, se debe llamar a Finalmente, si es necesario, se puede hacer uso del llamado de ayuda help('SolNE.sne_fd_6')

3.2. Requieren Derivar

3.2.1. sne_ud_1

Para demostrar el uso de esta función, se estará resolviendo la siguiente ecuación no lineal:

$$f(x) = x^2 - e^x - 3x + 2 \tag{26}$$

El paso número uno para demostrar la funcionalidad del método es definir la función a resolver. En este caso será la función mostrada anteriormente. Para esto, se debe escribir en la consola lo que se muestra en la Figura 22

```
import SolNE
f =  '(x^2)-(E^x)-(3*x)+2'
```

Figura 19: Definición de la función a resolver.

A continuación, se debe de hacer el llamado de la función con los argumentos necesarios. Estos son la función f previamente definida, un valor inicial x_0, una tolerancia tol y el parámetro de graf para determinar si se muestra el gráfico o no. Cabe destacar de que en caso de que el parámetro de graf ingresado es 0, no se mostrará la gráfica.

Para el ejemplo se utilizarán valores de:

$$x_0 = 0{,}7 \tag{27}$$
$$tol = 0{,}00001 \tag{28}$$
$$\tag{29}$$

Nótese que para ejemplificar, en este caso se omite un valor de entrada en el parámetro graf, por lo que la llamada a la función sería tal y como se muestra en la Figura 20.

```
import SolNE

f = '(x^2)-(E^x)-(3*x)+2'

SolNE.sne_ud_1(f, 0.7, 0.00001)
```

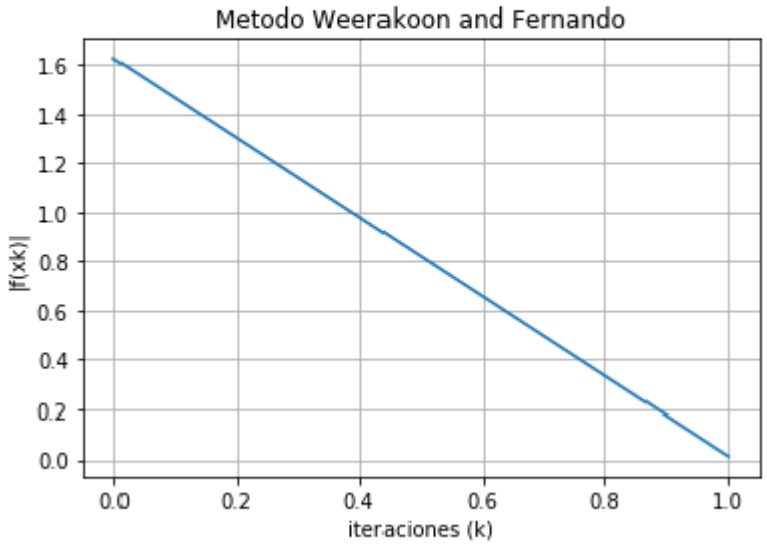
Figura 20: Llamado a la función sne_ud_1

Una vez ejecutado el comando anterior, se mostrará primeramente la gráfica con su respectivo título. Esto debido a que por lógica de la función, al no ingresar parámetro de graf, se mostrará la gráfica resultante por default. Además, posterior a la gráfica se mostrará el valor de x aproximado junto con el número de iteraciones realizadas, tal y como se muestra en la Imagen 21.

```
In [1]: import SolNE

In [2]: f = '(x^2)-(E^x)-(3*x)+2'

In [3]: SolNE.sne_ud_1(f, 0.7, 0.00001)
```




```
xaprox 0.2575302869809505
iter: 2
```

Figura 21: Resultado de la función sne_ud_1

Finalmente, si es necesario, se puede hacer uso del llamado de ayuda help('SolNE.sne_ud_1')

3.2.2. sne_ud_2

Este método es conocido como el método de Frontini and Sormani. Para demostrar el uso de este, se propone la siguiente ecuación no lineal a resolver:

$$f(x) = x^2 - e^x - 3x + 2 \tag{30}$$

Primeramente, se debe definir la función a utilizar, en este caso será la expresada anteriormente, asignada a una variable f. Para esto, se debe escribir en la consola lo que se muestra en la Figura 22

```
import SolNE
f =  '(x^2)-(E^x)-(3*x)+2'
```

Figura 22: Definición de la función a resolver.

Posteriormente, se debe de hacer el llamado de la función con los argumentos necesarios. Estos son: la función f, el valor inicial x_0, la tolerancia tol y el parámetro de graf para determinar si se muestra el gráfico o no, dependiendo del valor que reciba este último.

Para el ejemplo se utilizarán valores de:

$$x_0 = 0{,}7 \tag{31}$$
$$tol = 0{,}00001 \tag{32}$$
$$graf = 1 \tag{33}$$

De esta manera el llamado de la función sería el siguiente 23.

```
import SolNE

f = '(x^2)-(E^x)-(3*x)+2'

SolNE.sne_ud_2(f, 0.7, 0.00001,1)
```

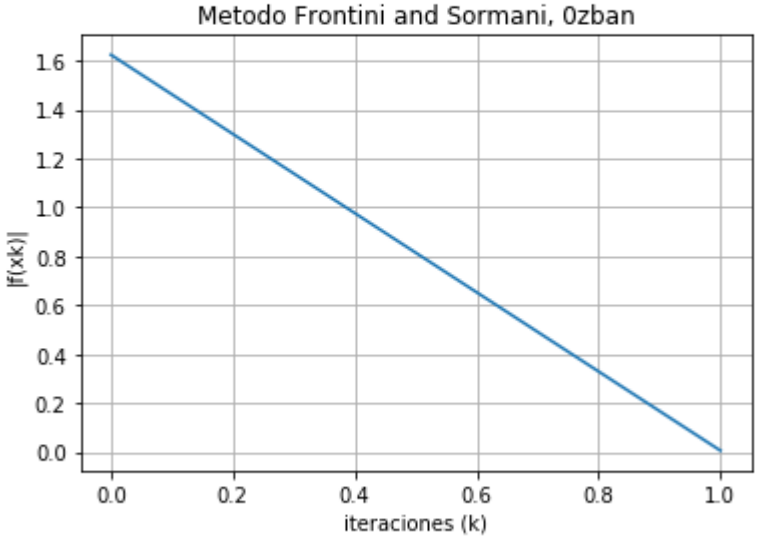
Figura 23: Llamado a la función sne_ud_2

Una vez ejecutado el comando anterior, se mostrará primeramente la gráfica con su respectivo título. Además, posterior a la gráfica se mostrará el valor de x aproximado junto con el número de iteraciones realizadas, tal y como se muestra en la Imagen 24.

```
In [1]: import SolNE

In [2]: f = '(x^2)-(E^x)-(3*x)+2'

In [3]: SolNE.sne_ud_2(f, 0.7, 0.00001,1)
```




```
xaprox 0.2575302854561175
iter: 2
```

Figura 24: Resultado de la función sne_ud_2

Finalmente, si es necesario, se puede hacer uso del llamado de ayuda help('SolNE.sne_ud_2')

3.2.3. sne_ud_3

Para demostrar el uso de esta función, se estará resolviendo la siguiente ecuación no lineal:

$$x^3 + 6x^2 - 18 \tag{34}$$

Primeramente, se debe definir la función a utilizar, en este caso será la expresada anteriormente, asignada a una variable f. Para esto, se debe escribir en la consola lo que se muestra en la Figura 34

```
import SolNE

f = "x^3+6*x^2-18"
```

Figura 25: Definición de la función a resolver.

Posteriormente, se debe de hacer el llamado del método a utilizar, en este caso se trata del método llamado Halley. Este debe contar con los argumentos necesarios, los cuales son: la función f, el valor inicial x_0, la tolerancia tol y el parámetro de graf para determinar si se muestra el gráfico o no, dependiendo del valor que reciba este último.

Para el ejemplo se utilizarán valores de:

$$x_0 = 0{,}7 \tag{35}$$
$$tol = 0{,}00001 \tag{36}$$
$$graf = 1 \tag{37}$$

De esta manera el llamado de la función sería el siguiente 26.

```
import SolNE

f = "x^3+6*x^2-18"

SolNE.sne_ud_3(f, 0.7, 0.00001,1)
```

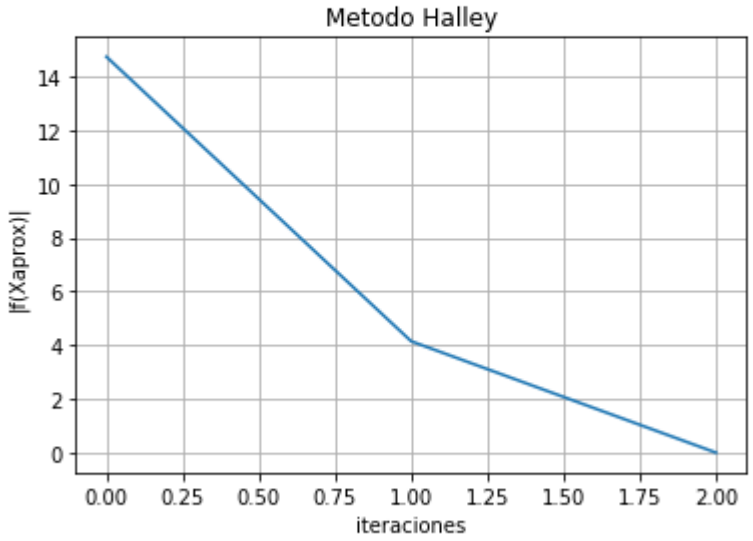
Figura 26: Llamado a la función sne_ud_3

Una vez ejecutado el comando anterior, se mostrará primeramente la gráfica con su respectivo título. Además, posterior a la gráfica se mostrará el valor de x aproximado junto con el número de iteraciones realizadas, tal y como se muestra en la Imagen 27.

```
In [1]: import SolNE

In [2]: f = "x^3+6*x^2-18"

In [3]: SolNE.sne_ud_3(f, 0.7, 0.00001,1)
```




```
xaprox 1.544606815244243
iter: 3
```

Figura 27: Resultado de la función sne_ud_3

Finalmente, si es necesario, se puede hacer uso del llamado de ayuda help('SolNE.sne_ud_3')

3.2.4. sne_ud_4

Para demostrar el uso de esta función, se estará resolviendo la siguiente ecuación no lineal:

$$x^3 + 6x^2 - 18 \tag{38}$$

Para empezar, se debe definir la función a utilizar, en este caso será la expresada anteriormente, asignada a una variable f. Para esto, se debe escribir en la consola lo que se muestra en la Figura 34

```
import SolNE

f = "x^3+6*x^2-18"
```

Figura 28: Definición de la función a resolver.

Posteriormente, se debe de hacer el llamado del método a utilizar, en este caso se trata del método llamado Darvishi and Barati. Este debe contar con los argumentos necesarios, los cuales son: la función f, el valor inicial x_0, la tolerancia tol y el parámetro de graf para determinar si se muestra el gráfico o no, dependiendo del valor que reciba este último.

Para el ejemplo se utilizarán valores de:

$$x_0 = 0{,}7 \tag{39}$$
$$tol = 0{,}00001 \tag{40}$$
$$graf = 1 \tag{41}$$

De esta manera el llamado de la función sería el siguiente 29.

```
import SolNE

f = "x^3+6*x^2-18"

SolNE.sne_ud_4(f, 0.7, 0.00001,1)
```

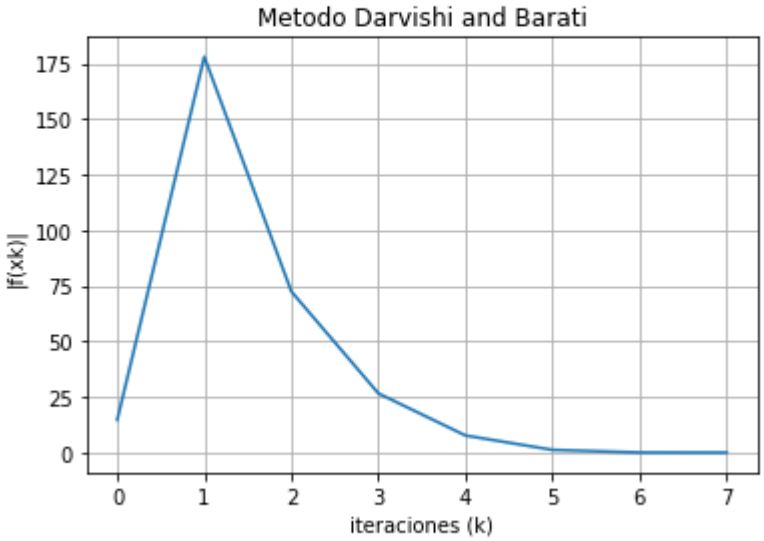
Figura 29: Llamado a la función sne_ud_4

Una vez ejecutado el comando anterior, se mostrará la gráfica resultante al comportamiento del método, junto con su respectivo título. Además, posterior a la gráfica se mostrará el valor de x aproximado junto con el número de iteraciones realizadas, tal y como se muestra en la Imagen 30.

```
In [1]: import SolNE

In [2]: f = "x^3+6*x^2-18"

In [3]: SolNE.sne_ud_4(f, 0.7, 0.00001,1)
```




```
xaprox 1.5446068153208634
iter: 8
```

Figura 30: Resultado de la función sne_ud_4

Finalmente, si es necesario, se puede hacer uso del llamado de ayuda help('SolNE.sne_ud_4')

3.2.5. sne_ud_5

Para demostrar el uso y funcionamiento de este método, se estará resolviendo la siguiente ecuación no lineal:

$$x^3 + 6x^2 - 18 \tag{42}$$

Para empezar, se debe definir la función a utilizar, en este caso será la expresada anteriormente, asignada a una variable f. Para esto, se debe escribir en la consola lo que se muestra en la Figura 34

```
import SolNE

f = "x^3+6*x^2-18"
```

Figura 31: Definición de la función a resolver.

Posteriormente, se debe de hacer el llamado del método a utilizar, en este caso se trata del método llamado Kou et al. Este debe contar con los argumentos necesarios para su desarrollo, estos argumentos son: la función f previamente definida, el valor inicial x_0, la tolerancia tol y el parámetro de graf para determinar si se muestra el gráfico o no, dependiendo del valor que reciba este último.

Para el ejemplo se utilizarán valores de:

$$x_0 = 3 \tag{43}$$
$$tol = 0{,}00001 \tag{44}$$
$$graf = 1 \tag{45}$$

De esta manera el llamado de la función sería el siguiente 32.

```
import SolNE

f = "x^3+6*x^2-18"

SolNE.sne_ud_5(f, 3, 0.00001,1)
```

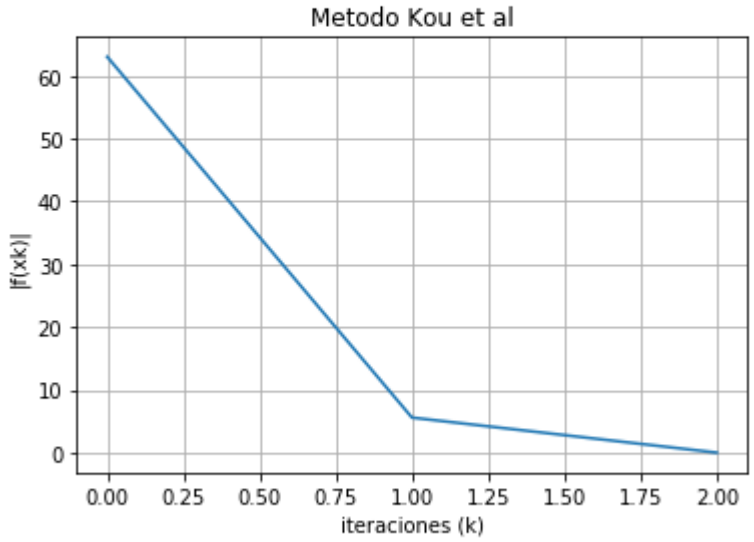
Figura 32: Llamado a la función sne_ud_5

Una vez ejecutado el comando anterior, se mostrará la gráfica resultante al comportamiento del método, junto con su respectivo título. Además, posterior a la gráfica se mostrará el valor de x aproximado junto con el número de iteraciones realizadas, tal y como se muestra en la Imagen 33.

```
In [1]: import SolNE

In [2]: f = "x^3+6*x^2-18"

In [3]: SolNE.sne_ud_5(f, 3, 0.00001,1)
```




```
xaprox 1.5446068165418632
iter: 3
```

Figura 33: Resultado de la función sne_ud_5

Finalmente, si es necesario, se puede hacer uso del llamado de ayuda help('SolNE.sne_ud_5')

3.2.6. sne_ud_6

Para la demostración del uso y funcionamiento de este método, se estará resolviendo la siguiente ecuación no lineal:

$$x^3 + 6x^2 - 18 \tag{46}$$

Para empezar, se debe definir la función a utilizar, en este caso será la expresada anteriormente, asignada a una variable f. Para esto, se debe escribir en la consola lo que se muestra en la Figura 34

```
import SolNE

f = "x^3+6*x^2-18"
```

Figura 34: Definición de la función a resolver.

A continuación, se debe de hacer el llamado del método a utilizar, en este caso se trata del método llamado Chun and Kim. Este llamado debe contar con los siguiente argumentos: la función f, el valor inicial x_0, la tolerancia tol y el parámetro de graf para determinar si se muestra el gráfico o no, dependiendo del valor que reciba este último.

Para el ejemplo se utilizarán valores de:

$$x_0 = 3 \tag{47}$$
$$tol = 0{,}00001 \tag{48}$$
$$graf = 1 \tag{49}$$

De esta manera el llamado de la función sería el siguiente 35.

```
import SolNE

f = "x^3+6*x^2-18"

SolNE.sne_ud_6(f, 3, 0.00001,1)
```

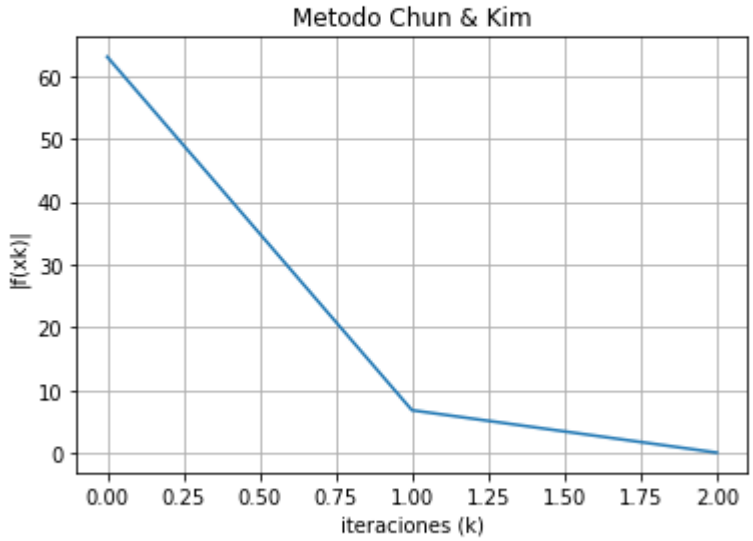
Figura 35: Llamado a la función sne_ud_6

Una vez ejecutado el comando anterior, se mostrará la gráfica resultante al comportamiento del método, junto con su respectivo título. Además, posterior a la gráfica se mostrará el valor de x aproximado junto con el número de iteraciones realizadas, tal y como se muestra en la Imagen 33.

```
In [1]: import SolNE

In [2]: f = "x^3+6*x^2-18"

In [3]: SolNE.sne_ud_6(f, 3, 0.00001,1)
```




```
xaprox 1.5446068319674096
iter: 3
```

Figura 36: Resultado de la función sne_ud_6

Finalmente, si es necesario, se puede hacer uso del llamado de ayuda help('SolNE.sne_ud_6')

Bibliografía

[1] Cordero, A. & Martinez E. (2012). Steffensen type methods for solving nonlinear equations. Journal of Computational and Applied Mathematics. Recuperado 25 agosto, 2019.

[2] Cordero, A. & Torregrosa. (2011). A class of Steffensen type methods with optimal order of convergence. Recuperado 25 agosto, 2019.

[3] Cordero, A & Torregrosa, J. & Soleymani F (2016). A family of Kurchatov-type methods and its stability. Recuperado 25 agosto, 2019.

[4] Kiran, R & Li, L. & Khandelwal K (2015). Performance of cubic convergent methods for implementing nonlinear constitutive models. Recuperado 25 agosto, 2019.