UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

LABORATORIO 1 MODELACIÓN Y SIMULACIÓN

INTRODUCCION A MATLAB

DIEGO POLANCO

DANY RUBIANO

Profesor: Gonzalo Acuña

Ayudantes: Bryan Guzmán

Robinson Oyarzún

Santiago - Chile

9 de abril de 2017

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

CAPÍTULO 1. INTRODUCCIÓN

1.1 MOTIVACIÓN Y ANTECEDENTES

Al observar nuestro alrededor podemos notar una serie de fenómenos que se explican o fundamentan a través de modelos matemáticos.

En ocasiones dichos modelos necesitan de gran exactitud, y una simple calculadora no alcanza a proveer esa necesidad. Suele ser en casos muy importante la perdida o la poca exactitud numérica, como lo es en el ámbito físico, astronómico, entre otros. Es por esto que es necesario utilizar un análisis más profundo para que a través de el diseño de algoritmos numéricos y reglas matemáticas se pueda obtener una mayor exactitud y explicación de fenómenos complejos de la realidad. El área de la informática ha sido un apoyo fundamental en la proliferación del desarrollo de herramientas con gran capacidad de computo capaces de calcular y graficar los distintos modelos matemáticos, permitiendo así construir análisis con mayor exactitud y profundidad. Algunos ejemplos de estas herramientas que se pueden mencionar son Numpy, software R, MATLAB, entre otros. Siendo esta ultima herramienta de software la llamada a utilizar en este caso.

1.2 OBJETIVOS Y ORGANIZACIÓN DEL DOCUMENTO

El principal objetivo de este documento es lograr aprender a utilizar la herramienta MATLAB y lograr familiarizarse con ella, para facilitar el desarrollo de los próximos laboratorios.

Es por esto que en el siguiente documento se mostrará a través de un marco teórico, el sustento base para la realización de este laboratorio introductorio. Luego se desarrollarán en dos partes, la gráfica de una función específica y la implementación del método de Newton-Raphson. A continuación se dará a conocer el funcionamiento del programa realizado en este laboratorio mediante un manual de uso, el cual especificará su correcta utilización mediante ejemplos. Finalmente se realizarán las conclusiones acerca de lo desarrollado en el presente documento.

CAPÍTULO 2. MARCO TEÓRICO

2.1 MÉTODO DE NEWTON-RAPHSON

El método de Newton-Raphson es considerado eficiente para encontrar aproximaciones de las raíces de una función real. Este método es un método que no asegura su convergencia global (método abierto). Es importante mencionar que al método Newton-Raphson se le dificulta la convergencia cuando la función tiene pendientes grandes o múltiples puntos de inflexión cerca de la raíz. La formula de iteración para este método es la siguiente:

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

2.2 BUBBLE SORT

Bubble sort o método de ordenamiento burbuja corresponde a un algoritmo simple de ordenamiento el cual se encarga de distribuir los elementos de una lista ya sea de manera creciente o decreciente. Su funcionamiento consiste básicamente en una comparación simple de los elementos ya sea arreglo o lista con su sucesor, comparando si el elemento es mayor o menor que su sucesor y desplazándolo o no según lo requiera la planificación del algoritmo. Este proceso se repite hasta realizar todas las combinaciones de comparación de la lista o arreglo, logrando que exista un orden, ya sea creciente o decreciente. Este es uno de los algoritmos más usados debido a su sencillez de programación. Sin embargo no es considerado como un algoritmo eficiente, ya que se puede apreciar que mientras mayor sea el tamaño de la lista o arreglo, aumentará considerablemente el número de comparaciones a realizar.

2.3 MATLAB

MATLAB corresponde a una herramienta la cual mediante a un entorno de desarrollo integrado o IDE permite realizar operaciones de índole matemática con un lenguaje de programación propio.

“La plataforma de MATLAB está optimizada para resolver problemas de ingeniería y científicos. El lenguaje de MATLAB, basado en matrices, es la forma más natural del mundo para expresar las matemáticas computacionales. Los gráficos integrados facilitan

la visualización de los datos y la obtención de información a partir de ellos. Una vasta librería de toolboxes preinstaladas le permiten empezar a trabajar inmediatamente con algoritmos esenciales para su dominio. El entorno de escritorio invita a experimentar, explorar y descubrir. Todas estas herramientas y prestaciones de MATLAB están probadas y diseñadas rigurosamente para trabajar juntas."(The MathWorks, Inc., 2017d).

2.3.1 Funciones importantes

- polyval: polyval o polynomial evaluation es una función que se encarga de calcular el valor final del polinomio en el punto indicado, por lo que recibe de parámetros el polinomio y el punto a evaluar (The MathWorks, Inc., 2017c).
- polyder: polyder o polynomial differentiation es una función que se encarga de calcular la derivada del polinomio ingresado, por lo que éste es el único parametro a ingresar (The MathWorks, Inc., 2017b).
- abs: abs corresponde a una función que se encarga de calcular el valor absoluto de una cifra indicada, por lo que solo requiere como parámetro algún valor numérico (The MathWorks, Inc., 2017a).

CAPÍTULO 3. DESARROLLO

3.1 PARTE I: EXPLICACIÓN DEL DESARROLLO DE CADA UNO DE LOS GRÁFICOS.

3.1.1 Gráficos de a(x) y b(x)

Las funciones establecidas para el desarrollo de esta actividad se detallan a continuación:

$$a(x) = 4log_5(9x - 2) \tag{3.1}$$

$$b(x) = cos(4log_6(2x + 4)) + sen(2log_6(2x + 5)) \tag{3.2}$$

A partir de ello, dado que los gráficos a realizar corresponden a las funciones por tramos, es necesario establecer el intervalo de acción en que se mueve la variable x, equivalente a $[0, 10\pi]$ y el espaciado entre sus valores dado por $0, 01$. En lo que sigue, para la representación de cada una de las funciones en MatLab, es necesario efectuar un cambio de base que permita emular los logaritmos presentes usando la siguiente ecuación:

$$log_b x = \frac{log_{10} x}{log_{10} b} \tag{3.3}$$

Una vez que $a(x)$ y $b(x)$ están representados, se procede a utilizar los comandos dispuestas por MatLab que permiten realizar los gráficos con las características deseadas. Para ello, se utiliza el comando $figure$ la que permite crear una nueva ventana para cada gráfico y $plot$ que crea un gráfico de los datos $a(x)$ o $b(x)$ según corresponda, en comparación con los valores del intervalo en que se mueve x. Este último permite también agregar ciertos parámetros que permiten especificar el estilo de linea, color y el símbolo marcador, por lo tanto son usados para las características pedidas para cada función, en conjunto con otros comandos que permiten agregar títulos y nombres de los ejes a cada uno de los gráficos.

A continuación se presenta el código escrito en MatLab que da lugar a la representación final de las funciones 3.1 y 3.2.

```
x=[0:0.01:10*pi]; %intervalo de x con espaciado de 0.01
a=4*log(9*x-2)/log(5);
b=cos(4*(log(2*x+3)/log(6)))+sin(2*(log(2*x+5)/log(6)));

figure; %permite crear una nueva ventana para cada grafico
plot(x,a,'g+'); %crea un grafico de los datos de la funcion en
    comparacion con los valores del intervalo en que se mueve x
xlabel({'x'},'Interpreter','latex'); %permite agregar el nombre del eje
     x
ylabel({'a(x)'},'Interpreter','latex'); %permite agregar el nombre del
    eje y
title({'a(x)=4log(9x-2)/log(5)'},'Interpreter','latex'); %permite
    agregar un titulo al grafico

figure;
plot(x,b,'r*');
xlabel({'x'},'Interpreter','latex');
ylabel({'f(x)'},'Interpreter','latex');
title({'b(x)=cos(4(log(2x+3)/log(6)))+sin(2(log(2x+5)/log(6)))'},'
    Interpreter','latex');

figure;
hold on; %retiene los graficos para agregar otros
plot(x,a,'g+');
plot(x,b,'r*');
xlabel({'x'},'Interpreter','latex');
ylabel({'b(x)'},'Interpreter','latex');
title({'Funciones a(x) y b(x)'},'Interpreter','latex');
legend('a(x)','b(x)');
hold off; %termina con la retension inicial
```

Ahora, basta con mostrar el resultado final de la ejecución del trozo de código mostrado anteriormente, expresado en las siguientes figuras:

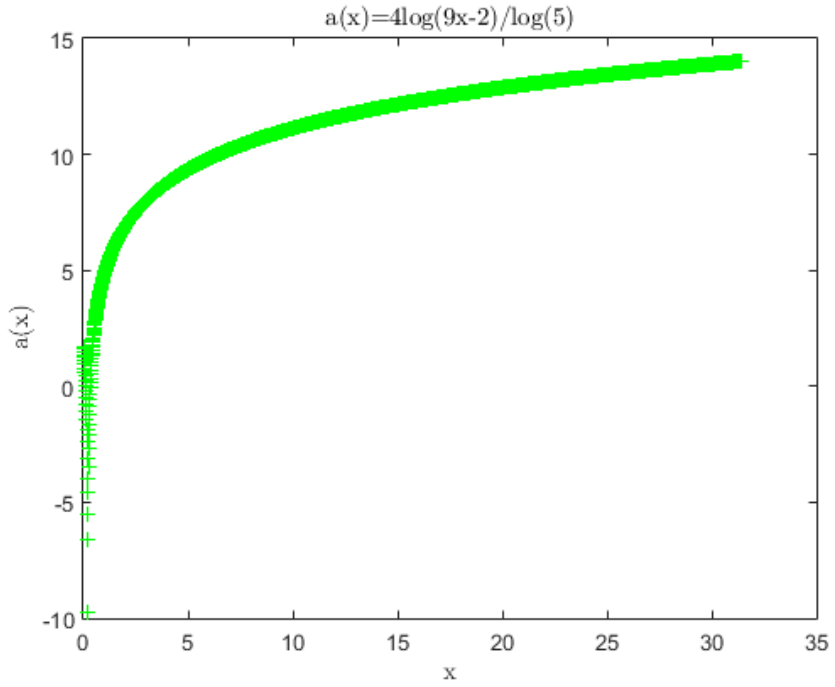

Figura 3-1: Función a(x)

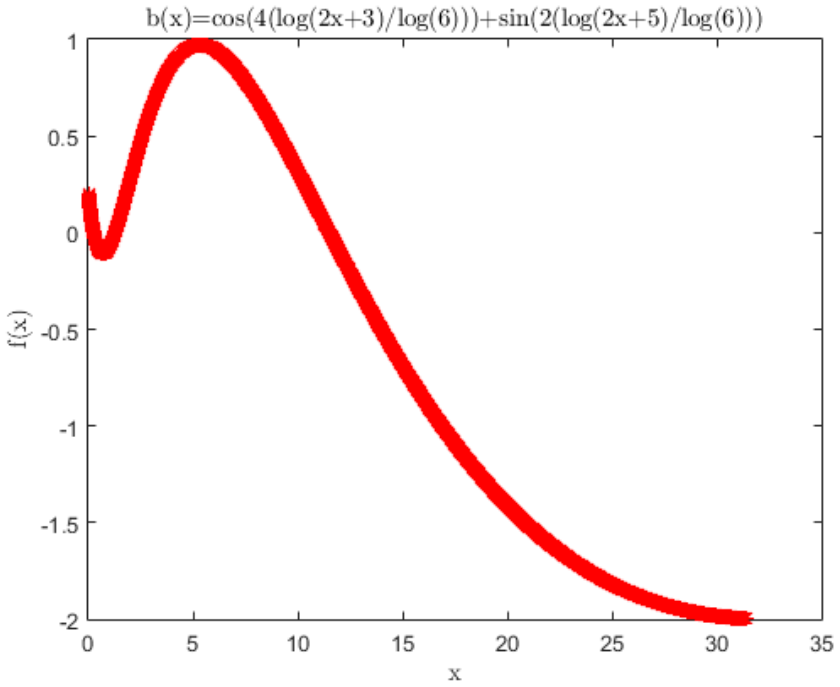

Figura 3-2: Función b(x)

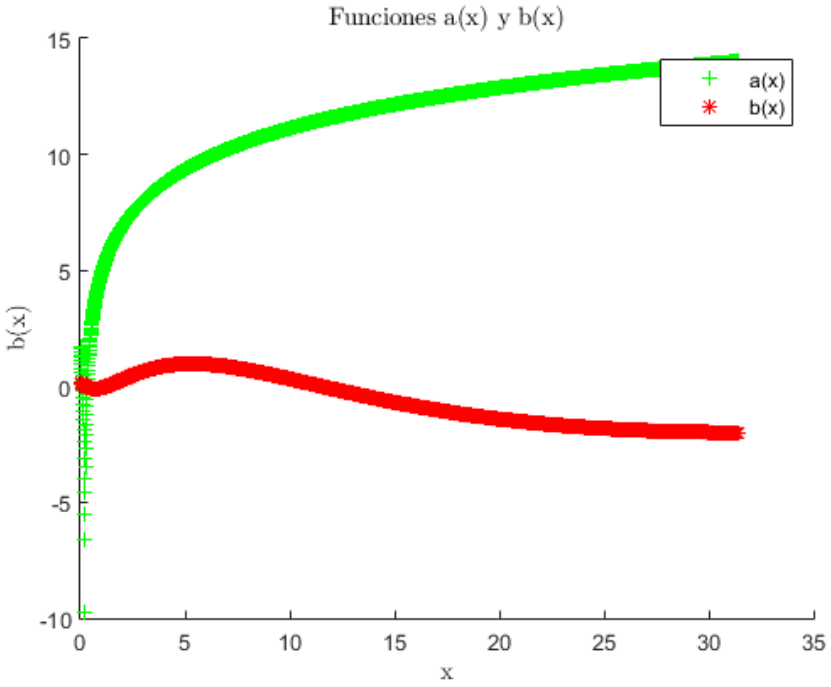

Figura 3-3: Funciones a(x) y b(x)

A modo de un pequeño análisis fundamentado en las características de los sistemas descritos por las funciones graficadas, de la figura 3.1, se puede observar que $a(x)$ en el intervalo $[0, 10\pi]$ se torna críticamente amortiguado de tal forma que van desapareciendo las oscilaciones, lo que indica que sus polos son mayores o iguales a 1, por lo que no presentan polos complejos conjugados, sino solamente polos reales.Todo lo contrario se puede detallar a partir de la figura 3.2, donde $b(x)$ se torna medianamente amortiguado y sus polos son complejos conjugados. Su diferencia se puede detallar de manera más clara en la figura 3.3.

Cabe señalar que en la figura 3.3 se usa el comando $hold$ para permitir acoplar las dos funciones bajo un mismo gráfico.

3.1.2 Gráfico de c(x)

En esta ocasión es el turno de la función c(x) representada a continuación:

$$c(x) = 6e^{(9x-4)} \tag{3.4}$$

Similar al proceso visto en la sección anterior, se define el tramo de la función en que se mueve la variable x correspondiente al intervalo de $[-10, 10]$ y el espaciado entre sus valores dado por $0,05$. Para la representación de la función, el comando exp de MatLab permite emular el euler detallado en la expresión. Para esta oportunidad se solicita graficar la función en escala normal y logarítmica, para lo cual se sigue un proceso similar utilizando los comandos vistos anteriormente, a excepción de que se hace necesario introducir un nuevo comando que permita realizar la representación logarítmica requerida. Dicho comando es $semilogy$ y reemplaza al comando $plot$.

A continuación se presenta el código escrito en MatLab que da lugar a la representación final de la función 3.4, en las diferentes esaclas solicitadas.

```
x=[-10:0.05:10]; %intervalo de x con espaciado de 0.05
c=6*exp(9*x-4);

figure;
plot(x,c,'b-');
xlabel({'x'},'Interpreter','latex');
ylabel({'c(x)'},'Interpreter','latex');
title({'c(x) en Escala Normal'},'Interpreter','latex');

figure;
semilogy(x,c,'g-'); %grafica con cambio de escala logaritmica en base
    10 al eje c(x)
xlabel({'x'},'Interpreter','latex');
ylabel({'c(x)'},'Interpreter','latex');
title({'c(x)en Escala Logartimica'},'Interpreter','latex');
```

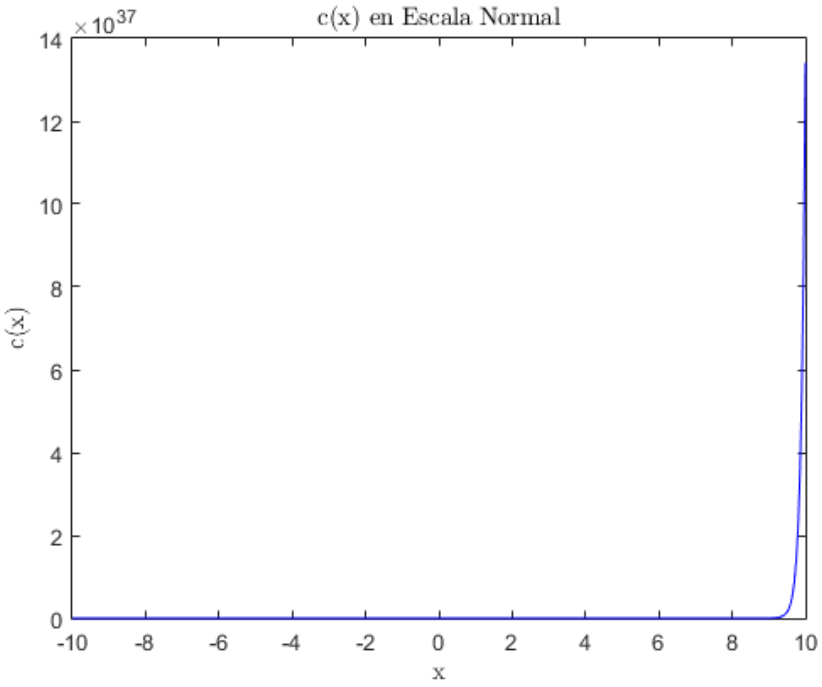

Figura 3-4: c(x) en escala normal

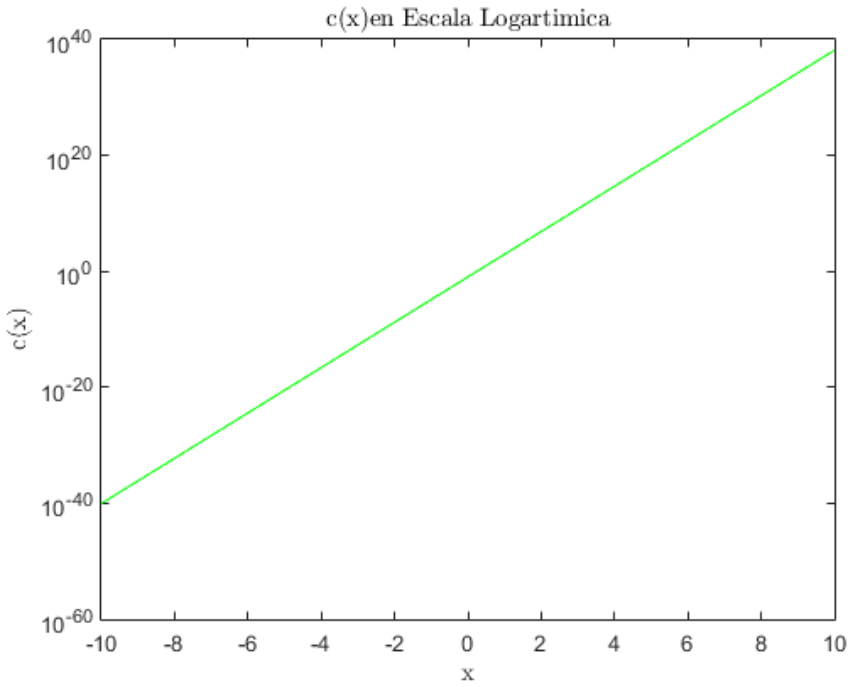

Figura 3-5: c(x) en escala logarítmica

En la figura 3.4 se puede observar propiamente las características curvas de una función exponencial en su escala normal. En más detalle, la función en el intervalo detallado tiene una pendiente nula siendo constante en $c(x) \approx 0$, por lo que a primera vista se podría decir que la función presenta una asíntota horizontal en $c(x) = 0$. Luego se observa un abrupto cambio, en donde la pendiente pasa a ser indefinida matemáticamente y esta vez se podría decir que existe una asíntota vertical en $x = 10$. De la figura 3.5 se puede observar que la función presenta una pendiente constante, ya que el comando utilizado hace el cambio de escala logarítmica en base 10 al eje c(x). De esta forma se tiene un rango más manejable de los datos presentados, a diferencia del gráfico de la figura 3.4 en donde la escala normal refleja la amplia gama de valores que ocupa la función en el intervalo definido. Por lo que se puede inferir que el uso de la escala logarítmica puede proporcionar un medio de visualización de los datos más simple y fácil de comprender.

3.2 PARTE II: EXPLICACIÓN DE LA IMPLEMENTACIÓN DE CADA UNO DE LOS ALGORITMOS.

A continuación se presentan las funciones solicitadas en el laboratorio.

3.2.1 Newton-Raphson

La función N_Raphson recibe como parámetros polinomio que corresponde a la expresión algebraica a iterar, max_iter que corresponde al numero de iteraciones que tendrá este método, error que tiene relación con el margen de certeza del método y finalmente x_0 que corresponde al valor o punto inicial. En un comienzo se almacena en raíz el punto inicial ingresado y en df el polinomio derivado. Se comprueba si el valor absoluto del polinomio evaluado en el punto ingresado es menor o igual al error ingresado, entonces se dice que el punto es raíz del polinomio (condición de termino). Sin embargo, si no se cumple con la condición anterior se calcula un nuevo punto a evaluar, el cual corresponde a la resta entre el punto inicial y la división entre la evaluación del punto en el polinomio por la evaluación del punto en la derivada del polinomio. Luego se realiza una llamada a la función con este nuevo punto calculado, realizando la recursión. A continuación se puede apreciar la función antes descrita.

```
function []= N_Raphson(polinomio,max_iter,error,x0)
    raiz=x0;
    df=polyder(polinomio);

    if error >= abs (polyval(polinomio,raiz))
        fprintf ('La raiz aproximada es: %4f\n',raiz)
        return;
    else
        xn= x0 - polyval(polinomio,x0)/polyval(df,x0);
        N_Raphson(polinomio,max_iter - 1,error,xn);
    end
end
```

3.2.2 Manejar vectores

La función $manejarVectores$ recibe como parámetro un vector de números. En pri-

mera ocasión se realiza una comprobación del parámetro ingresado para que éste sea tanto de carácter numérico y que contenga al menos 4 elementos en el vector. Se deja un mínimo de 4 elementos, permitiendo que los 4 elementos mayores sean igual que los 4 elementos menores. Luego se realiza un ordenamiento mediante al método burbuja de manera creciente, almacenando en $vector1$ los 4 mayores valores ingresados y en $vector2$ los 4 menores valores ingresados. A continuación se realiza un ciclo for almacenando los valores en $acumulador1$ y $acumulador2$ la suma de los elementos mayores y la suma de las raíces de los elementos menores respectivamente. Siguiendo con el proceso, se le calcula la raíz a la suma de los elementos mayores y se resta con $acumulador2$, para finalmente mostrarlo por consola. A continuación se puede apreciar la función antes descrita.

```
function [] = manejarVectores(vector)

    if isnumeric(vector)==0 %verifica si el vector ingresado contiene
        solo valores numericos
        fprintf('El formato de los datos es invalido. Se deben ingresar
            valores numericos.')
    end
    if length(vector)<4 %verifica si el vector posee minimo cuatro
        elementos. Si posee 4 elementos la operacion debe ser cero, ya
        que los 4 mayores son igual que los 4 menores
        fprintf('La entrada de datos requerida debe tener 4 elementos
            como minimo.')
    end
    if isnumeric(vector)==1 && length (vector)>=4
        n=length(vector);
        %ordenamiento decreciente mediante burbuja
        for a= 1:n
            for j=1:n-1
                if vector(j)>vector(j+1)
                    aux=vector(j);
                    vector(j)=vector(j+1);
                    vector(j+1)=aux;
```

```
            end
        end
    end
    vector1=vector(n-3:n); %Vector con los 4 mayores valores
    vector2=vector(1:4); %Vector con los 4 menores valores

    acumulador1=0.0;
    acumulador2=0.0;
    for i=1:4
        acumulador1=acumulador1+(vector1(i)); %suma de los 4
            mayores valores
        acumulador2=acumulador2+sqrt(vector2(i)); %suma de la raiz
            de los 4 menores valores
    end
    acumulador1=sqrt(acumulador1); %raiz de la suma de los 4
        mayores valores
    resultado=acumulador1-acumulador2;
    fprintf('El resultado de la raiz cuadrada de la suma de los 4
        elementos de mayor valor,\nmenos el resultado de la suma de
        la raiz cuadrada de los 4 elementos de menor valor es: %8f\n
        ',resultado);
  end
end
```

CAPÍTULO 4. MANUAL DE USO

4.1 NEWTON-RAPHSON

Para la correcta utilización del método de Newton-Raphson, se debe ejecutar en la consola la función $N\_Raphson()$, ingresando como parámetros el polinomio como un vector, el numero de iteraciones como un entero, el error como un flotante y finalmente el valor o punto inicial como un entero. Por ejemplo: *N_Raphson([elemento_1, elemento_2,..., elemento_n], numero de iteraciones, error o margen de iteración, valor inicial de evaluación)*. A continuación se pueden apreciar tres ejemplos de la ejecución.

```
>> N_Raphson([1 2 4 5 6 7],5,0.01,3)
La raiz aproximada es: -1.378223
```

Figura 4-1: Newton-Raphson, ejemplo 1

Para el ejemplo anterior, el polinomio corresponde a: $x^5 + 2x^4 + 4x^3 + 5x^2 + 6x + 7$, con 5 iteraciones, margen de error de $0{,}01$ y valor inicial de evaluación igual a 3.

```
>> N_Raphson([15 4 34 21 -1],12,0.004,0)
La raiz aproximada es: 0.044422
```

Figura 4-2: Newton-Raphson, ejemplo 2

Para el ejemplo anterior, el polinomio corresponde a: $15x^4 + 4x^3 + 34x^2 + 21x - 1$, con 12 iteraciones, margen de error de $0{,}004$ y valor inicial de evaluación igual a 0.

```
>> N_Raphson([3 0 6 7 1],8,0.1,5)
La raiz aproximada es: -0.161566
```

Figura 4-3: Newton-Raphson, ejemplo 3

Para el ejemplo anterior, el polinomio corresponde a: $3x^4 + 6x^2 + 7x + 1$, con 8 iteraciones, margen de error de $0{,}1$ y valor inicial de evaluación igual a 5.

4.2 MANEJAR VECTORES

Para la correcta utilización de la función encargada de restar raíces, se debe ejecutar en la consola la función $manejarVectores()$, ingresando como parámetro las cifras a calcular como un vector. Por ejemplo: *manejarVectores([elemento_1, elemento_2, elemento_3,elemento_4,..., elemento_n]*. A continuación se pueden apreciar tres ejemplos de la ejecución.

```
>> manejarVectores([23,5,4,6])
El resultado de la raiz cuadrada de la suma de los 4 elementos de mayor valor,
menos el resultado de la suma de la raiz cuadrada de los 4 elementos de menor valor es: -5.316975
```

Figura 4-4: Manejar Vectores, ejemplo 1

```
>> manejarVectores([1,-2,5,78,98,56,4,2,0,6,67,77])
El resultado de la raiz cuadrada de la suma de los 4 elementos de mayor valor,
menos el resultado de la suma de la raiz cuadrada de los 4 elementos de menor valor es: 15.474330
```

Figura 4-5: Manejar Vectores, ejemplo 2

```
>> manejarVectores([2,3,7])
La entrada de datos requerida debe tener 4 elementos como minimo.
```

Figura 4-6: Manejar Vectores, ejemplo 3

En el último ejemplo se puede observar el error generado al ingresar menos de 4 elemtos en el vector.

CAPÍTULO 5. CONCLUSIONES

Como conclusiones, se puede mencionar en primer lugar que se pudo observar a partir de los gráficos desarrollados en la primera parte de la experiencia, algunos de los principios tratados en la cátedra del ramo, pudiendo encontrar las implicancias de los polos en la manera como el sistema encuentra su respuesta permanente después de la amortiguación de su respuesta temporal. Así mismo se pudo observar cómo la aplicación de una escala logarítmica puede proporcionar un medio de visualización de los datos más simple y fácil de comprender a diferencia de lo presentado a través de una escala normal, ya que se pasa de la representación de los datos en un amplio abanico de valores a un rango más manejable.

Por otra lado, el desarrollo de la segunda parte de esta experiencia, permitió revivir algunos métodos numéricos a partir del método de Newton Raphson desarrollado para esta ocasión, teniendo una leve dificultad en establecerlo de una manera recursiva.

Así mismo, se pudo hacer uso de uno de los algoritmos de ordenamiento para la disposición de los elementos del vector trabajado de menor a mayor, y de este modo poder realizar la operación solicitada.

Finalmente, se puede mencionar en primer lugar que se logró llevar a cabo los objetivos propuestos, de manera que se logró la introducción al trabajo con la herramienta MatLab y sus respectivos componentes.

En más detalle, a pesar de que en ramos anteriores se había hecho uso de esta herramienta, por propia naturaleza humana, estos conceptos se habían olvidado. Es por esto que esta experiencia ha sido de gran utilidad para recordar y aprender de algunos de los componentes que permitirán la resolución de los problemas a los que nos enfrentaremos en lo que resta de la asignatura.

CAPÍTULO 6. BIBLIOGRAFÍA

The MathWorks, Inc. (2017a). Documentation - Abs. Recuperado desde https://www.mathworks.com/help/matlab/ref/abs.html?searchHighlight=abs&s_tid=doc_srchtitle

The MathWorks, Inc. (2017b). Documentation - Polyder. Recuperado desde https://www.mathworks.com/help/matlab/ref/polyder.html

The MathWorks, Inc. (2017c). Documentation - Polyval. Recuperado desde https://www.mathworks.com/help/matlab/ref/polyval.html

The MathWorks, Inc. (2017d). Matemáticas. Gráficos. Programación. Recuperado desde https://es.mathworks.com/products/matlab.html