

MODELACIÓN Y SIMULACIÓN: LABORATORIO 1
INTRODUCCIÓN A MATLAB

PABLO CÁCERES LUZANTO
BENJAMÍN HERNÁNDEZ CORTÉS

Profesor:
Gonzalo Acuña
Ayudante:
Francisco Muñoz

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS.....	v
ÍNDICE DE CUADROS	vi
CAPÍTULO 1. INTRODUCCIÓN.....	7
1.1 OBJETIVOS ESPECÍFICOS	7
1.2 ESTRUCTURA DEL INFORME	7
CAPÍTULO 2. MARCO TEÓRICO.....	9
2.1 MÉTODO DE NEWTON-RAPHSON	9
2.2 ESCALA LOGARÍTMICA	9
CAPÍTULO 3. DESARROLLO	11
3.1 Parte 1	11
3.1.1 Gráfico 1: $a(x)$	11
3.1.2 Gráfico 2: $b(x)$	11
3.1.3 Gráfico 3: $a(x)$ y $b(x)$	11
3.1.4 Gráfico 4: $c(x)$	13
3.1.5 Gráfico 5: $c(x)$ en escala logarítmica	13
3.2 Parte 2	15
3.2.1 Método de Newton-Raphson Recursivo	15
3.2.2 Algoritmo	16
CAPÍTULO 4. MANUAL DE USO.....	19
4.1 REQUISITOS PREVIOS	19
4.2 PRIMERA PARTE	19
4.3 SEGUNDA PARTE	20
4.3.1 Ejemplo 1: Solución con el método de Newton-Raphson	20
4.3.2 Ejemplo 2: Solución alternativa con el método de Newton-Raphson	20
4.3.3 Ejemplo 3: Operación matemática con vectores	21

CAPÍTULO 5. CONCLUSIONES	23
CAPÍTULO 6. BIBLIOGRAFÍA.....	25

ÍNDICE DE FIGURAS

3.1	Gráfica de la función $a(x) = 3 \log_5(3x + 1)$ con $x \in [0, 15\pi]$	12
3.3	Gráfica de la función $a(x)$ y $b(x)$ con $x \in [0, 15\pi]$	12
3.2	Gráfica de la función $b(x) = \sin(2(\log_2(x + 11))) + \cos(5(\log_6(3x + 27)))$ con $x \in [0, 15\pi]$	13
3.4	Gráfica de la función $c(x)$ con $x \in [-10, 10]$	14
3.5	Gráfica de la función $c(x)$ con $x \in [-10, 10]$ y en escala logarítmica	14
3.6	Método de Newton-Raphson recursivo implementado en <i>MATLAB</i>	15
3.7	Implementación en <i>MATLAB</i> del Ingreso del largo del vector	16
3.8	Implementación en <i>MATLAB</i> del Ingreso de los valores del vector	16
3.9	Implementación en <i>MATLAB</i> del algoritmo deseado	17
4.1	Ejecución primera parte del Laboratorio	19
4.2	Introducción de los datos y solución para el ejemplo 1	20
4.3	Introducción de los datos y solución para el ejemplo 2	21
4.4	Introducción de los datos y solución para el ejemplo 3	21

ÍNDICE DE CUADROS

CAPÍTULO 1. INTRODUCCIÓN

Hoy en día MATLAB (abreviatura de *Matrix Laboratory*) es un IDE de programación en su propio lenguaje M, el cual es muy utilizado principalmente para la manipulación de matrices, la representación gráfica de datos y funciones, y la implementación de distintos algoritmos para llevar a cabo cierto objetivo en particular.

De acuerdo a lo descrito anteriormente, es que en el presente informe se ahondará y se llevará a la práctica los conceptos fundamentales y que son necesarios conocer para llevar a cabo los objetivos planteados a continuación y en general, a lo largo de todo el curso de Modelación y Simulación.

1.1 OBJETIVOS ESPECÍFICOS

■ Primera parte

- Graficar por separado y en conjunto, las siguientes funciones:

- $a(x) = 3 \log_5(3x + 1)$

- $b(x) = \sin(2(\log_2(x + 11))) + \cos(5(\log_6(3x + 27)))$

- Graficar en escala normal y logarítmica la siguiente función y además realizar comparación de los gráficos generados, indicando ventajas y desventajas de cada uno:

- $c(x) = 2e^{x+10}$

■ Segunda parte

- Implementar el algoritmo de Newton-Raphson de forma recursiva.
- Crear un archivo *.m* que reciba como entrada un vector y despliegue 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. Debe manejar el ingreso erróneo de los valores del vector y de la cantidad de elementos del vector.

1.2 ESTRUCTURA DEL INFORME

Este documento está conformado por seis capítulos, el primero y actual es el referente a la **Introducción** del informe. Luego, se presenta un breve **Marco Teórico**, en donde se repasan ciertos conceptos que son necesarios conocer al momento de realizar la experiencia y leer este documento. A continuación, se presenta el capítulo de **Desarrollo** en donde se explica, en el caso de la Parte 1, el desarrollo de cada uno de los gráficos, y en el caso de la Parte 2, la implementación de cada uno de los algoritmos. Posterior al capítulo de Desarrollo, se presenta un **Manual de Uso**, en donde -como su nombre lo indica- se presenta un breve

manual de lo desarrollado en la experiencia, dando al menos tres ejemplos de uso de la Parte 2 del Laboratorio. Finalmente, en el capítulo final se presentan las **Conclusiones** que se obtienen luego del desarrollo de la experiencia y acorde a los objetivos planteados inicialmente. Además, se presentan todas las referencias expuestas a lo largo del texto en la **Bibliografía**.

CAPÍTULO 2. MARCO TEÓRICO

2.1 MÉTODO DE NEWTON-RAPHSON

Es un método eficiente para encontrar aproximaciones de las raíces de una función real. Además, es abierto, lo que en sencillas palabras significa que no está asegurada su convergencia global. La única forma de alcanzar la convergencia es seleccionar un valor lo suficientemente cercano a la raíz deseada. Es importante mencionar que este método tiene dificultades cuando la función tiene pendientes grandes o múltiples puntos de inflexión cerca de la raíz, lo que trae como consecuencia un aumento de probabilidad de que no converja a una solución [1].

La fórmula de iteración utilizada para obtener un valor aproximado de las raíces de una función real para este método es la siguiente.

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

Donde se van calculando los siguientes números con la ecuación 2.1 hasta tener la precisión deseada o, en otras palabras, hasta que el error ya sea tolerable de acuerdo a la tolerancia establecida previamente.

2.2 ESCALA LOGARÍTMICA

La escala logarítmica es una escala de medida que se utiliza para representar más cómodamente magnitudes físicas, empleando el logaritmo de las magnitudes (con base decimal o exponencial) en lugar de la propia magnitud. Este tipo de escala de medida resulta útil cuando se maneja un amplio rango de valores, debido a que los logaritmos crecen muchos más despacio que los valores en sí [2].

Si se utiliza en un eje de coordenadas una escala logarítmica y en el otro eje una escala aritmética se dice que se está en presencia de un sistema de referencia semilogarítmico. Si en ambos ejes se utiliza escalas logarítmicas, el sistema de referencia es logarítmico.

CAPÍTULO 3. DESARROLLO

Antes de iniciar la explicación del desarrollo de las gráficas, es importante recordar que funciones son utilizadas para obtener cada una de ellas. Para la Parte 1, se utilizan las funciones: $a(x) = 3 \log_5(3x + 1)$, $b(x) = \sin(2(\log_2(x + 11))) + \cos(5(\log_6(3x + 27)))$ y $c(x) = 2e^{x+10}$.

3.1. Parte 1

3.1.1. Gráfico 1: $a(x)$

Para la obtención de la primera gráfica, en primer lugar, se define el dominio de la función, que tal como se solicita en el enunciado del laboratorio, debe ser entre 0 y 15π con una separación entre cada punto de 0,01. Una vez definido el dominio de la función se define la función propiamente tal en el dominio establecido previamente. Luego, se procede a graficar la función en sí en el intervalo deseado y además, se configuran las distintas opciones para el gráfico, que se solicitan en el enunciado, a saber: color de la función en rojo, marcador asterisco (*), título del gráfico y etiquetas en los ejes coordenados. Una vez realizado todo lo anterior, se obtiene la figura 3.1.

3.1.2. Gráfico 2: $b(x)$

Para la obtención de la segunda gráfica, se utiliza el mismo dominio de la función definido en el punto anterior (x entre 0 y 15π). Una vez definido el dominio de la función se define la función propiamente tal en el dominio establecido previamente. Luego, se procede a graficar la función en sí en el intervalo deseado y además, se configuran las distintas opciones para el gráfico, que se solicitan en el enunciado, a saber: color de la función en verde, marcador cruz (+), título del gráfico y etiquetas en los ejes coordenados. Una vez realizado todo lo anterior, se obtiene la figura 3.2.

3.1.3. Gráfico 3: $a(x)$ y $b(x)$

Para este gráfico, simplemente se mantienen las gráficas anteriores en un mismo gráfico, se cambian las etiquetas de los ejes coordenados según corresponda, en conjunto con el título del gráfico y además se añade una leyenda indicando, a través del color y marcador, que función es cada una de las exhibidas en el gráfico, resultando lo expuesto en la figura 3.3.

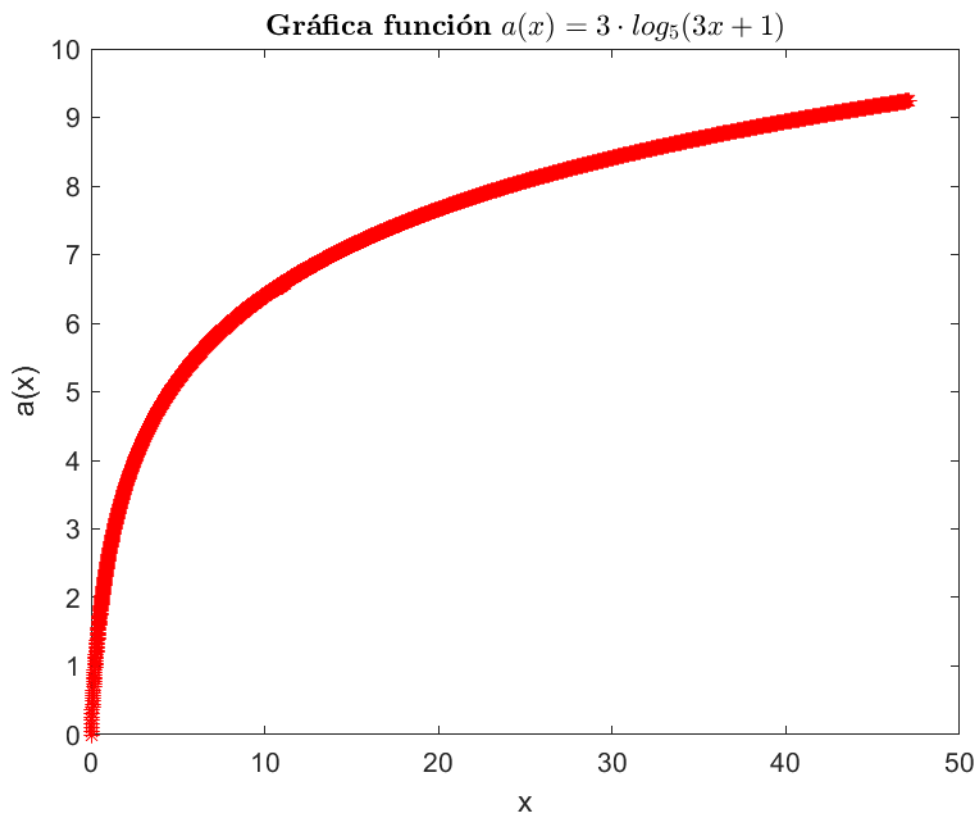


Figura 3.1: Gráfica de la función $a(x) = 3 \log_5(3x + 1)$ con $x \in [0, 15\pi]$

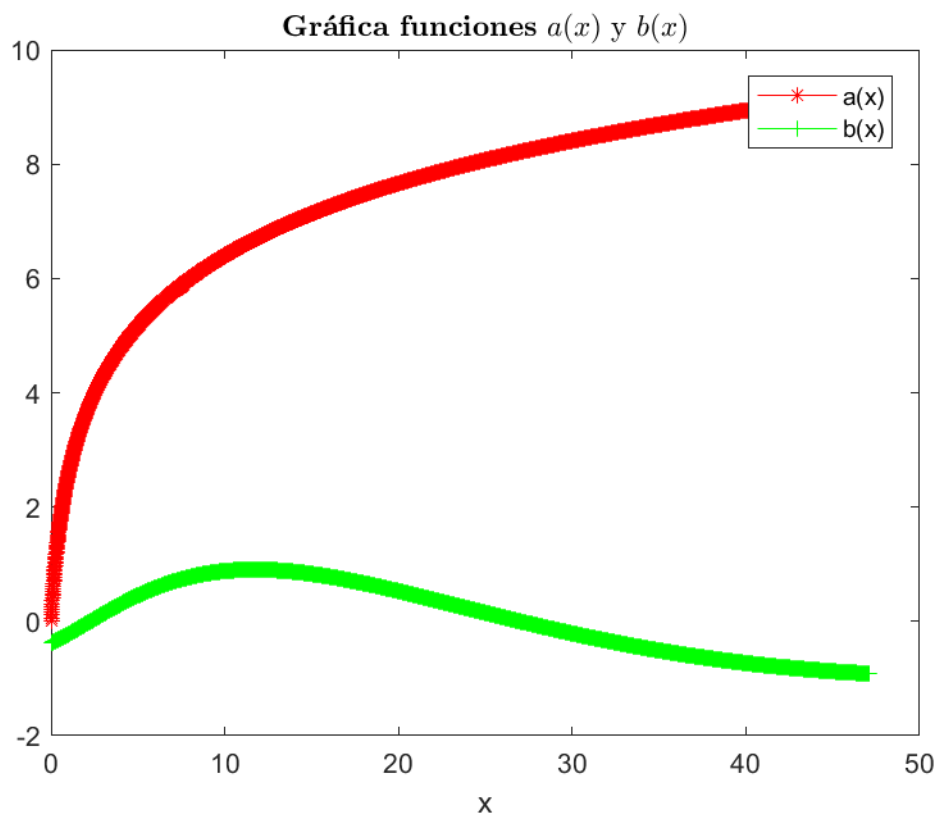


Figura 3.3: Gráfica de la función $a(x)$ y $b(x)$ con $x \in [0, 15\pi]$

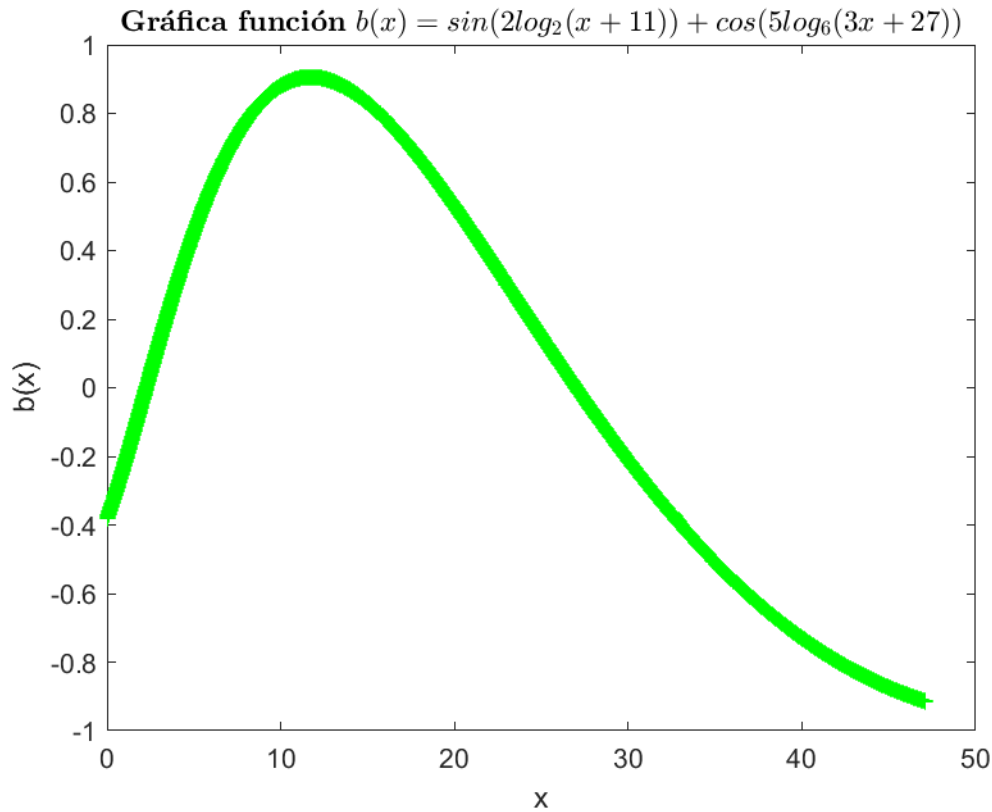


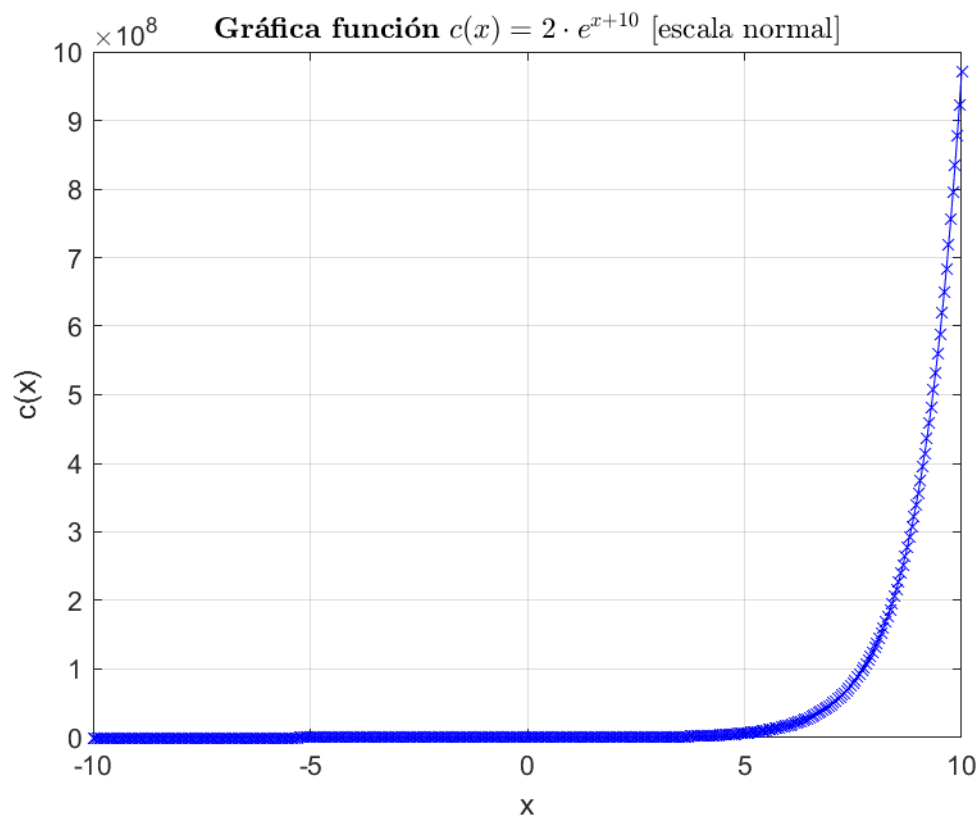
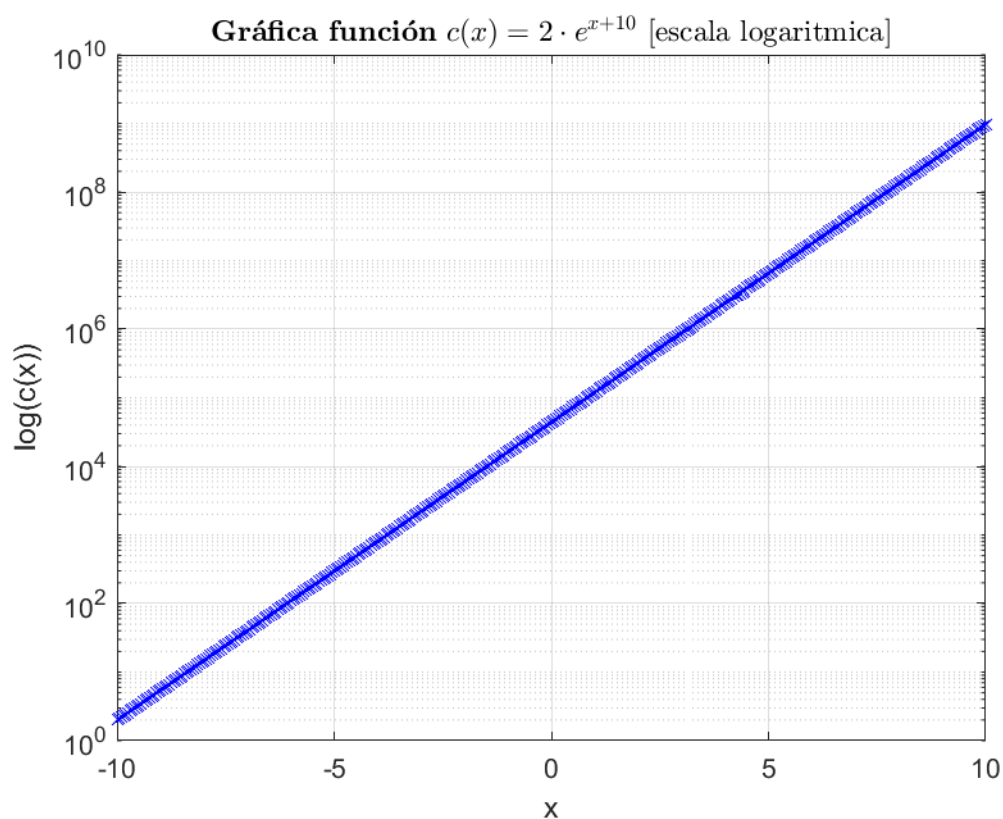
Figura 3.2: Gráfica de la función $b(x) = \sin(2(\log_2(x + 11))) + \cos(5(\log_6(3x + 27)))$ con $x \in [0, 15\pi]$

3.1.4. Gráfico 4: $c(x)$

Para el cuarto gráfico se define el dominio de la función, el cual para esta oportunidad debe ser manejado dentro del intervalo $[-10, 10]$ con una separación entre cada punto de 0,05. Una vez definido el dominio de la función se define la función propiamente tal en el dominio establecido previamente. Luego, se procede a graficar la función en sí en el intervalo deseado, en dónde se solicita el cuadriculado de la figura (presencia de grilla), mientras que el color de la función y el estilo del marcador quedan a elección, empleándose en esta oportunidad un marcador de cruz (x) y un color de línea azul. Además, se configura el título del gráfico y etiquetas en los ejes coordenados. Una vez realizado todo lo anterior, se obtiene la figura 3.4.

3.1.5. Gráfico 5: $c(x)$ en escala logarítmica

El último gráfico solicitado presenta casi las mismas características e indicaciones señaladas en el cuarto gráfico, con la diferencia de que éste se debe graficar empleando una escala logarítmica. Una vez realizado el cambio de escala, se obtiene la figura 3.5

Figura 3.4: Gráfica de la función $c(x)$ con $x \in [-10, 10]$ Figura 3.5: Gráfica de la función $c(x)$ con $x \in [-10, 10]$ y en escala logarítmica

Como se logra apreciar en la figura 3.4, los puntos marcado en el gráfico que van desde $x = -10$ hasta $x = 5$ se acoplan demasiado y debido a sus magnitudes (dada por la función) no es posible apreciar fácilmente los cambios presentados entre cada uno de los puntos, a diferencia del gráfico presentado en la figura 3.5, el cual muestra una distribución de los puntos totalmente lineal y fácilmente diferenciable. La principal ventaja que presenta el cuarto gráfico es la de representar el comportamiento “real” de una determinada función, al mantener las magnitudes en bruto. En cuanto a la quinta gráfica, su principal ventaja esta dada por la ‘normalización’ de los datos, a fin de mostrar de forma más clara cada uno de los puntos involucrados en la gráfica, sin embargo, se debe ser cautelosos al observar la gráfica, dado el correspondiente cambio en la escala de medida para el eje de las ordenadas ($c(x)$).

3.2. Parte 2

3.2.1. Método de Newton-Raphson Recursivo

Como toda función recursiva, este algoritmo necesita de al menos una condición de borde y un llamado recursivo a sí mismo. En esta oportunidad, las condiciones de borde escogidas tienen relación con el máximo error admitido del método, y el número de iteraciones del mismo. De esta forma, si la solución obtenida mediante uno de los llamados recursivos logra tener un error menor al máximo aceptado o supera el número máximo de iteraciones predefinido, entonces se detiene y se entrega el último valor de la solución (raíz) obtenido. Es importante recordar, que el calculo del valor de la solución o raíz aproximada se realiza utilizando la ecuación 2.1 expresada en el Capítulo 2. Todo lo descrito anteriormente, se puede visualizar de mejor forma en la figura 3.6

```
function resultado = newton_raphson(polinomio, max_iteraciones, error, x_n)

% Condición de borde 1: máximo de iteraciones alcanzado.
if max_iteraciones == 0
    resultado = x_n;
    return;
end

% Obtención de una aproximación a la raíz.
x_n1 = x_n - (polyval(polinomio, x_n) / polyval(polyder(polinomio), x_n));

% Condición de borde 2: error mínimo.
if abs(x_n1 - x_n) <= error
    resultado = x_n1;
    return;
else
    % Llamada recursiva.
    resultado = newton_raphson(polinomio, max_iteraciones - 1, error, x_n1);
    return;
end
```

Figura 3.6: Método de Newton-Raphson recursivo implementado en *MATLAB*

3.2.2. Algoritmo

Para explicar el desarrollo del algoritmo solicitado, este se divide en tres grandes secciones, las cuales corresponden a la petición del largo del vector a ingresar, la inserción de los valores del vector, y la realización del algoritmo propiamente tal una vez superadas las dos etapas anteriores. A continuación, se describen cada una de esta etapas.

- Respecto a la **petición de ingreso del largo del vector por parte del usuario**, se realizan básicamente tres validaciones: que el largo ingresado sea un valor numérico, que el largo ingresado sea un número entero, y que el largo del vector sea al menos de 4. La implementación de lo anterior se puede apreciar en la figura 3.7.

```

while true
    % Se pide el ingreso del largo del vector a generar.
    largo = input('Ingrese el largo del vector a generar con un numero entero (mayor o igual a 4): ', 's');
    % Se verifica que el largo ingresado sea un valor numérico.
    if isnan(str2double(largo))
        fprintf('ERROR: El dato ingresado no es numerico. Por favor, ingrese un valor numerico.\n');
    % Se verifica que el largo ingresado sea un número entero.
    elseif mod(str2double(largo), 1) ~= 0
        fprintf('ERROR: El dato ingresado no es un numero entero. Por favor, ingrese un numero entero.\n');
    % Se verifica que el largo del vector sea al menos 4.
    elseif str2double(largo) < 4
        fprintf('ERROR: El largo ingresado es menor que 4. Por favor, ingrese un largo mayor\n');
    else
        largo = str2double(largo);
        break;
    end
end
end

```

Figura 3.7: Implementación en *MATLAB* del Ingreso del largo del vector

- Luego, se solicita al usuario que **ingrese los valores del vector** y la única validación que se realiza es la de confirmar que los números ingresados sean realmente números y no caracteres de otro tipo. Su implementación se visualiza en la figura 3.8.

```

while i <= largo
    mensaje = join(['Ingrese el dato ', num2str(i), ': ']);
    dato = input(mensaje, 's');
    % Se verifica que el dato a ingresar en el vector sea numérico.
    if isnan(str2double(dato))
        fprintf('ERROR: El dato ingresado no es numerico. Por favor, ingrese un valor numerico.\n');
    else
        vector = [vector, str2double(dato)];
        i = i + 1;
    end
end
end

```

Figura 3.8: Implementación en *MATLAB* del Ingreso de los valores del vector

- Finalmente, respecto al **algoritmo** en sí, en primer lugar, se ordena el vector ingresado por el usuario de forma ascendente. Luego, se suman los cuatro valores de mayor cuantía del vector y se obtiene su raíz cuadrada. A continuación, se obtiene la raíz cuadrada de los cuatro menores valores del vector y se suman. Luego, se muestra por pantalla la diferencia de los resultados obtenidos de ambos grupos. Su implementación en *MATLAB* se aprecia en la figura 3.9.


```
% Se ordena el vector con los números de forma ascendente
vector = sort(vector, 'ascend');

% Se calculan los términos de la resta a realizar.
primer_termino = sqrt(sum(vector(end-3:end)));
segundo_termino = sum(sqrt(vector(1:4)));

%
% Visualización del resultado
%
fprintf('El resultado es %f\n', primer_termino - segundo_termino);
```

Figura 3.9: Implementación en *MATLAB* del algoritmo deseado

CAPÍTULO 4. MANUAL DE USO

4.1 REQUISITOS PREVIOS

Para asegurar que los códigos que se presentan funcionen correctamente, es deseable que se cumplan los siguientes requisitos antes de su ejecución: Utilizar la versión R2017b de MATLAB y el Sistema Operativo Windows 10.

4.2 PRIMERA PARTE

Lo único que se debe realizar para ejecutar el código de esta parte del Laboratorio es tener el archivo denominado *parte_1.m*. Luego, basta con abrir el archivo *parte_1.m* con MATLAB y ejecutarlo, ya sea haciendo clic en el botón en la barra superior de la interfaz que se aprecia en la figura 4.1, apretando la tecla *F5* o bien escribiendo en la misma ventana de comandos: *parte_1*. Posterior a eso, se empiezan a desplegar de forma automática los gráficos solicitados y los mismos serán guardados en el directorio donde se ubique el código.

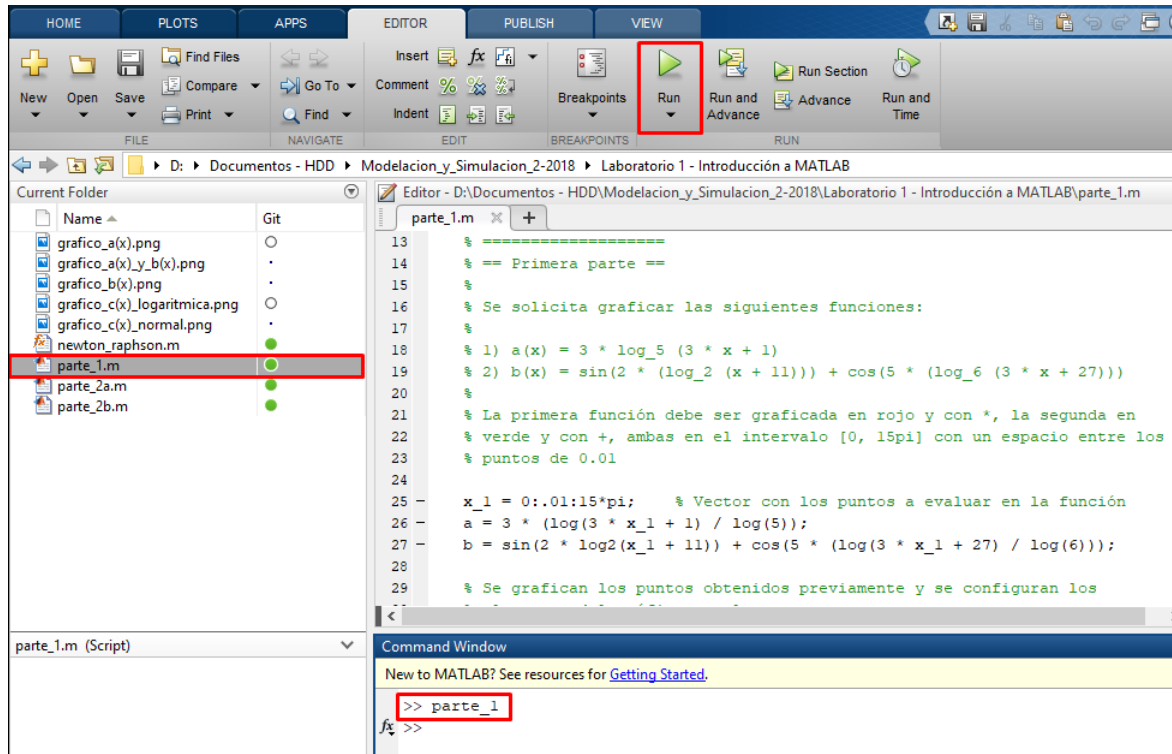


Figura 4.1: Ejecución primera parte del Laboratorio

4.3 SEGUNDA PARTE

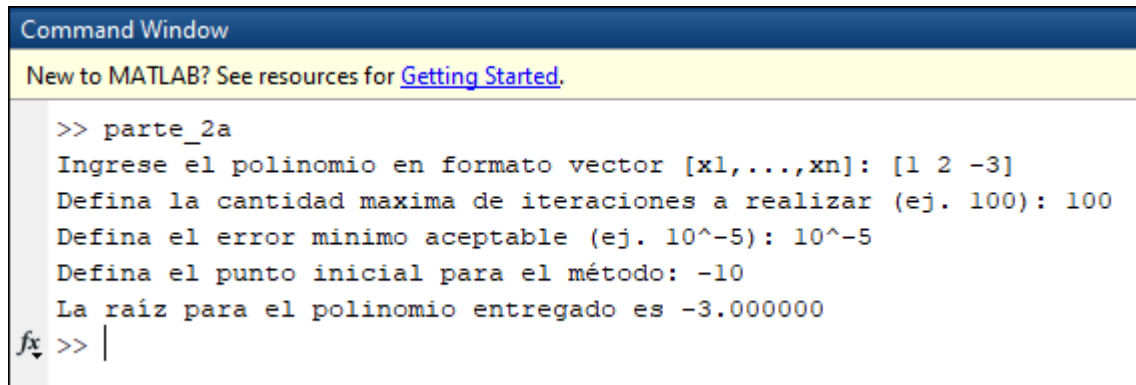
Para esta sección se presentan tres códigos: *parte_2a.m*, *parte_2b.m* y *newton_raphson.m*. El archivo *parte_2a.m* fue orientado e implementado para poner a prueba el método de Newton-Raphson desarrollado en el archivo *newton_raphson.m*, razón por la cual ambos archivos deben estar contenidos en el mismo directorio para funcionar de manera correcta. Por otro lado, el archivo *parte_2b.m* se encargará de realizar la operación matemática solicitada para un vector definido por el usuario.

4.3.1. Ejemplo 1: Solución con el método de Newton-Raphson

Para desarrollar este ejemplo, ejecute el archivo *parte_2a.m* siguiendo las mismas instrucciones que mencionadas en el manual de uso para la primera parte, es decir, haciendo clic en el botón **Run** en la barra superior de la interfaz, apretando la tecla *F5* o bien escribiendo en la misma ventana de comandos: *parte_2a*.

Para este ejemplo, se buscará la raíz del polinomio $x^2 + 2x - 3$. Una vez que ha ejecutado el código, siga las instrucciones que aparecen en la ventana de comandos (*Command Window*) e introduciendo los datos tal como se muestran en la figura 4.2, es decir:

- Polinomio en formato vector: $[1 \ 2 \ -3]$
- Cantidad máxima de iteraciones a realizar: 100
- Error mínimo aceptable: 10^{-5}
- Punto inicial: -10



```

Command Window
New to MATLAB? See resources for Getting Started.

>> parte_2a
Ingrese el polinomio en formato vector [x1,...,xn]: [1 2 -3]
Defina la cantidad maxima de iteraciones a realizar (ej. 100): 100
Defina el error minimo aceptable (ej. 10^-5): 10^-5
Defina el punto inicial para el método: -10
La raíz para el polinomio entregado es -3.000000
fx >> |
  
```

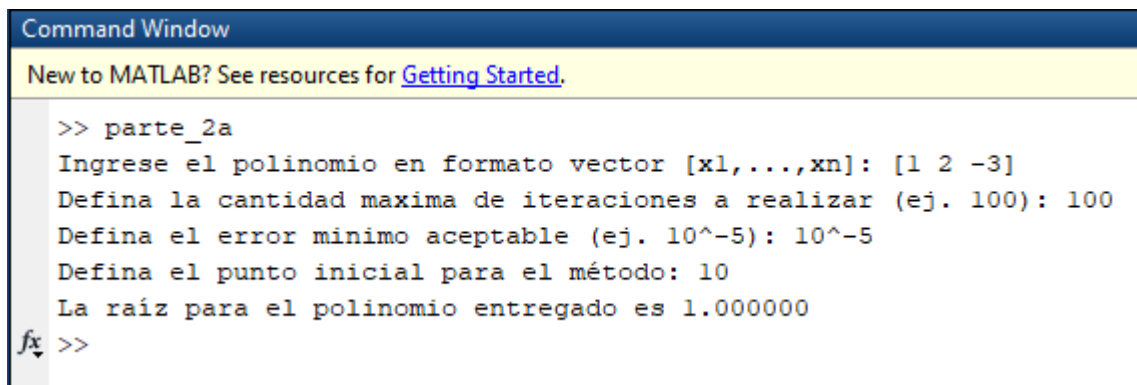
Figura 4.2: Introducción de los datos y solución para el ejemplo 1

Una vez introducida toda la información necesaria, el programa mostrará a través de la ventana de comandos la raíz obtenida para el polinomio definido, la cual para este ejemplo resulta en -3 y que efectivamente corresponde a una raíz del polinomio.

4.3.2. Ejemplo 2: Solución alternativa con el método de Newton-Raphson

Para este ejemplo, siga las mismas instrucciones indicadas en el ejemplo 1, con la salvedad de cambiar el valor del punto inicial de -10 a 10 . Una vez introducida la información, la raíz que se muestra a través de

la ventana de comandos corresponde al valor 1, tal como se aprecia en la figura 4.3. Este valor corresponde a otra solución para el mismo polinomio indicado en el ejemplo anterior y con esto, se trata de evidenciar que la solución ofrecida por el método de Newton-Raphson depende (en un cierto grado) del punto de partida escogido.

The screenshot shows the MATLAB Command Window with a dark blue title bar. Below the title bar is a yellow banner with the text "New to MATLAB? See resources for [Getting Started.](#)". The command window contains the following text: ">> parte_2a", "Ingrese el polinomio en formato vector [x1,...,xn]: [1 2 -3]", "Defina la cantidad maxima de iteraciones a realizar (ej. 100): 100", "Defina el error minimo aceptable (ej. 10^-5): 10^-5", "Defina el punto inicial para el método: 10", "La raíz para el polinomio entregado es 1.000000", and "fx >>".

```
Command Window
New to MATLAB? See resources for Getting Started.
>> parte_2a
Ingrese el polinomio en formato vector [x1,...,xn]: [1 2 -3]
Defina la cantidad maxima de iteraciones a realizar (ej. 100): 100
Defina el error minimo aceptable (ej. 10^-5): 10^-5
Defina el punto inicial para el método: 10
La raíz para el polinomio entregado es 1.000000
fx >>
```

Figura 4.3: Introducción de los datos y solución para el ejemplo 2

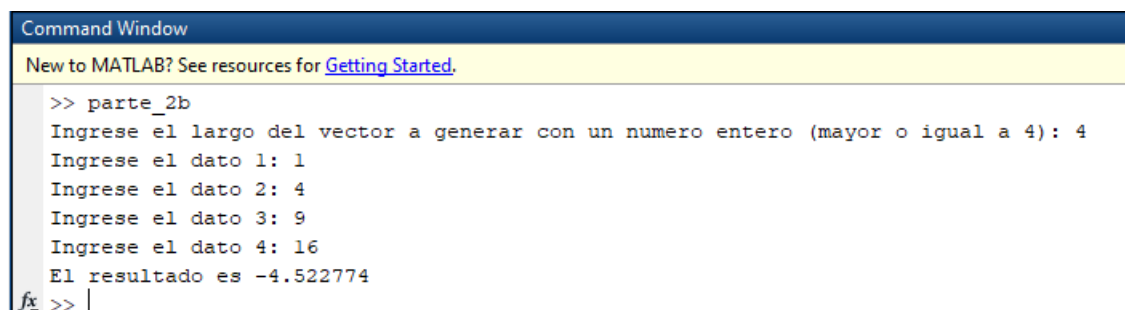
4.3.3. Ejemplo 3: Operación matemática con vectores

En este último ejemplo, ejecute el archivo *parte_2b.m* siguiendo las mismas instrucciones mencionadas en ejemplos anteriores, es decir, haciendo clic en el botón **Run** en la barra superior de la interfaz, apretando la tecla *F5* o bien escribiendo en la misma ventana de comandos: *parte_2b*.

A continuación, siga la instrucciones que aparecerán en la ventana de comandos. En esta ocasión, introduzca los siguientes datos de prueba tal como se aprecian en la figura 4.4:

- Largo del vector: 4
- Datos 1, 2, 3 y 4 respectivamente: 1, 4, 9 y 16

Una vez introducida la información necesaria, el programa mostrará en la ventana de comandos el resultado de la operación matemática realizada con el vector, que para este ejemplo resulta en el valor -4.522774 .

The screenshot shows the MATLAB Command Window with a dark blue title bar. Below the title bar is a yellow banner with the text "New to MATLAB? See resources for [Getting Started.](#)". The command window contains the following text: ">> parte_2b", "Ingrese el largo del vector a generar con un numero entero (mayor o igual a 4): 4", "Ingrese el dato 1: 1", "Ingrese el dato 2: 4", "Ingrese el dato 3: 9", "Ingrese el dato 4: 16", "El resultado es -4.522774", and "fx >>".

```
Command Window
New to MATLAB? See resources for Getting Started.
>> parte_2b
Ingrese el largo del vector a generar con un numero entero (mayor o igual a 4): 4
Ingrese el dato 1: 1
Ingrese el dato 2: 4
Ingrese el dato 3: 9
Ingrese el dato 4: 16
El resultado es -4.522774
fx >>
```

Figura 4.4: Introducción de los datos y solución para el ejemplo 3

CAPÍTULO 5. CONCLUSIONES

Respecto a los objetivos que se presentaron en el capítulo inicial del informe, es posible afirmar que se han cumplido todos y cada uno de ellos. En específico, en la primera parte de este Laboratorio, se aprendió o más bien se recordó cómo graficar distintos tipos de funciones y a personalizar las distintas graficas modificando ciertos parámetros a nivel de código en *MATLAB*, como lo son las escalas, las etiquetas de los ejes, entre otros.

En lo que respecta a la segunda parte del Laboratorio, se recordó e implementó el algoritmo de Newton-Raphson de forma recursiva, utilizando la fórmula presentada en el capítulo 2. Por otra parte, en la implementación del algoritmo solicitado, se tuvo que restringir y al mismo tiempo solicitar datos al usuario para que este algoritmo pudiera funcionar de manera correcta, siendo este uno de los desafíos más interesantes del Laboratorio.

Finalmente, se escribió un Manual de Usuario en el capítulo 4, en donde básicamente se muestra al usuario a través de figuras varias, ejemplos, etc. a utilizar y ejecutar cada uno de los programas escritos en *MATLAB*.

CAPÍTULO 6. BIBLIOGRAFÍA

- [1] I. L. Cañestro, *Método de Newton-Raphson*, 2018. dirección: <https://www.geogebra.org/m/XCrwWHzy> (visitado 05-11-2018).
- [2] J. L. Á. García, *Escalas logarítmicas*, 2017. dirección: http://geogebra.es/gauss/materiales_didacticos/eso/actividades/funciones/representaciones/escalas_logaritmicas/actividad.html (visitado 05-11-2018).