UNIVERSIDAD DE SANTIAGO DE CHILE

FACULTAD DE INGENIERÍA

Departamento de Informática

Modelación y Simulación

Laboratorio 1

Richard Torti

Max Chacon

Profesor: Gonzalo Acuña

Ayudante: Francisco Muñoz

Santiago – Chile

2018

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

CAPÍTULO 1. INTRODUCCIÓN

El presente laboratorio tiene como objetivo general introducir el trabajo en el software MATLAB, con el fin de aprender las bases del lenguaje para realizar los futuros trabajos de modelación de sistemas. Como objetivos específicos se tienen la capacidad de graficar funciones con las consideraciones indicadas como colores y formas del gráfico, graficar en escala logarítmica, implementar el algoritmo de Newton-Rapson, con un polinomio, error, valor inicial e iteraciones como valores de entrada. El desarrollo de este trabajo se encuentra dividido en dos partes, una para cada sección indicada en el enunciado, otras secciones del informe son conjuntas para ambas partes como el marco teórico y el manual de usuario.

El presente informe cuenta con un marco teórico, en el cual serán explicados los conceptos necesarios para la correcta comprensión del desarrollo de la experiencia, posteriormente se procederá con el desarrollo de ambas partes del laboratorio,para estas serán explicadas las funciones creadas y los distintos algoritmos empleados en la resolución del los problemas planteados; indicando como fue el proceso para encontrar esta solución y detallando las variables de entrada y salida del algoritmo. Posteriormente el informe presenta un manual de usuario para las dos partes, específicamente para la segunda parte, explicando con ejemplos claros como utilizar la función para obtener los resultados con las entradas deseadas y las referencias utilizadas para la realización de las soluciones y el informe. Finalmente se presentan las conclusiones correspondientes al trabajo realizado, en donde se realiza una reflexión sobre como fue abordado el trabajo, si los objetivos generales y específicos fueron cumplidos a totalidad, las dificultades que puedan haber sido encontradas en el desarrollo de este y lecciones aprendidas para futuros laboratorios.

CAPÍTULO 2. MÁRCO TEÓRICO

2.1 MATLAB

Matlab es una herramienta tecnológica, que cuenta con un entorno integrado de desarrollo para su lenguaje de programación M. Esta herramienta fue desarrollada por Cleve Moler y lanzada en 1984. La herramienta se clasifica como un software matemático, ya que su principal uso es científico para resolver grandes modelos matemáticos o utilizar técnicas avanzadas de matemática, a la par se han desarrollado muchos complementos para hacer mas completo el software, con las cuales se pueden graficar las funciones o modelos ingresados a la herramienta, estos complementos han ayudado a mejorar a través del tiempo, manteniéndose siempre actualizado para los usuarios que deseen utilizar sus herramientas. En el presente trabajo utilizaremos el software para representar funciones y graficarlas de manera especifica, también en la segunda parte se debe implementar un algoritmo matemático conocido como Newton-Rapson, con el fin de modelas este algoritmo e indicar de manera correcta las salidas, acorde a las entradas ingresadas. (MathWorks, 2006)

2.2 NEWTON RAPSON

Este es un método eficiente para encontrar los ceros o las raíces de una función real. En este método no está asegurada su convergencia, para poder alcanzar la convergencia en este método el valor que se a elegir debe ser cercano a la raíz, de esta manera asegurando su correcto funcionamiento.

El método N-R es muy rápido y eficiente ya que cuenta con una convergencia del tipo cuadrática, sin embargo la convergencia de este método depende de la forma que tome la función; en las proximidades del punto de iteración, es decir si la función tiene muchos o muy grandes puntos de inflexión cerca de la raíz buscada, es posible que no exista la convergencia en

el método. La formula ocupada para las diferentes iteraciones es la siguiente:

$$X_{n+1} = X_n - \frac{f(X_n)}{f'(X_n)} \tag{2.1}$$

De esta manera se va iterando por cada numero, utilizando la formula de la ecuación hasta llegar a la precisión deseada.

2.3 ESCALA ARITMÉTICA

Esta es la escala normal utilizada para graficar o para representar puntos en una recta, en esta escala si existe una diferencia es por unidades, es decir de 10 a 20 se representa gráficamente igual que de 20 a 30, todas son variaciones absolutas, se miden por la misma unidad y los niveles están separados a la misma distancia uno de otro, las variación tiene un comportamiento lineal. esto es como se realizan la mayoría de los gráficos pero no siempre es la mejor opción por que podemos tener dispersión de datos muy variante, por ejemplo muchos datos agrupados cerca de un numero bajo y otros pocos datos en un numero muy lejano, lo que dificultaría la representación es una escala de una dimensión.(Cruz, 2015)

2.4 ESCALA LOGARÍTMICA

Esta es una escala creada para representar datos cuyos valores son difíciles de apreciar en la escala aritmética, como fue explicado anteriormente existen casos en que la escala aritmética no entrega una buen modelo de los datos, por lo que es necesario cambiar la escala en la cual graficamos, para apreciar de mejor manera las diferencias o similitudes. Con este problema se creo la escala logarítmica, la cual utiliza variaciones porcentuales o relativas, teniendo una variación logarítmica y no lineal, para esta es necesario calcular el logaritmo en base 10 de cada dato, comenzando la representación desde 1, de esta manera se calculan los legítimos de los siguientes numero y a esa distancia se colocan los valores, recordar que las etiquetas seguirán

siendo las mismas y no cambian por lo logaritmos. Esta escala nos permite observar todos los cambios porcentuales del mismo valor a la misma distancia, es decir de 10 a 20 se encuentran a la misma distancia que 50 de 100, por que ambos tienen una variación del 100 % en la escala aritmética se situarían mucho mas distante 50 de 100 que 10 de 20, por lo tanto en situaciones es mejor utilizar esta escala. (Cruz, 2015)(Departamento de Fisica Aplicada, 2013)

CAPÍTULO 3. DESARROLLO PRIMERA PARTE

3.1 PARTE A

En esta parte del laboratorio, se pide que se grafique la función $a(x) = log_5(6 * x + 3)$ y la función $b(x) = sin(4 * log_2(x + 1)) + cos(4 * log_6(x + 9))$, sin embargo, escrito tal cual es imposible para matlab procesar y graficar logaritmos que no son naturales. Ante esto, se ve la necesidad de aplicar una de las propiedades de los logaritmos, la cual es $log_x(y) = \frac{log(y)}{log(x)}$.
Una vez transformada la ecuación y teniendo previamente el intervalo de puntos con su respectiva separación entre ellos (la cual es de 0.01), se procedió con evaluar cada punto con cada una de las funciones, para dar lugar a los vectores resultantes. A estos vectores se les graficó de manera separadas a cada uno y luego un gráfico donde se aprecian ambas funciones en conjunto. A continuación se muestran las figuras obtenidas a partir de las funciones:

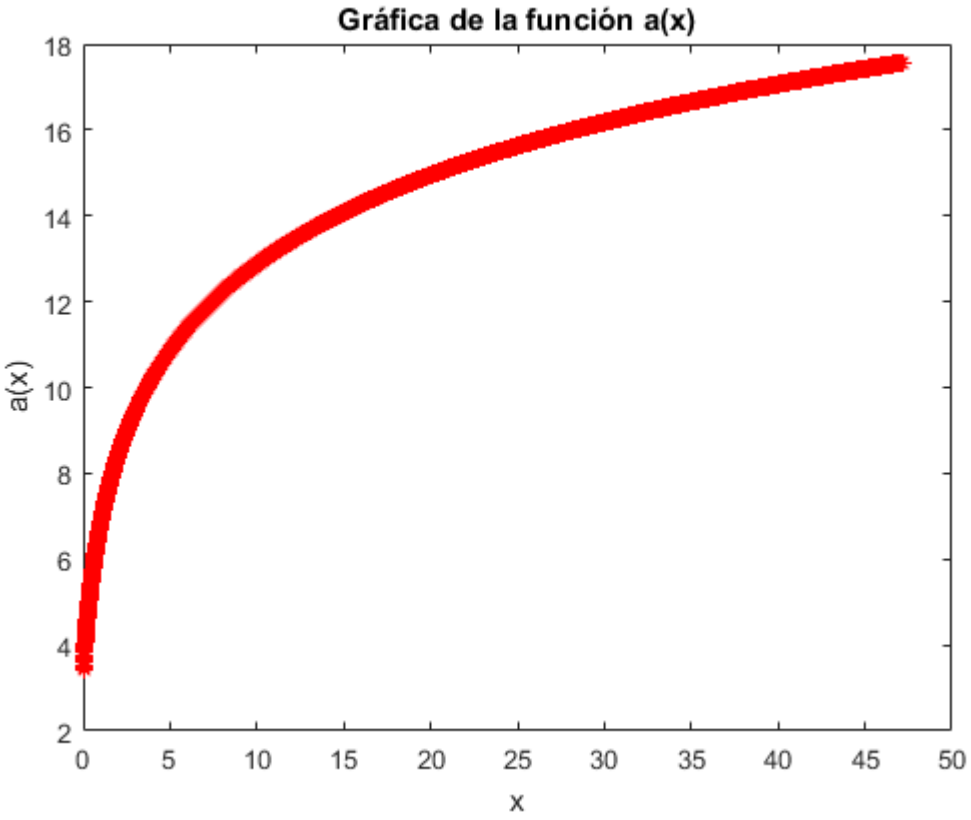

Figura 3.1: Gráfica de la función $a(x) = log_5(6 * x + 3)$.

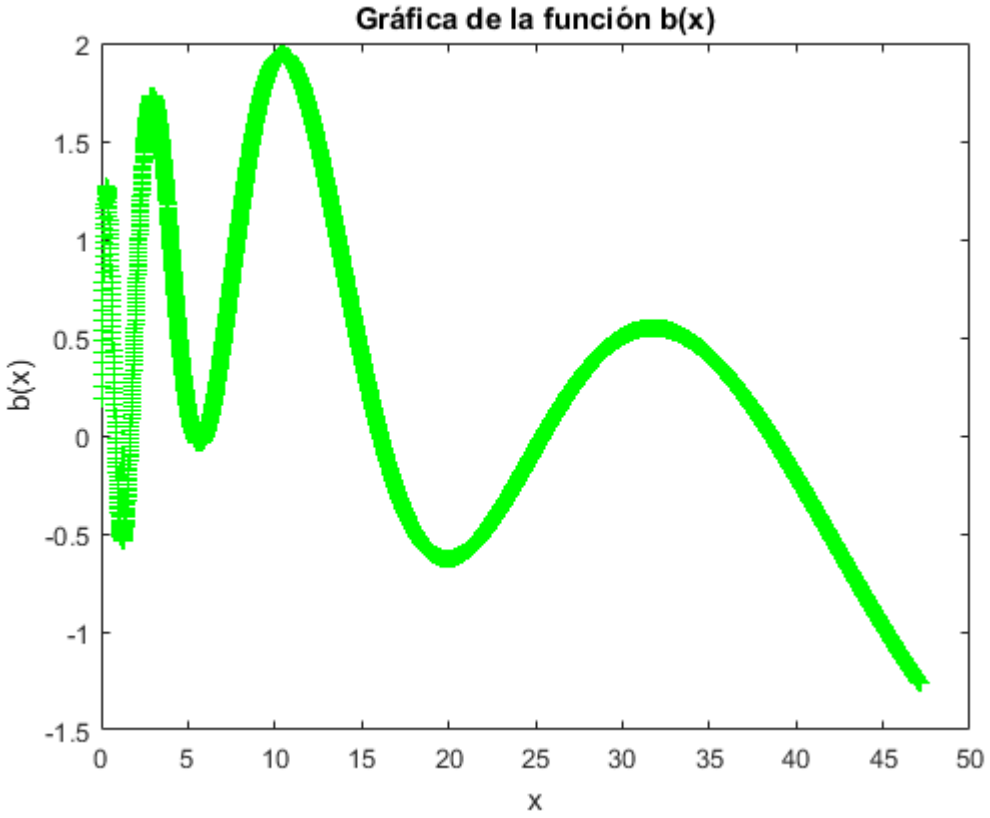

Figura 3.2: Gráfica de la función $b(x) = sin(4 * log_2(x + 1))$.

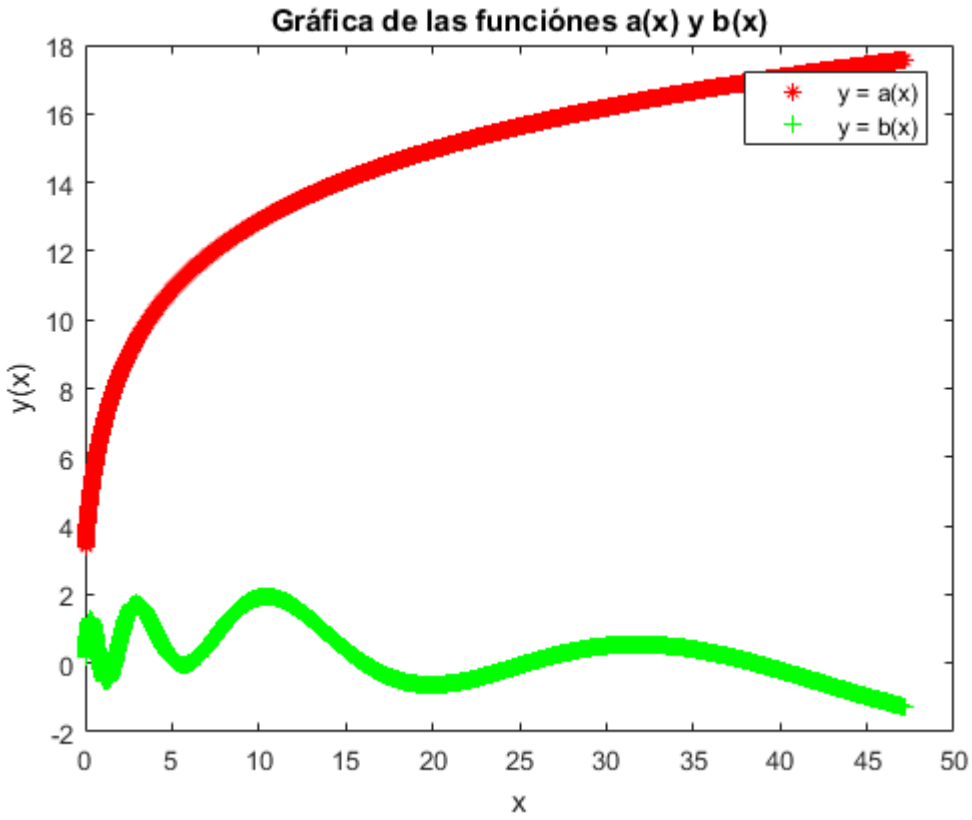

Figura 3.3: Gráfica de las funciones $a(x)$ y $b(x)$ juntas.

3.2 PARTE B

Para la parte b del laboratorio, se pide hacer un pequeño análisis de la escala logarítmica y la escala normal. Para esto, se crea un vector donde empieza desde el -10 y termina en 10 con una separación de $0{,}05$. Aplicando la función $e^{*}2x - 10$ a cada uno de los puntos obtenemos los puntos que se quieren graficar. Se le agrega grilla al gráfico, y luego se procede a comparar entre las dos escalas anteriormente mencionadas.

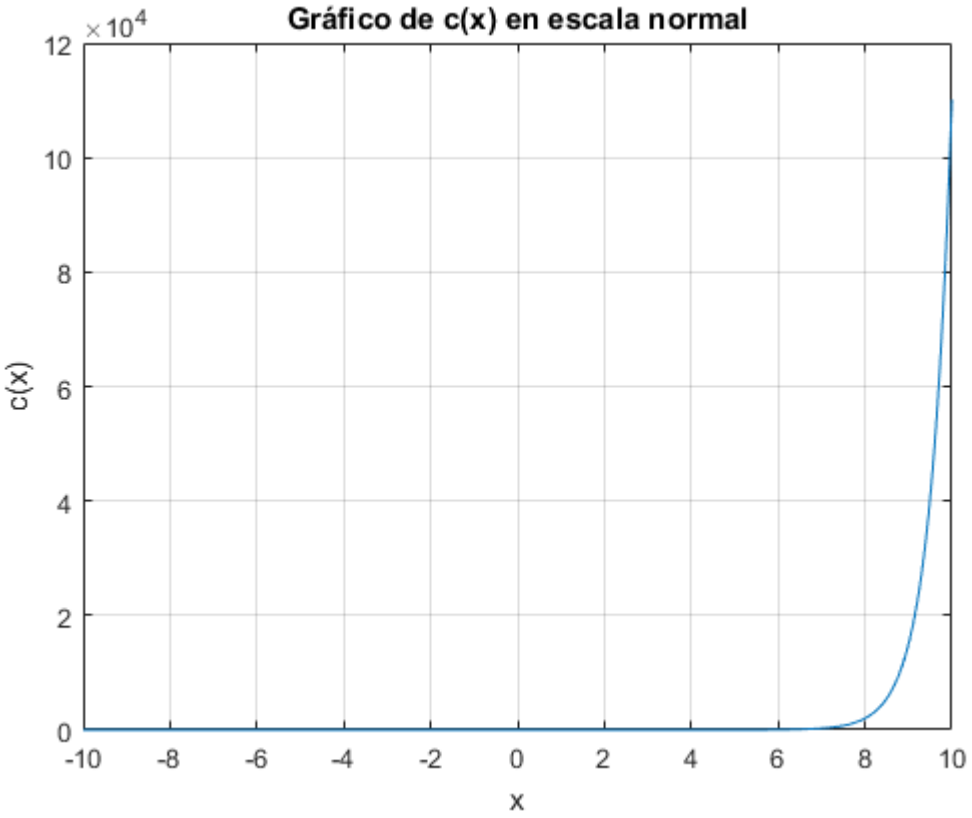

Figura 3.4: Gráfica de la función $c(x) = e^{2x-10}$ en escala aritmética (o normal).

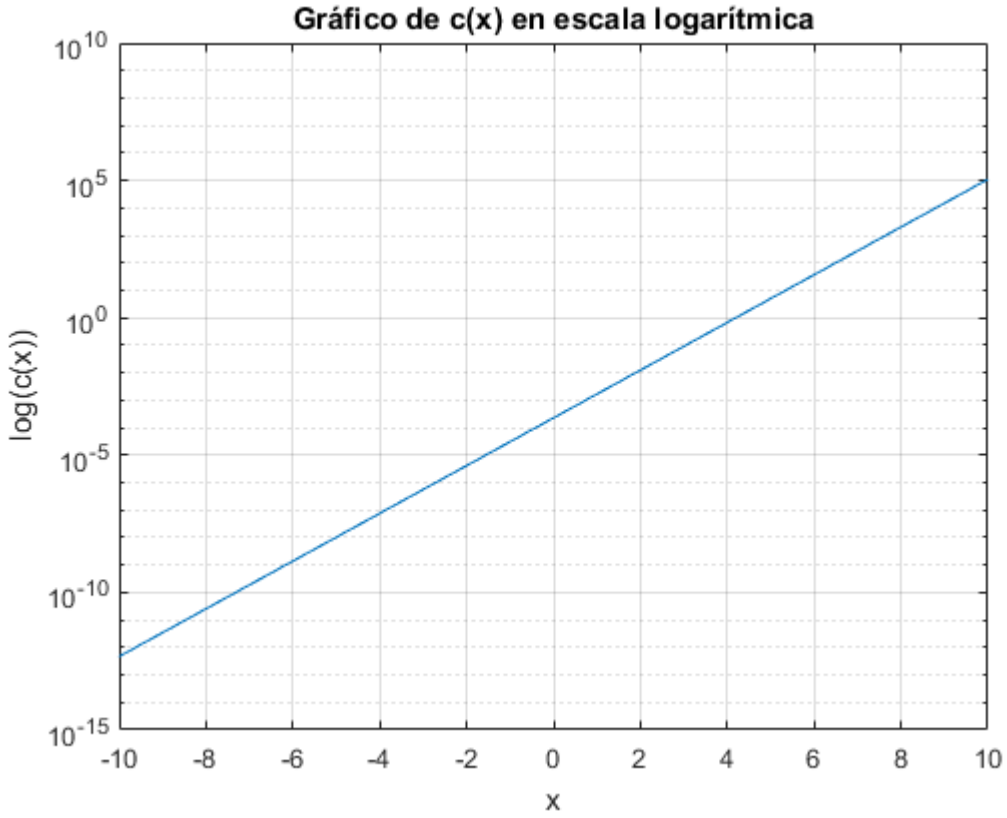

Figura 3.5: Gráfica de la función $c(x) = e^{2x-10}$ en escala logarítmica.

En lo que respecta a los gráficos, jamás se podrá decir cual es mejor que otro, ya que esto dependerá netamente del caso. Esto se da porque en la escala logarítmica se ve la tendencia de la curva, mientras que en la escala aritmética solamente se gráfica los valores de la curva. Resumiendo esto, se puede encontrar que en la escala logarítmica (semi-logarítmica) se tiene como ventaja mostrar la tendencia a largo plazo en funciones que crecen exponencialmente y así lograr mostrar de mejor forma la curva en un intervalo y no ver una especie de ”dos rectas”, una que se mantiene en el eje x y otra que sube infinitamente en el eje y. Para valores pequeños, con una variación porcentual muy pequeña, podría aprovecharse la ventaja de la escala aritmética, la cual es ser mucho más intuitiva que cualquier otra escala.

CAPÍTULO 4. DESARROLLO SEGUNDA PARTE

4.1 PARTE A

En esta parte se precisa implementar el algoritmo de Newton-Raphson, el cual retorna como resultado después de un numero definido de iteraciones (o por defecto, que la solución encontrada tenga un error mínimo permisible) el valor de una raíz de un polinomio cualquiera con solución. Para la implementación de éste, es necesario tener ciertos puntos a tener en consideración, por ejemplo el numero máximo de iteraciones permitido, el error a considerar para tener una raíz çonfiablez el punto de origen en el cual empezar a iterar (ya que si el polinomio tiene varias soluciones, la raíz a la que llegará dependerá del punto inicial).

A continuación se muestra el algoritmo utilizado para lograr una mayor comprensión al lector. En anexos se puede encontrar el código utilizado para implementar la función en MATLAB.

Algoritmo 4.1: Algoritmo para Newton-Raphson.

Data: polinomio, número máximo de iteraciones, error, valor inicial
Result: Raíz del polinomio

```
Calcular la derivada del polinomio;
Iteración ← 0;
while Queden iteraciones y el error es mayor al permitido do
    Aumentar la iteración;
    Guardar la raíz anterior;
    Calcular la nueva raíz a partir de la anterior;
end
Retornar la raíz x
```

4.2 PARTE B

Como última parte del laboratorio, se requiere una función robusta en términos de

validación de entradas, la cual dado un vector con números, se retorne el resultado de la suma de las raíces de los últimos cuatro números restado con la suma de las raíces de los primeros cuatro números. Para las validaciones se tomaron en cuenta tres aspectos:

- El vector no puede contener menos de 8 números
- El vector solamente puede contener dígitos
- El vector no puede contener números negativos

A continuación se muestra el algoritmo utilizado para lograr una mayor comprensión al lector. En anexos se puede encontrar el código utilizado para implementar la función en MATLAB.

Algoritmo 4.2: Algoritmo para restaRaices pedido en la parte 2b del laboratorio.

Data: Números

Result: Resta entre la suma de las raíces de los cuatro números de mayor valor menos la suma de las raíces de los cuatro valores inferiores.

```
Ordenar de menor a mayor los números;
if El numero más pequeño es negativo then
    Avisar de que no pueden ser números menores a 0;
    Salir;
end
if La cantidad de números es menor a 8 then
    Avisar de que se necesitan al menos 8 números;
    Salir;
end
if Hay algún valor no numérico then
    Avisar de que deben ser solamente números;
    Salir;
end
Calcular las raíces de los primeros cuatro números;
Sumar cada uno de los valores anteriores ← primerosCuatro;
Calcular las raíces de los últimos cuatro números;
Sumar cada uno de los valores anteriores ← ultimosCuatro;
Retornar la resta de ultimosCuatro − primerosCuatro
```

CAPÍTULO 5. MANUAL DE USUARIO

5.1 FUNCIÓN DE NEWTON-RAPHSON

Esta función necesita 4 parámetros, los cuales son:

1. Vector del polinomio en su representación

2. Numero máximo de iteraciones permitido

3. Valor aceptable del error

4. Punto de inicio

A continuación se muestran ejemplos de como llamar a la función para retornar el resultado esperado, teniendo el polinomio $P(x) = x^2 + 5x + 6$. Note que la función se llama "part2a":

```
>> part2a([1,5,6], 50, 10^-10, 5)

ans =

   -2.0000
```

Figura 5.1: Ejemplo 1 (Correcto)

```
>> part2a([1,5,6], 50, 10^-10, -5)

ans =

    -3
```

Figura 5.2: Ejemplo 2 (Correcto)

```
>> part2a([1,5,6], 10, 10^-20, -50)

ans =

   -3.0000
```

Figura 5.3: Ejemplo 3 (Correcto)

5.2 FUNCIÓN PARTE 2: B

Se procede a mostrar ejemplos de funcionamiento de la implementación de lo pedido en el laboratorio parte 2b, mostrando las validaciones y el caso en donde las entradas son correctas:

```
>> part2b([1,2,3,4,5,6,7,8,-1])
Debe ingresar numeros mayores o iguales a cero
```

Figura 5.4: Ejemplo 1 (incorrecto)

```
>> part2b([1,2,3,4,5,6,7,8,'c'])
Debe ingresar solo numeros
```

Figura 5.5: Ejemplo 2 (incorrecto)

```
>> part2b([1,2,3,4,5,6,7])
Debe ingresar al menos ocho numeros
```

Figura 5.6: Ejemplo 3 (incorrecto)

```
>> part2b([1,2,3,4,5,6,7,8])

ans =

    4.0135
```

Figura 5.7: Ejemplo 4 (Correcto)

CAPÍTULO 6. CONCLUSIONES

En el presente capitulo se presenta el cierre al trabajo realizado, se realiza una retrospectiva de como fue abordada la solución, se evalúa si los objetivos generales y específicos han sido cumplidos, dificultades presentadas en el desarrollo de la solución y lecciones aprendidas para futuros laboratorios.

Al observar el trabajo realizado, los resultados obtenidos y el análisis pertinente de estos resultados, se puede decir que se esta conforme con lo planteado en la solución, ya que los resultados fueron los esperados para ambas partes del trabajo y estos fueron obtenidos sin presentar mayores dificultades en la programación. A partir del trabajo realizado, el cual se cree que cumplió con las expectativas, ya que se considera que los objetivos generales y específicos fueron logrados de buena manera en el trabajo; se familiarizo con el lenguaje de programación Matlab, por lo cual los próximos laboratorios podrán ser realizados de buena manera, también se logro desarrollar las habilidades necesarias para graficar las funciones en ambas escalas, por otra parte también se tuvo éxito al implementar el método de Newton-Rapson. Tambien se considera que el informe fue echo a totalidad con explicaciones claras de ocmo se generaron los algoritmos para la . Como ya se ha mencionado las soluciones fueron creadas sin presentarse mayores complicaciones, ya sea a la hora de programar o comprender el enunciado; por esto no se presentar mayores lecciones aprendidas en cuanto a la realización del laboratorio, si se espera que lo realizado en esta experiencia sirva para poder desarrollar de mejor manera futuros trabajos, en los cuales se espera realizar poder entregar solucionas mas completas junto a un análisis mas en profundidad sobre lo obtenido.

CAPÍTULO 7. ANEXOS

7.1 CÓDIGO DE IMPLEMENTACIÓN PARTE 2A DEL LABORATORIO

```
1     %Se crea la función
2     function [x] = newton(y, maxIt, error, init)
3     dy = polyder(y); %dy es la derivada
4     x = init; %valor inicial
5     x_ant = Inf; %x anterior
6     iter = 0;
7     %Mientras el error sea mayor que el tolerado y no se
8     %haya pasado con el maximo de iteraciones:
9     while abs(x_ant-x) > error && iter < maxIt
10        iter = iter + 1;
11        x_ant = x;
12        %Se calcula el nuevo x y el anterior pasa a ser actual
13        x = x - polyval(y,x)/polyval(dy,x);
14    end
15    %Retorna x, que es el valor root
16    end
```

Figura 7.1: Implementación en MATLAB de Newton-Raphson.

7.2 CÓDIGO DE IMPLEMENTACIÓN PARTE 2B DEL LABORATORIO

```
1   function [result] = restaRaices(values)
2   %Se ordenan de menor a mayor
3   values = sort(values);
4
5   %Validations:
6   %No pueden ser menor a cero
7   if values(1) < 0
8       fprintf("Debe ingresar numeros mayores o iguales a cero\n")
9       return
10  end
11  %El largo del arreglo debe ser por lo menos 8
12  if length(values) < 8
13      fprintf("Debe ingresar al menos ocho numeros\n")
14      return
15  end
16
17  %Deben ser solo numeros
18  if ~isnumeric(values)
19      fprintf("Debe ingresar solo numeros\n")
20      return
21  end
```

Figura 7.2: Implementación en MATLAB de restaRaices, parte 1.

```
22
23  %Se procede a hacer el resultado
24  primerosCuatro = 0;
25  %Se suman los cuadrados de los primeros cuatro
26  for c = 1:4
27      primerosCuatro = primerosCuatro + sqrt(values(c));
28  end
29
30  %Se procede a sumar el cuadrado de los utlimos 4
31  ultimosCuatro = 0;
32  for c = length(values)-3:length(values)
33      ultimosCuatro = ultimosCuatro + sqrt(values(c));
34  end
35
36  %Se retorna el valor final
37  result = ultimosCuatro - primerosCuatro;
38  end
```

Figura 7.3: Implementación en MATLAB de restaRaices, parte 2.

BIBLIOGRAFÍA

Cruz, I. D. L. (2015). La escala aritmética y la logarítmica en los gráficos de análisis técnico. Recuperado desde https://www.ismaeldelacruz.es/la-escala-aritmetica-y-la-logaritmica-en-los-graficos-de-analisis-tecnico/".

Departamento de Fisica Aplicada, U. d. S. (2013). Escalas y graficas logaritmicas. Recuperado desde http://laplace.us.es/wiki/index.php/Escalas_y_gr%C3%A1ficas_logar%C3%ADtmicas#Escalas_logar.C3.ADtmicas".

MathWorks (2006). Descripcion general. Recuperado desde https://la.mathworks.com/products/matlab.html".