TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica

Análisis Numérico para Ingeniería

Manual SolNE Octave

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 GNU Octave. 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 Octave 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 : *Octave*
- Dentro de la carpeta *Octave*, encontrar el archivo $SolNE.tar.gz$
- En este punto, debe abrir Octave, y en el buscador de archivos, buscar el archivo $SolNE.tar.gz$ en la dirección anterior.
- Abrir la terminal en la ubicación actual archivo $SolNE.tar.gz$ o bien una vez abierto el IDE de su preferencia ubicarse en la ruta $GNUOctave \backslash SolNE$ e insertar el comando "pkg install SolNE.tar.gz"
- Seguidamente la terminar, cargar el paquete, utilizando el comando "pkg load solne". Es importante utilizar minúsculas en esta sección, de lo contrario habrá un error.

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 Octave en su versión 5.1.

3. Funciones Implementadas

3.1. No Requieren Derivar

3.1.1. sne_fd_1

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

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

Para ejecutar el ejemplo se escribe la función f y el comando para ejecutar el método de la siguiente forma:

```
f = 'x^2 - E^x - 3*x + 2';
[x0, itr] = sne_fd_1 (f, 0.7, 0.000001,1)
```

Figura 1: Definición y ejecución de la función a resolver.

3.1.2. sne_fd_2

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

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

Para ejecutar el ejemplo se escribe la función f y el comando para ejecutar el método de la siguiente forma:

```
>> f = 'x^2 - E^x - 3*x + 2';
>> [x0, itr] = sne_fd_2 (f, 0.7, 0.000001,1)
```

Figura 2: Definición y ejecución de la función a resolver.

3.1.3. sne_fd_3

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

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

Para ejecutar el ejemplo se escribe la función f y el comando para ejecutar el método de la siguiente forma:

```
>> f = 'x^2 - E^x - 3*x + 2';
>> [x0, itr] = sne_fd_3 (f, 0.7, 0.000001,1)
```

Figura 3: Definición y ejecución de la función a resolver.

3.1.4. sne_fd_4

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

$$f(x) = x^2 - 18 \tag{4}$$

Para ejecutar el ejemplo se escribe la función f y el comando para ejecutar el método de la siguiente forma:

```
>> f = 'x^2 -18'
f = x^2 -18
>> [x0, itr] = sne_fd_4 (f, 3, 0.000001,1)
```

Figura 4: Definición y ejecución de la función a resolver.

3.1.5. sne_fd_5

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

$$f(x) = (1 - x)^2 \tag{5}$$

Para ejecutar el ejemplo se escribe la función f y el comando para ejecutar el método de la siguiente forma:

```
>> f = '(1-x)^2'
f = (1-x)^2
>> [x0, itr] = sne_fd_5 (f, 3, 0.000001,1)
```

Figura 5: Definición y ejecución de la función a resolver.

3.1.6. sne_fd_6

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

$$f(x) = x^2 - 18 \tag{6}$$

Para ejecutar el ejemplo se escribe la función f y el comando para ejecutar el método de la siguiente forma:

```
>> f = 'x^2 - 18'
f = x^2 - 18
>> [x0, itr] = sne_fd_6 (f, 3, 0.000001,1)
```

Figura 6: Definición y ejecución de la función a resolver.

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:

$$f(x) = x^2 - 2^x - 3x + 2 \tag{7}$$

Para ejecutar el ejemplo se escribe la función f y el comando para ejecutar el método de la siguiente forma:

```
>> f = ' x^2 - 2^x - 3*x + 2'
f =  x^2 - 2^x - 3*x + 2
>> [xaprox, iter] = sne_ud_1(f, 1, 0.0001, 1)
```

Figura 7: Definición y ejecución de la función a resolver.

3.2.2. sne_ud_2

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

$$f(x) = x^3 - 6 * (x^2) - 18 \tag{8}$$

Para ejecutar el ejemplo se escribe la función f y el comando para ejecutar el método de la siguiente forma:

```
[xaprox, iter] = sne_ud_2("x^3 + 6*x^2 - 18",3,0.0001, 1)
```

Figura 8: Definición y ejecución de la función a resolver.

3.2.3. sne_ud_3

Para la demostración del método de Halley, se utilizará la ecuación no lineal:

$$f(x) = x^3 + 6x^2 - 18 \tag{9}$$

Para utilizar esta función se debe ingresar:

```
[Xaprox, iter] = sne_ud_3(función, valor inicial, tolerancia, gráfica)
```

donde la función debe ser un string, el valor inicial debe ser un entero y muestra donde va comenzar las iteraciones, la tolerancia debe ser un entero y define el margen de error entre el valor real y el aproximado. Si desea que se muestre una gráfica, el campo de gráfica debera tener el valor de 1.

Cuando se corre el comando mencionado previamente en Octave, aparecerá primero una pestaña con la gráfica de error, como se muestra en la *Figura 9*. En la consola aparecerá la aproximación de la raíz junsto a la cantidad de iteraciones que se realizó, como se muestra en la *Figura 10*

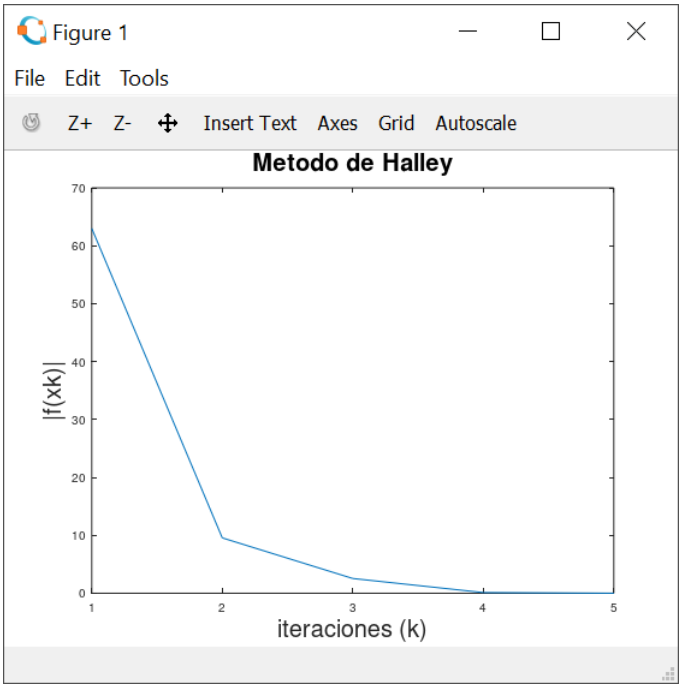

Figura 9: Gráfica del Metodo de Halley

```
>> [xaprox, iter] = sne_ud_3("x**3 + 6*x**2 - 18",3,0.0001,1)
xaprox =  1.5446
iter =  5
```

Figura 10: Resultado del método de Halley

3.2.4. sne_ud_4

Para la demostración del método de Darvishi y Barati, se utilizará la ecuación no lineal:

$$f(x) = x^3 + 6x^2 - 18 \tag{10}$$

Para utilizar esta función se debe ingresar:

```
[Xaprox, iter] = sne_ud_4(función, valor inicial, tolerancia, gráfica)
```

donde la función debe ser un string, el valor inicial debe ser un entero y muestra donde va comenzar las iteraciones, la tolerancia debe ser un entero y define el margen de error entre el valor real y el aproximado. Si desea que se muestre una gráfica, el campo de gráfica debera tener el valor de 1.

Cuando se corre el comando mencionado previamente en Octave, aparecerá primero una pestaña con la gráfica de error, como se muestra en la *Figura 11*. En la consola aparecerá la aproximación de la raíz junsto a la cantidad de iteraciones que se realizó, como se muestra en la *Figura 12*

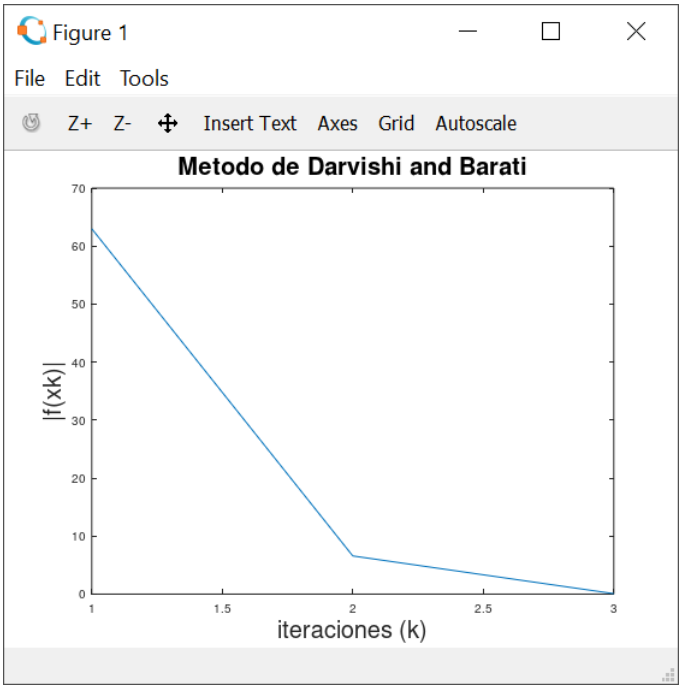

Figura 11: Gráfica del Metodo de Darvishi y Barati

```
>> [xaprox, iter] = sne_ud_4("x^3 + 6*x^2 - 18",3,0.0001, 1)
x1 =  1.7778
x1 =  1.5477
x1 =  1.5446
xaprox =  1.5446
iter =  3
```

Figura 12: Resultado del método de Darvishi y Barati

3.2.5. sne_ud_5

Para la demostración del método de Chun y Kim, se utilizará la ecuación no lineal:

$$f(x) = x^3 + 6x^2 - 18 \tag{11}$$

Para utilizar esta función se debe ingresar:

```
[Xaprox, iter] = sne_ud_5(función, valor inicial, tolerancia, gráfica)
```

donde la función debe ser un string, el valor inicial debe ser un entero y muestra donde va comenzar las iteraciones, la tolerancia debe ser un entero y define el margen de error entre el valor real y el aproximado. Si desea que se muestre una gráfica, el campo de gráfica debera tener el valor de 1.

Cuando se corre el comando mencionado previamente en Octave, aparecerá primero una pestaña con la gráfica de error, como se muestra en la *Figura 13*. En la consola aparecerá la aproximación de la raíz junsto a la cantidad de iteraciones que se realizó, como se muestra en la *Figura 14*

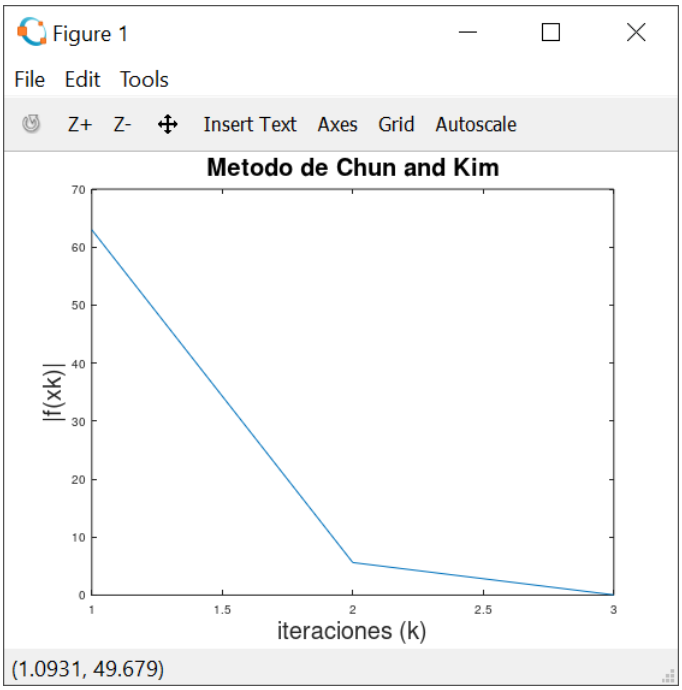

Figura 13: Gráfica del Metodo de Chun y Kim

```
>> [xaprox, iter] = sne_ud_5("x^3 + 6*x^2 - 18",3,0.0001, 1)
x1 =  1.7460
x1 =  1.5463
x1 =  1.5446
xaprox =  1.5446
iter =  3
```

Figura 14: Resultado del método de Chun y Kim

3.2.6. sne_ud_6

Para la demostración del método de Chun y Kim, se utilizará la ecuación no lineal:

$$f(x) = x^3 + 6x^2 - 18 \tag{12}$$

Para utilizar esta función se debe ingresar:

```
[Xaprox, iter] = sne_ud_3(función, valor inicial, tolerancia, gráfica)
```

donde la función debe ser un string, el valor inicial debe ser un entero y muestra donde va comenzar las iteraciones, la tolerancia debe ser un entero y define el margen de error entre el valor real y el aproximado. Si desea que se muestre una gráfica, el campo de gráfica debera tener el valor de 1.

Cuando se corre el comando mencionado previamente en Octave, aparecerá primero una pestaña con la gráfica de error, como se muestra en la *Figura 15*. En la consola aparecerá la aproximación de la raíz junsto a la cantidad de iteraciones que se realizó, como se muestra en la *Figura 16*

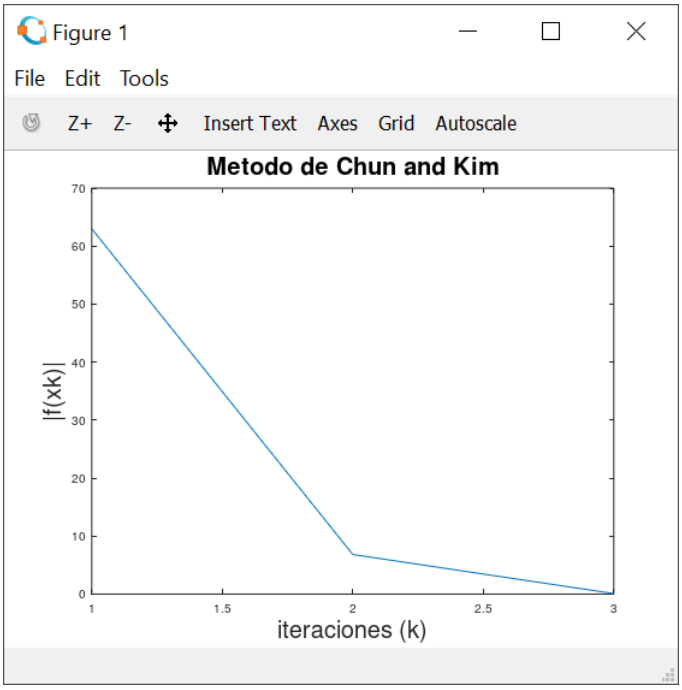

Figura 15: Gráfica del Metodo de Chun y Kim

```
>> [xaprox, iter] = sne_ud_6("x^3 + 6*x^2 - 18",3,0.0001, 1)
xaprox =  1.5446
iter =  3
```

Figura 16: Resultado del método de Chun y Kim

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.