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

MODELACIÓN Y SIMULACIÓN

LABORATORIO 1

INTRODUCCIÓN A MATLAB

JUAN PABLO ROJAS ROJAS
NICOLÁS MARIÁNGEL TOLEDO

Profesor:
Gonzalo Acuña
Ayudante:
Francisco Muñoz

Santiago - Chile
7 de noviembre de 2018

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

CAPÍTULO 1. INTRODUCCIÓN

1.1 MOTIVACIÓN

MATLAB es una herramienta muy utilizada en el mundo actual de los científicos e ingenieros involucrados en la computación numérica y técnica. Esto se debe a que MATLAB posee un abundante conjunto de herramientas que permiten simplificar problemas complejos, además de también tener una muy buena documentación y comunidad.

1.2 OBJETIVOS

Para la presente experiencia se define un objetivo principal, el cual será .ªcercarse al manejo de MATLAB, con una introducción a este lenguaje interpretado."[1] Posteriormente será contrastado con todo el desarrollo de la experiencia y permitirá concluir si la misma fue provechosa, se definen los siguientes objetivos específicos.

- Realizar gráficos de funciones en MATLAB y poder cambiar la forma y el color con el que se muestra la figura.
- Realizar gráficos con escala normal y logarítmica de una función.
- Obtener la raíz aproximada de un polinomio cualquiera utilizando el método de Newton-Raphson.
- Poder reconocer errores de entrada en una función comprobando que la entrada de tal función es efectivamente un vector.

1.3 ORGANIZACIÓN DEL DOCUMENTO

El presente documento está estructurado básicamente en cinco capítulos. Comenzando con el siguiente capítulo donde se presenta la explicación de los conceptos necesarios para comprender este informe. Continuando con el capítulo donde se desarrolla la primera parte del enunciado [1], donde se grafican algunas funciones y se explican tales gráficos. Luego viene el capítulo del desarrollo de la segunda parte, donde se explican los algoritmos desarrollados para solucionar esta parte. En el próximo capítulo se muestran un breve manual donde se aclara el uso de las funciones utilizadas en la segunda parte. Finalmente, se presentan las conclusiones en donde se expone lo aprendido durante el desarrollo de la experiencia.

CAPÍTULO 2. MARCO TEÓRICO

2.1 MÉTODO NEWTON-RAPHSON

El método de Newton para hallar las raíces de la ecuación f(x) = 0, es el más conocido, y a menudo, el más efectivo. [2] Este método consiste en tomar una aproximación inicial, $\bar{x}$, y obtener aproximaciones x_n más refinadas mediante la siguiente fórmula recursiva:

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

En la siguiente figura se muestra la representación geométrica de este método para entender mejor como funciona:

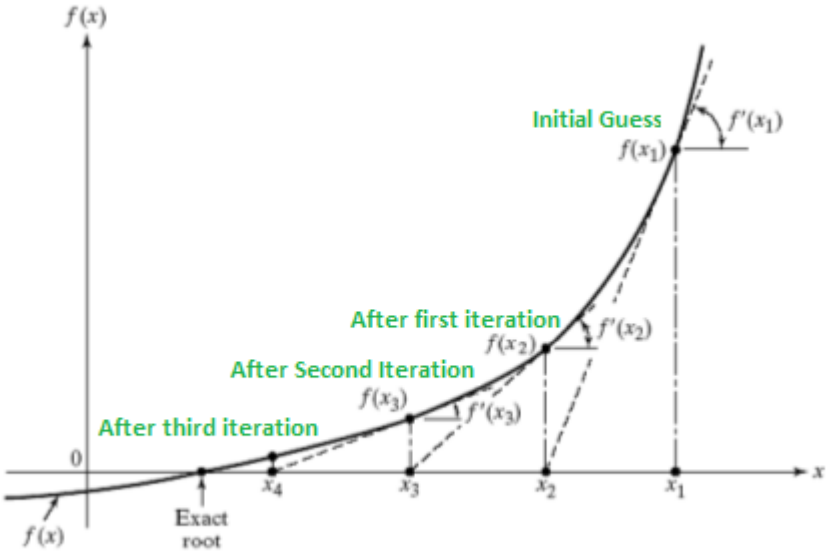

Figura 2.1: Interpretación geométrica del método Newton-Raphson [3].

La ecuación de la recta tangente que pasa por el punto $(x_n, f(x_n))$ viene dada por:

$$y - f(x_n) = f'(x_n)(x - x_n)$$

Si hacemos $y = 0, x = x_n + 1$, obtenemos la expresión (2.1):

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

2.2 ESCALA LOGARÍTMICA

Generalmente las relaciones entre variables se comportan de manera no lineal, es decir, la relación que existe entre una variable independiente y dependiente, no necesariamente se comporta de la forma lineal característica $y = Ax + B$. Claramente al tener una forma lineal de relación entre variables facilita en gran medida el análisis de dependencia y los distintos componentes que tiene una función lineal, por lo cual se intenta reducir las relaciones no lineales a una relación lineal.

Uno de los casos más frecuentes que existe es la dependencia exponencial, la cual tiene una función de la forma $y = Ke^{Ax}$, al analizar una función exponencial y obtener información de interés se torna bastante complejo, por lo que siempre se busca reducir esta expresión a una función lineal, para esto se utiliza la escala logarítmica en donde se aplica a la función exponencial en cuestión un procedimiento matemático para obtener esta función de forma lineal usando logaritmo natural:

Sea la función exponencial:

$$y = Ke^{\alpha x} \tag{2.2}$$

con K y α constantes. Luego se aplica logaritmo natural a la expresión:

$$ln(y) = ln(Ke^{\alpha x}) \tag{2.3}$$

$$ln(y) = ln(K) + \alpha x \tag{2.4}$$

Sea $ln(K)$ una constante A, la forma lineal final es de la forma:

$$ln(y) = A + \alpha x \tag{2.5}$$

Así la transformación de esta función exponencial permite realizar un análisis más sencillo a partir de su forma lineal, además gráficamente se comportan de la siguiente forma:

Escala normal:

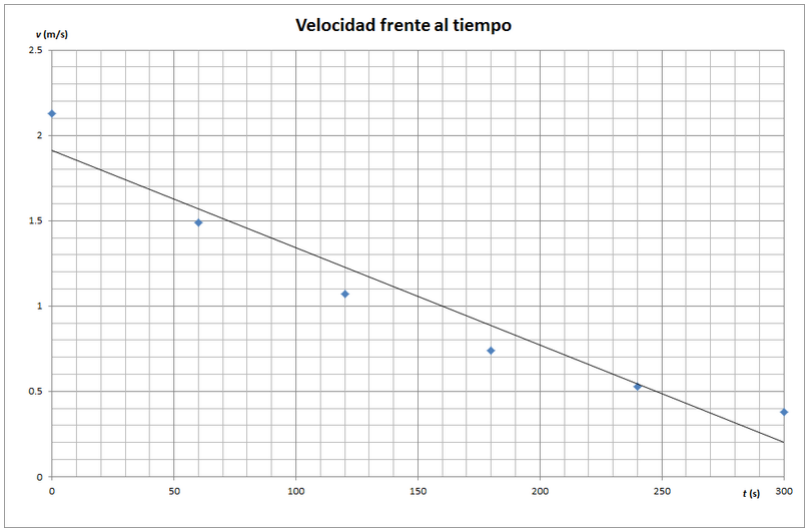

Figura 2.2: Gráfico velocidad vs tiempo: $v = Ke^{-\lambda t}$ [4]

Escala logarítmica:

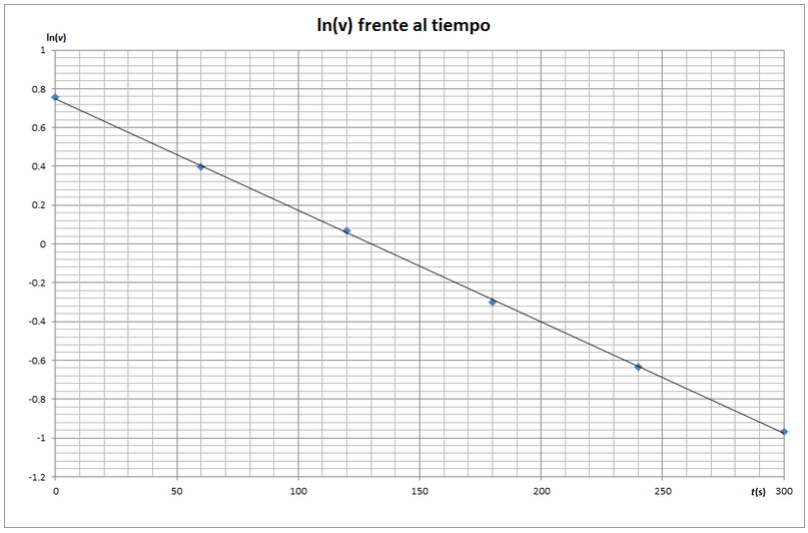

Figura 2.3: Gráfico ln(v) vs tiempo: $ln(v) = ln(Ke^{-\lambda t})$ [4]

CAPÍTULO 3. DESARROLLO DE LA PRIMERA PARTE

Para esta primera parte del documento, como bien se mencionó en los objetivos, se debe graficar funciones en MATLAB.

3.1 GRÁFICO DE FUNCIONES

En primera instancia se graficaron dos funciones con las cuales se consiguió realizar configuraciones de color y de figuras que representan cada uno de los puntos de la curva, dando vista a cada una de las funciones por separado y otra donde se encuentren ambas funciones en un mismo plano.
Las funciones graficadas son las siguientes:

$$a(x) = 3\log_5(3x+1) \tag{3.1}$$

$$b(x) = sen(2(log_2(x+11))) + cos(5(log_6(3x+27))) \tag{3.2}$$

Cabe destacar que para efectos de programación se realizó una serie de cambios de base para ajustar el logaritmo en base 10(el cual está por defecto en MATLAB) a logaritmos con base según fuera requerido por las funciones $a(x)$ o $b(x)$.
Algunos aspectos a considerar:

- La función 3.1 se debe graficar en rojo con símbolo $*$ para cada punto.
- La función 3.2 se debe graficar en azul con símbolo $+$ para cada punto.
- El intervalo dominio debe ser $[0, 15\pi]$ con espacio 0.01 entre ellos.

Por otro lado, en la siguiente instancia se pide graficar la siguiente función:

$$c(x) = 2e^{x+10} \tag{3.3}$$

Algunos aspectos a considerar:

- La curva debe tener colores y estilo a elección.
- El intervalo dominio debe ser [-10,10] con espacio de 0.05 entre ellos,
- Además del gráfico en escala normal, se debe incluir un gráfico en escala logarítmica de dicha función.
- El plano de los gráficos debe ser cuadriculado.

3.1.1. Funciones para graficar en MATLAB

El procedimiento para graficar consta de dos partes, una función en MATLAB para graficar funciones por separado dando la posibilidad de elegir entre escala normal y logarítmica y otra función para graficar ambas curvas en un mismo plano.
La primera función es la siguiente:

$$function \quad [] = graficar(funcion, ejex, ejey, nombre\_grafico, color\_figura, log)$$

Donde:

- funcion: nombre de la función a graficar, usado para nombrar eje de coordenadas Y.
- ejex: vector correspondiente al eje x del plano definido en un intervalo con cierta distancia entre sus valores.
- ejey: vector obtenido de la aplicación de una expresión sobre el vector .ejex"que corresponde al eje y del plano.
- nombre_grafico: título para el gráfico.
- color_figura: color con la respectiva figura para los puntos(+,*,etc).
- log: booleano para indicar si el gráfico va en escala normal o logarítmica.

A través de la función propia de MATLAB $plot$ se puede realizar la gráfica dado dos vectores de entrada correspondientes al eje x e y de un plano, lo cual claramente esta implementado al interior de esta función con la característica de poder editar el color de la curva como también la forma de los puntos representados en el plano. Por otro lado, esta función permite graficar en escala logarítmica si es necesario, ya que mediante el parámetro de entrada log con valor booleano $true$ permite activar la sentencia de control $if - else$ del código, para dar paso a la función propia de MATLAB $semilogy$ que recibiendo los mismos parámetros y permitiendo la edición de color de curva y forma de los puntos que la función MATLAB $plot$ da la visualización de la gráfica en escala logarítmica.
Por otro lado, se implementa la siguiente función para graficar en conjunto las funciones 3.1 y 3.2:

$$function \quad [] = graficar\_ambos(funciones, ejex, ejey1, ejey2, nombre\_grafico, color\_figura1, color\_figura2)$$

Donde:

- funciones - nombre de las funciones utilizadas para nombrar el eje y
- ejex - vector correspondiente al eje x del plano definido en un intervalo con cierta distancia entre sus valores.

- ejey1 - primer vector obtenido de la aplicación de una expresión sobre el vector .^{ej}ex"que corresponde al eje y del plano.
- ejey2 - segundo vector obtenido de la aplicación de una expresión sobre el vector .^{ej}ex"que corresponde al eje y del plano.
- nombre_gráfico - título para el gráfico.
- color_figura1 - color con la respectiva figura para los puntos(+,*,etc) del primer gráfico.
- color_figura2 - color con la respectiva figura para los puntos(+,*,etc) del segundo gráfico.

Luego, para juntar ambas funciones en el mismo plano, al igual que en los procedimientos anteriores se utiliza la función propia de MATLAB $plot$ además de las sentencias $hold\,on - hold\,off$ para generar las gráficas en el mismo plano.

3.1.2. Resultados obtenidos

Los resultados obtenidos constan de los siguientes gráficos:
En primera instancia se presentan los gráficos correspondientes a las funciones 3.1 y 3.2, cada una por separado y luego el gráfico con ambas curvas en el mismo plano:

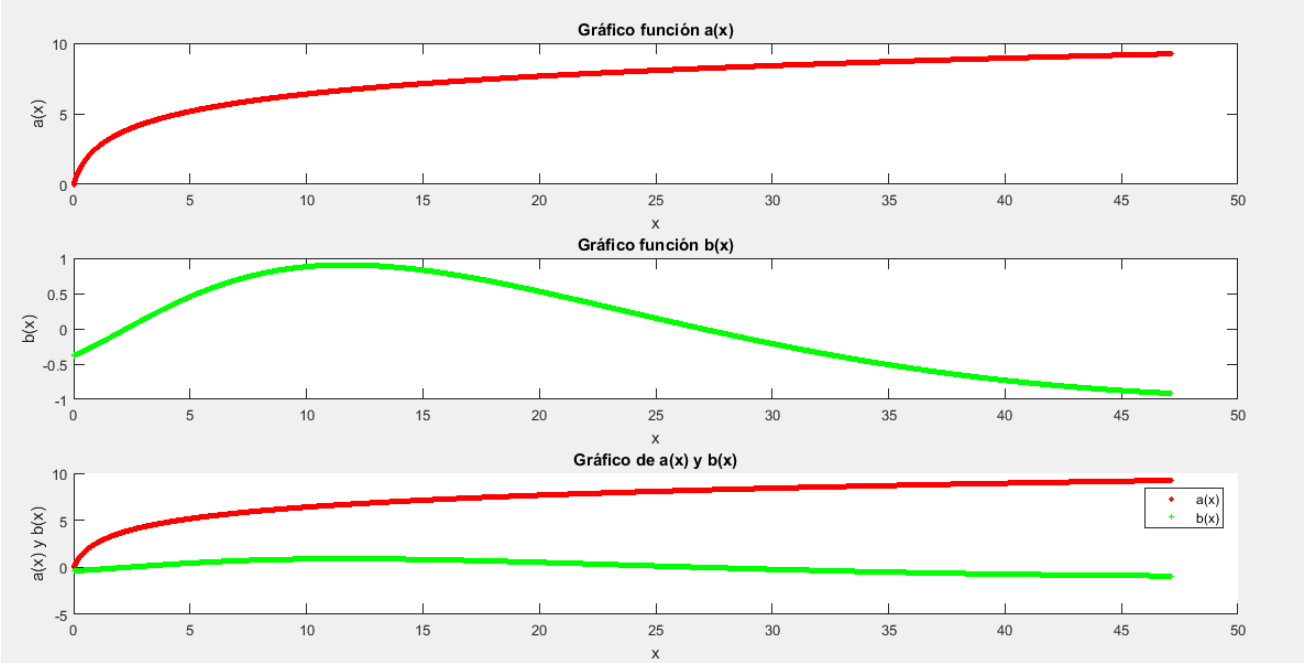

Figura 3.1: Gráficos funciones a(x) y b(x)

Por último, la gráfica correspondiente a la función 3.3, como también esta misma en escala logarítmica.

Para las cuadriculas en los planos se utilizó las sentencias propias de MATLAB correspondiente a $grid\,on$ y $grid\,minor$ para reducir el tamaño de estas.

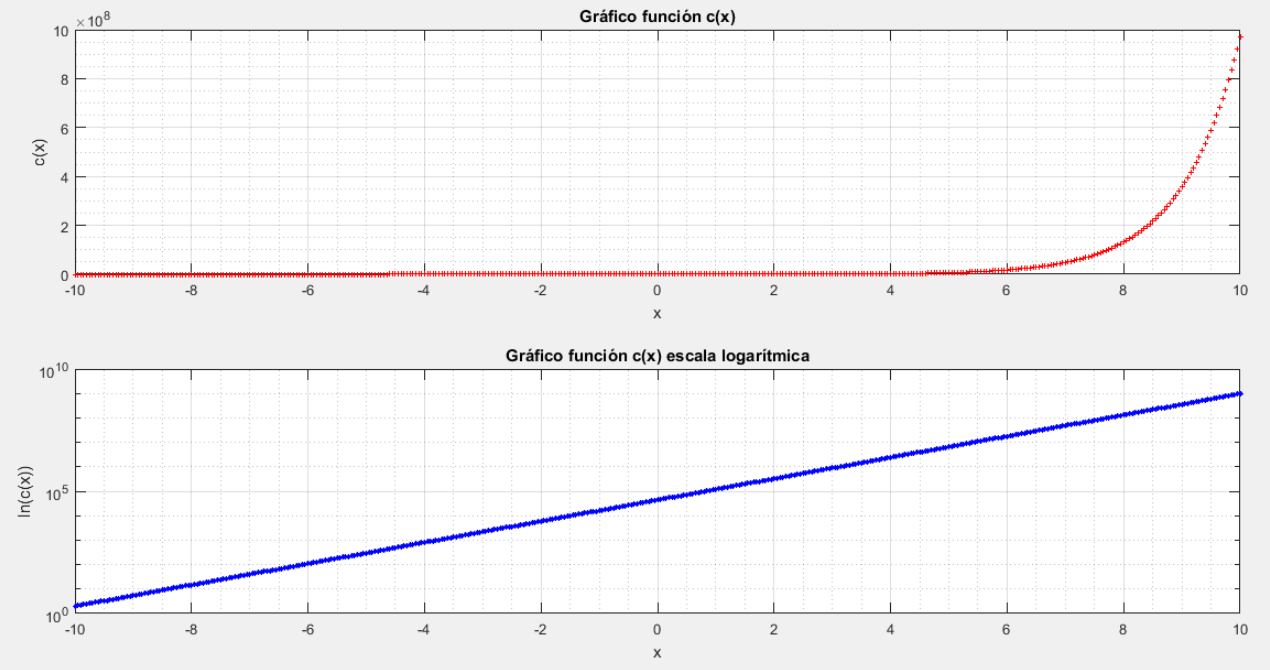

Figura 3.2: Gráficos funciones c(x) y escala ln(c(x))

3.1.3. Ventajas y desventajas de escalas usadas

Como bien se ha indicado en el capítulo 2, donde se aborda la escala logarítmica, el cambio de escala es con el principal objetivo de facilitar el análisis de la relación entre variable dependiente e independiente. Para el caso anterior luego de tener c(x), una función exponencial creciente así sin más, no nos permite obtener información específica y simple de relaciones que puede haber entre componentes, pero debido a la escala logarítmica, la transformación a una función lineal permite la ventaja de obtener componentes como pendiente e intercepto que apelan al análisis simple y directo de los componentes en cuestión. Si bien la función que se presenta es genérica, esto se podría extender a funciones que apelen al análisis temporal, análisis de relación directa o inversa, porcentaje de crecimiento directo, etc.

Si bien la escala logarítmica reduce los datos a una forma más manejable, en algunos casos si no es usado adecuadamente puede generar perdida de información o no ayudar a conseguir resultados que se esperan del análisis de datos y dependencias.

CAPÍTULO 4. DESARROLLO DE LA SEGUNDA PARTE

4.1 FUNCIÓN NEWTON-RAPHSON

$$function \quad solX = newton\_raphson(pol, maxIter, minErr, initialValue)$$

Donde:

- solX : La raíz aproximada que se calcula por medio del método de Newton.
- pol : Polinomio en el formato definido en el enunciado [1]
- maxIter : Máxima cantidad de iteraciones que debe ejecutar el algoritmo antes de retornar la solución.
- minErr : El error mínimo que debe tener la aproximación de la raíz que se quiere calcular para detener el algoritmo.
- initialValue : Valor inicial de aproximación que se utiliza para aproximar a la raíz del polinomio.

Esta función actúa recursivamente, tiene un sólo caso base, que es cuando la cantidad de iteraciones es igual a 0, cuando se cumple esto la función retorna la raíz aproximada que ha calculado hasta esta iteración. Luego en el caso recursivo se calcula la raíz aproximada utilizando el método de Newton-Raphson y el valor inicial recibido como parámetro, luego se calcula el error que tiene esta aproximación en comparación a la anterior. Dependiendo del error calculado se tienen dos casos, en el primer caso que sucede cuando el error obtenido es menor al error mínimo definido, se detiene el algoritmo y se retorna la raíz aproximada recién calculada. En el segundo caso que es cuando el error obtenido es mayor al error mínimo definido, se llama a esta misma función, pero se quita una iteración del contador maxIter y se cambió el valor de initialValue al valor obtenido de la raíz aproximada en esta iteración.

En la sección 5.2 se muestran ejemplos del uso de esta función y los resultados que entrega para cada caso.

4.2 FUNCIÓN RESTA DE RAÍCES CUADRADAS

$$function \quad vector\_sum\_minus(vector)$$

Esta función recibe un vector en el formato:

[90 34 98 21 38 29 10...]

Donde los elementos que tiene este vector pueden no ser necesariamente números y puede que no haya suficientes datos para tener 4 elementos menores y 4 elementos mayores donde ambos conjuntos sean distintos entre sí (no repetir elementos en cada conjunto, esto no significa que no se puedan repetir los mismos números). Debido a esto, esta función también comprueba que los datos ingresados en el vector sean todos del tipo numérico y que haya al menos 8 elementos en el vector antes de realizar los cálculos pertinentes, en caso de que no se cumplan ambas condiciones se muestra un error por pantalla indicando que se debe solucionar (esto se puede ver en la sección 5.2).

Luego esta función calcula el resultado de la raíz cuadrada de la suma de los 4 elementos de mayor valor, menos el resultado de la suma de la raíz cuadrada de los 4 elementos de menor valor para imprimir este valor por pantalla.

CAPÍTULO 5. MANUAL DE USO

Los programas usados para la confección de este informe se crearon en el entorno de MATLAB 2017b, por lo que lo primero que se debe hacer para poder utilizar estos programas es tener instalado este programa. En esta sección se muestra cómo utilizar la función de Newton-Raphson y la función que resta las raíces.

5.1 MANUAL NEWTON-RAPHSON

Acordar que la función de Newton-Raphson es la siguiente:

$$function \quad solX = newton\_raphson(pol, maxIter, minErr, initialValue)$$

Para ocupar esta función, se debe definir un polinomio al cuál se le quiere obtener una aproximación de la raíz, la cantidad máxima de iteraciones que debe ejecutar el algoritmo antes de detenerse, el error mínimo que debe tener la solución encontrada para detener el algoritmo y el valor inicial que se debe utilizar para empezar a buscar una aproximación. A continuación, se presentan tres ejemplos del uso de esta función en MATLAB.

```
>> solucion = newton_raphson([1 2 1], 100, 10^(-10), 2)

solucion =

   -1.0000
```

Figura 5.1: Primer ejemplo del uso de la aproximación de raíces por el método Newton.

```
>> solucion = newton_raphson([3 2 1 5], 1000, 10^(-12), 15)

solucion =

   -1.3428
```

Figura 5.2: Segundo ejemplo del uso de la aproximación de raíces por el método Newton.

```
>> solucion = newton_raphson([2 -5 -2 3 2 1 5], 1000, 10^(-12), 10)

solucion =

    2.5692
```

Figura 5.3: Tercer ejemplo del uso de la aproximación de raíces por el método Newton.

5.2 MANUAL RESTA DE RAÍCES CUADRADAS

Acordar que esta función es la siguiente:

$$function \quad vector\_sum\_minus(vector))$$

Y que si esta función recibe un vector que tiene menos de 8 elementos y/o recibe un vector que posee elementos que no sean numéricos, muestra un error en pantalla en vez de desplegar por pantalla el resultado de la raíz cuadrada de la suma de los 4 elementos de mayor valor, menos el resultado de la suma de la raíz cuadrada de los 4 elementos de menor valor. A continuación, se muestran tres ejemplos del uso de esta función además de 2 casos en los que la función mostraría un error en pantalla.

```
>> vector_sum_minus([1 5 6 1 23 6 6 1 2 6 35 1 2 6])
El resultado obtenido al restarle la raíz cuadrada de la suma de los 4 elementos
de menor valor, al resultado de la suma de la raíz cuadrada de los 4 elementos
de mayor valor es: 6.366600
```

Figura 5.4: Primer ejemplo del uso de la función que resta raíces cuadradas de los mayores y menores elementos del vector.

```
>> vector_sum_minus([-125 125125 125162 46967 346235 -12525 251269 7542 0])
El resultado obtenido al restarle la raíz cuadrada de la suma de los 4 elementos
de menor valor, al resultado de la suma de la raíz cuadrada de los 4 elementos
de mayor valor es: 920.755668
```

Figura 5.5: Segundo ejemplo del uso de la función que resta raíces cuadradas de los mayores y menores elementos del vector.

```
>> vector_sum_minus([1 1 1 1 1 1 1 1 1 1])
El resultado obtenido al restarle la raíz cuadrada de la suma de los 4 elementos
de menor valor, al resultado de la suma de la raíz cuadrada de los 4 elementos
de mayor valor es: 0.000000
```

Figura 5.6: Tercer ejemplo del uso de la función que resta raíces cuadradas de los mayores y menores elementos del vector.

```
>> vector_sum_minus([-125 125125 125])
ERROR: El vector debe tener al menos 8 elementos numéricos.
```

Figura 5.7: Ejemplo del error de tamaño de la función que resta raíces cuadradas de los mayores y menores elementos del vector.

```
>> vector_sum_minus([1 1 1 1 1 1 1 'a' 1 1])
ERROR: Hay uno o más elementos del vector que no son datos númericos.
```

Figura 5.8: Ejemplo del error de tipo de datos de la función que resta raíces cuadradas de los mayores y menores elementos del vector.

CAPÍTULO 6. CONCLUSIONES

MATLAB es una herramienta muy útil para el estudio de los distintos campos matemáticos de interés. En esta oportunidad parte de los objetivos se centraban en el poder utilizar esta herramienta como apoyo para graficar funciones de carácter sinusoidal y exponencial, lo cual llevo a resultados satisfactorios debido a que las dos funciones sinusoidales pudieron ser representadas en distintos planos y en uno a la vez, distinguiéndose entre sí debido a las distintas configuraciones de visualización que provee MATLAB. Por otro lado, la función exponencial dada, además de ser efectivamente representada en el plano con escala normal, se pudo concretar satisfactoriamente la representación de esta a través de una escala logarítmica, que gracias a la versatilidad de MATLAB no fue necesaria la implementación de un script que realizara la transformación a dicha escala, sino que el propio MATLAB proporciona una función nativa que puede realizar este procedimiento, con lo cual al tener dos gráficos de la misma función genérica con distinta escala, lleva plantearse casos hipotéticos en los cuales estas distintas escalas son útiles según información que se desea obtener. MATLAB provee en si versatilidad en cuanto a recursos matemáticos de distinta índole se refiere, en esta primera instancia los resultados fueron satisfactorios, ya que de manera muy simple se consiguió llegar a los objetivos.

Por otra parte, en esta experiencia también se logra obtener una raíz aproximada de un polinomio cualquiera utilizando el método Newton-Raphson, cuya implementación en MATLAB se realiza de manera recursiva, por lo que en cada uso de esta función se utiliza la última aproximación obtenida para acercarse cada vez más a una de las raíces del polinomio, deteniéndose este algoritmo cuando se la nueva aproximación calculada se encuentra bajo un error dado en comparación a la aproximación anterior.

Finalmente se implementa una función simple que imprime en pantalla el resultado de la raíz cuadrada de la suma de los 4 elementos de mayor valor, menos el resultado de la suma de la raíz cuadrada de los 4 elementos de menor valor, con el objetivo de verificar si es que la entrada de esta función es efectivamente un vector de datos numéricos. En esta función se logra comprobar que todos los datos del vector sean números y que haya suficientes datos (en este caso 8) para poder realizar sin errores la resta de raíces cuadradas.

CAPÍTULO 7. BIBLIOGRAFÍA

[1] F. Muñoz y G. Acuña, *Laboratorio 1 Introducción a MATLAB*, 2018. dirección: http://www.udesantiagovirtual.cl/moodle2/.

[2] J. C. Gorostizaga, *El método de Newton-Raphson (para hallar raíces de una ecuación f(x)=0)*, 2011. dirección: http://www.ehu.eus/juancarlos.gorostizaga/mn11b/temas/newton_ecuac.pdf.

[3] A. Smit, *Program for Newton Raphson Method*, 2016. dirección: https://www.geeksforgeeks.org/program-for-newton-raphson-method/.

[4] U. de Sevilla, *Escalas y gráficas logarítmicas*, 2013. dirección: http://laplace.us.es/wiki/index.php/Escalas_y_gr%C3%A1ficas_logar%C3%ADtmicas.