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

Laboratorio 1: Introducción a MATLAB

Integrantes:	Alberto Rodríguez
	Chun-zen Yu
Curso:	Modelación y Simulación
Sección	A-1
Profesor:	Gonzalo Acuña
Ayudante:	Alan Barahona

8 de Noviembre de 2020

Tabla de contenidos

1. Introducción

Actualmente la matemática es una herramienta muy útil para comprender fenómenos de la realidad y como predecirlos. Una forma de analizar el comportamiento de estos es a través de modelos, lo cuales facilitan el análisis.

Una de las herramientas utilizadas para este trabajo es MATLAB, el cual permite procesar grandes cantidades de datos, además de brindar simplicidad en funciones y operatorias básicas.

El objetivo principal es comprender como utilizar MATLAB en distintos temas, manipulando datos, realizar gráficos de funciones, aplicar escalas normal y logarítmica, aplicar el algoritmo de Newton Rhapson y hacer cálculos con vectores. Todo el desarrollo de lo anterior será explicado con detalle, además de incluir un manual de usuario de como el usuario puede ingresar los datos.

2. Marco teórico

2.1. Método de Newton Raphson

El método de Newton-Raphson es un método numérico abierto utilizado para encontrar aproximaciones de los ceros o raíces de una función no lineal. También puede ser usado para encontrar el máximo o mínimo de una función. El resultado de esta aproximación depende directamente del numero de iteraciones que se realice y el umbral de error que aceptara, habitualmente entre mas iteraciones el resultado es mas cercano a la raíz. El método se puede ver representado gráficamente en la figura 1.

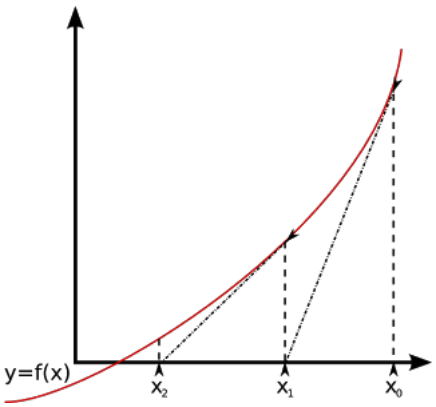

Figura 1: Expresión gráfica método Newton Raphson.

Este método se deriva de la serie de Taylor, donde se aproxima una función por medio de la serie y esa serie se iguala a cero, para encontrar la aproximación a la raíz se despeja de la serie truncada de primer orden.(Villanueva, 1998) Además se utiliza un x_0 el cual corresponde al valor inicial con la que se hace la primera aproximación.

$$0 = f(x) = f(x_0) + f'(x_0)(x - x_0) + ...$$

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Figura 2: Expresión matemática de método Newton Raphson.

2.2. Escala logarítmica

La escala logarítmica consiste en una escala de medida que utiliza el logaritmo del valor en vez del valor propio. Esta escala presenta un comportamiento no lineal debido a que los números 10, 100 y 1000 se encuentran a la misma distancia, es decir moverse una unidad en la escala logarítmica significa que el valor ha sido multiplicado por alguna factor dependiendo de la ubicación en la escala.

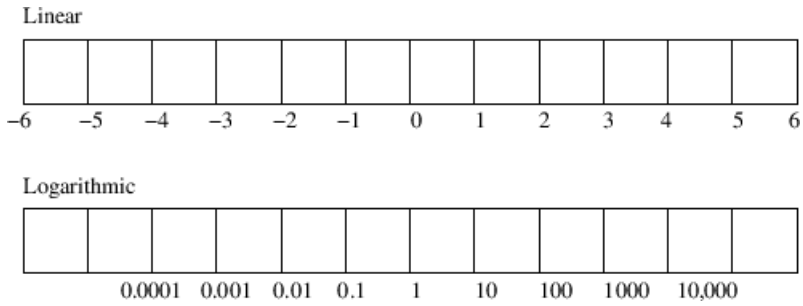

Figura 3: Comparación escala lineal con logarítmica.

Debido a la naturaleza de la escala los valores mas grandes se encuentran comprimidos a comparación de los valores mas pequeños, esto permite que se pueda abarcar un rango mas amplio de datos en un espacio reducido.

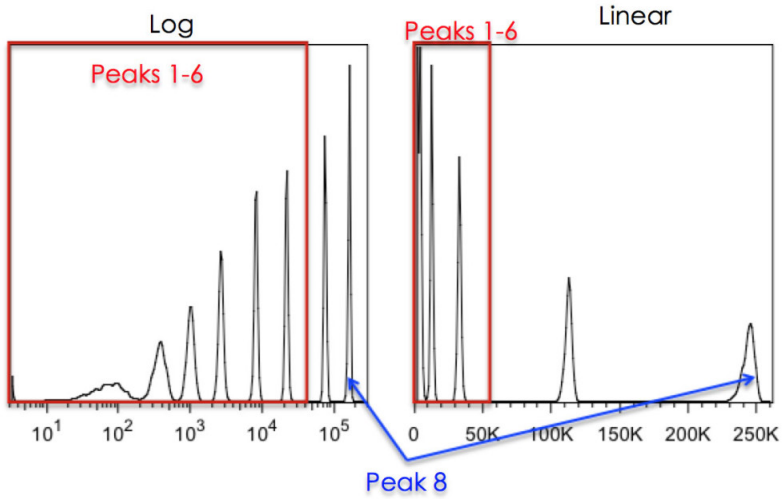

Figura 4: Comparación gráfico lineal con gráfico logarítmico.

3. Desarrollo

3.1. Primera Parte

Se realiza los gráficos de las siguientes funciones:

$$a(x) = 8log_5(4x + 12) \tag{1}$$

$$b(x) = sen(6(log_2(x + 9))) + cos(7(log_6(4x + 32))) \tag{2}$$

Ambas son graficadas en el intervalo [0,15π] con un espacio entre ellos de 0.01, siendo la primera graficada en rojo con varios ´´*", la segunda en verde con varias ´´+".

Primero se definen los intervalos del vector, y este se multiplica por la función creada, para poder crear el gráfico (figura 5).

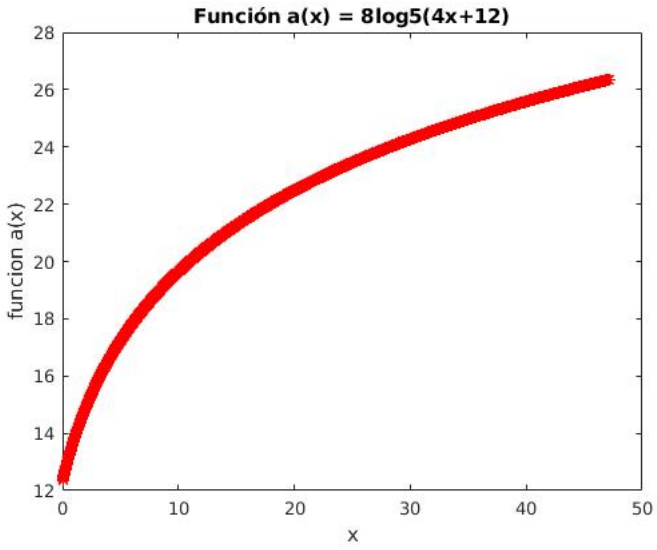

Figura 5: Función a(x).

Para la implementación correcta de los logaritmos, se usa la propiedad de cambio de base, porque MATLAB solo tiene logaritmos en base 10. (figura 6)

Uniendo las dos funciones, el gráfico en conjunto se ve de la siguiente forma (figura 7)

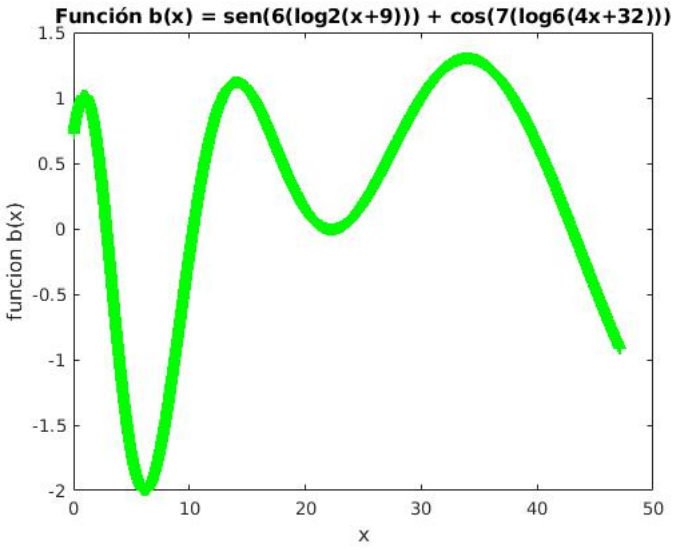

Figura 6: Función b(x).

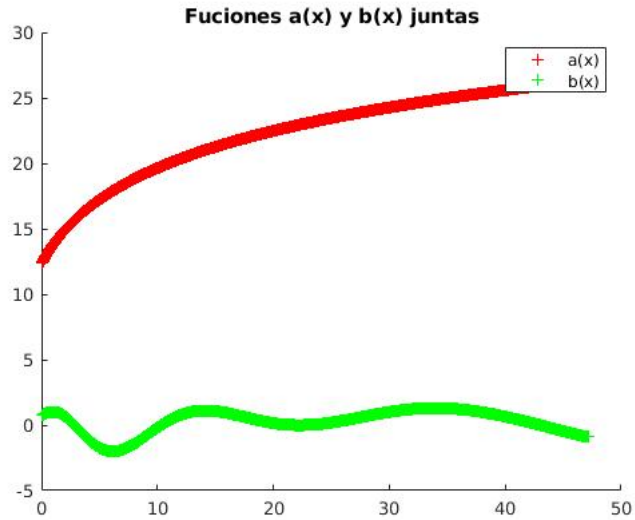

Figura 7: Funciones a(x) y b(x)

Lo siguiente a realizado es la comparación de escala normal y logarítmica en el intervalo [-10,10] con espaciado de 0.05 para la función:

$$c(x) = 6e^{x+18} \tag{3}$$

La escala logarítmica (figura 9) sirve para representar un rango mas amplio a comparación de la escala normal (figura 8) en un espacio mas reducido. En la escala logarítmica

se ve una mejor perspectiva el desarrollo de una función exponencial, en la escala normal no se aprecia esto, al ser una curva que crece rápidamente en un cierto punto.

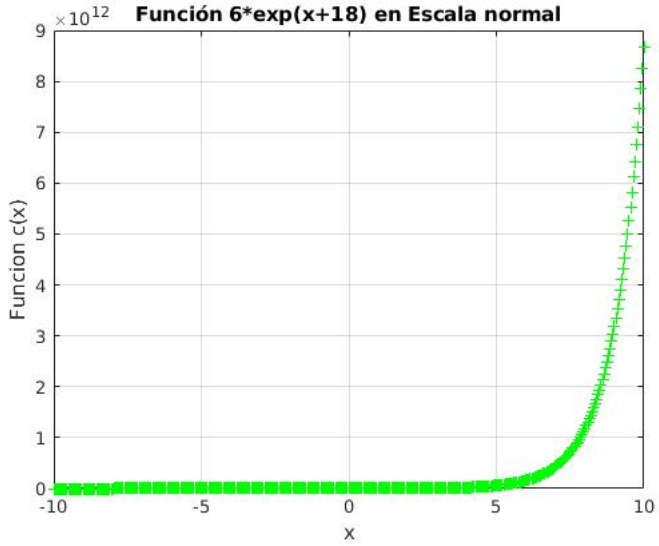

Figura 8: Función c(x), Escala Normal

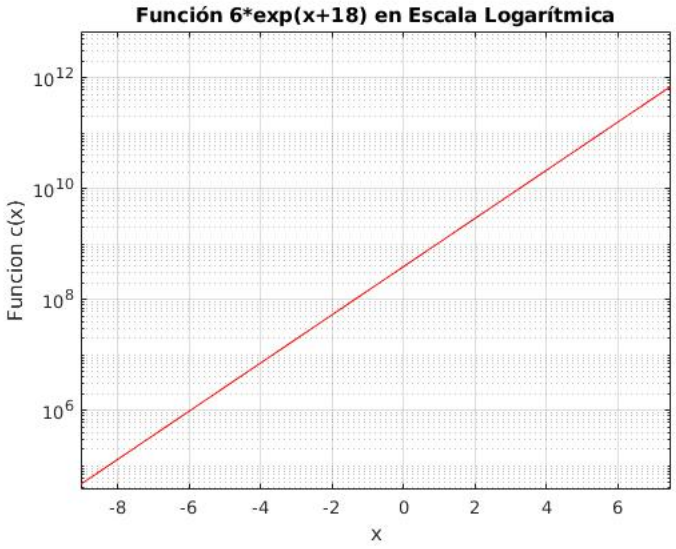

Figura 9: Función c(x), Escala Logarítmica

3.2. Segunda Parte

3.2.1. Newton Raphson

Para la resolución del método de Newton Raphson se utilizo la siguiente expresión:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{4}$$

Se define un valor inicial x_0, la cantidad de iteraciones y el umbral de error. A continuación se evalúa x_0 en la ecuación para obtener x_1, después se verifica si el valor de $f(x_1)$ es menor que el umbral de error definido. Si es así se termina la iteración, de lo contrario se vuelve a evaluar x_1 de la misma manera hasta que se encuentre un valor que caiga dentro de los valores permitidos o se termine el numero de iteraciones.

En MATLAB se define la función newtonRaphson() la cual sera encargada de resolver el método con un procedimiento recursivo, esta función tendrá los parámetros de entrada *polinomio*, *iteraciones*, *error* y *x0*, cada una correspondiente con los valores definidos anteriormente. Para evaluar las raíces en los polinomios se utilizo *polyval* y para derivar los polinomios se utilizo la función *polyder*, ambas de estas funciones se encuentran disponibles en librerías de MatLab. El código se puede observar en la figura 10.

```
function [valorX] = newtonRaphson(polinomio,iteraciones,error,x0)
    if(iteraciones==0) %Condicion inicial
        valorX =x0;
    else
        y= polyval(polinomio,x0); %Se evaluan los puntos en la funcion polinomial
        derPol = polyder(polinomio); %Calculo la derivada del polinomio
        derivada= polyval(derPol,x0); %Obtengo los valores de la derivada del polinomio
        x1 = x0 - (y/derivada); %Aplico funcion de newton-rhapson
        y1 = polyval(polinomio,x1);
        if(y1<error)
            valorX=x1;
        else
            valorX = newtonRaphson(polinomio,iteraciones-1,error,x1);
        end
    end
end
```

Figura 10: Código MatLab Newton Raphson.

3.2.2. Resta de vectores

Para la próxima sección se utiliza la función maxk y mink, ambas funciones son utilizadas para encontrar los 4 valores mas grandes y mas pequeños del vector respectivamente, además se utiliza isnumeric para verificar la entrada hacia la función. El código de la función se puede observar en la figura 11.

```
vector = input('Ingrese vector: ');

check = isnumeric(vector);
if check == 1
    vectorSize = size(vector);
    if vectorSize(1) == 1 && vectorSize(2) > 4
        mayor = maxk(vector,4);
        menor = mink(vector,4);
        resultado = sqrt(sum(mayor)) - sqrt(sum(menor))
    else
        disp("El vector debe ser mayor a 4");
    end
else
    disp("El vector no fue ingresado de manera correcta");
end
```

Figura 11: Código MatLab resta de vectores.

4. Manual de uso

Esta sección da a explicar como usar el programa para utilizar el método de Newton Raphson y el archivo "parte2b.m" que permite hacer la raíz cuadrada de los 4 números de menor valor.

Para la ejecución del código primero se tiene que asegurar que estén todos los archivos .m necesarios, luego tiene que abrir MATLAB y dirigirse al directorio donde se encuentran los archivos, a continuación debe hacer clic en el archivo que desea ejecutar y presionar *run* para comenzar su ejecución, somos se demuestra en la figura 12.

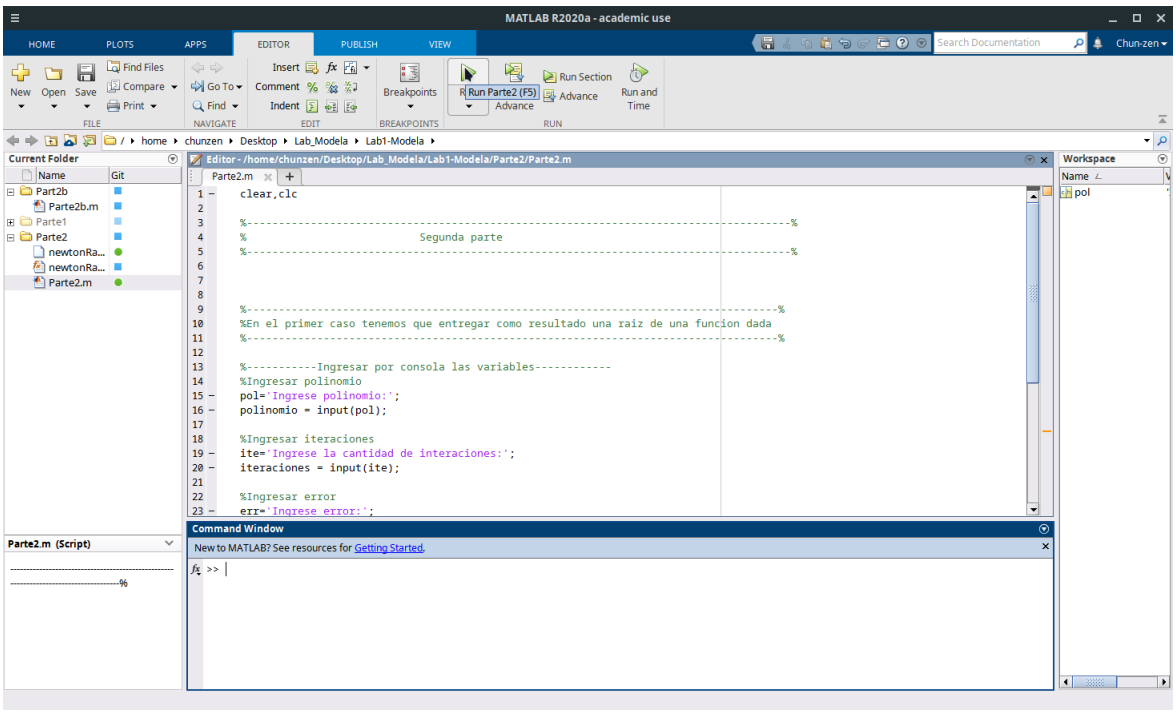

Figura 12: Ejemplo de ejecución de "Parte2.m"

Al ejecutar el código la sección inferior de *Command Window* solicitara los valores necesario para su funcionamiento y se mostrara por pantalla los resultados de la ejecución.

4.1. Primer ejemplo Newton Raphson

Al ejecutar el archivo "Parte2.m"la ventana *Command Window* comenzara a solicitar los valores necesarios para resolver un polinomio de Newton Raphson, los valores solicitados serán: polinomio, cantidad de iteraciones, error y valor inicial.

Figura 13: Ejecución de "Parte2.m"

En este ejemplo se ingresa el polinomio $1 + 2x + 3x^2 + 4x^3$, un limite de 50 iteraciones, un umbral de error del 0.002 y un valor inicial de $x_0 = 4$

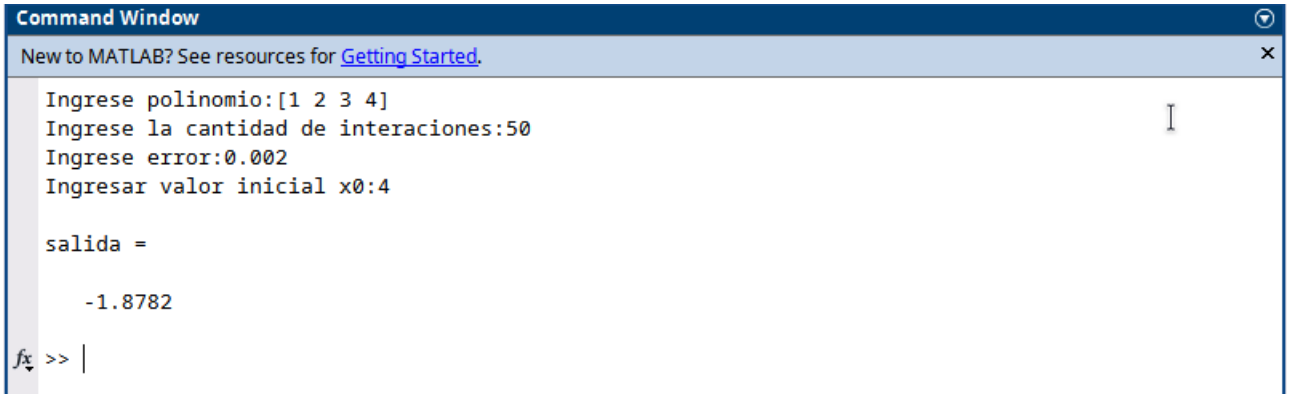

Figura 14: Resultado obtenido de archivo "Parte2.m"

Una vez terminado la ejecución se mostrara el resultado del método en la ventana *Command Window*.

4.2. Primer ejemplo raíces cuadradas

Al ejecutar el código "Parte2b.m"la consola *Command Window* solicitara ingresar el vector el cual se desea operar.

Figura 15: Ejecución de "Parte2b.m"

En la figura 16 se ingresa el vector $(200, 30, 40, 103, 9, 3, 6, 4, 6)$

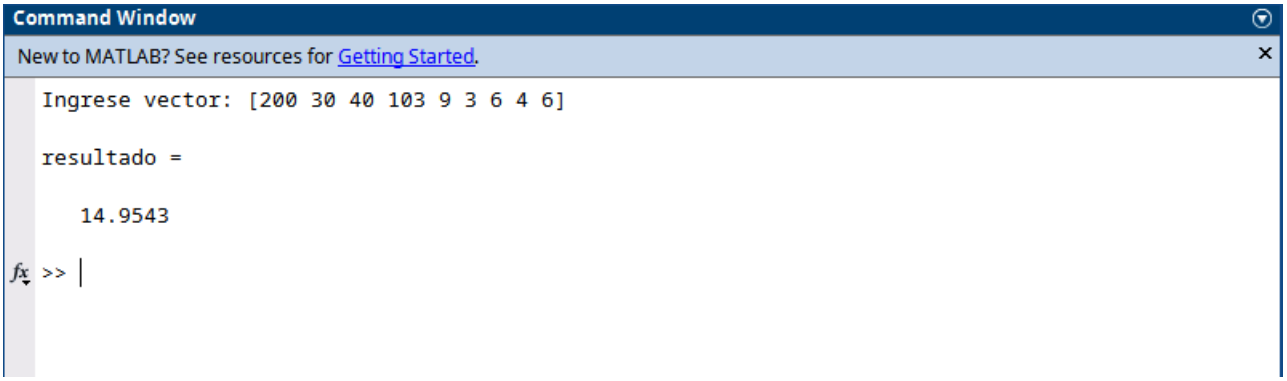

Figura 16: Resultado de "Parte2b.m"

Al finalizar se muestra por *Command Window* el resultado final obtenido.

4.3. Segundo ejemplo raíces cuadradas

En el siguiente ejemplo se ingresa un vector de $(1, 20, 3)$

Figura 17: Resultado de "Parte2b.m"

La consola retorna *El vector debe ser mayor a 4*, esto se debe a que se necesita un vector que contenga por lo menos 4 valores para el funcionamiento correcto del programa.

5. Conclusiones

Con los resultados obtenidos en las secciones anteriores se puede apreciar todas las herramientas que MATLAB te puede brindar. Uno de estos son los gráficos, donde estos se muestran detalladamente los tipos de escala, regiones se podrían marcar para evaluar. Se pueden diferenciar las funciones graficadas si se quiere mostrar múltiples curvas en un mismo plano.

Al realizar el método de Newton-Raphson podemos mencionar que es una buena alternativa para realizar el cálculo de las raíces de la función. Es un algoritmo que es mas rápido que otros en ciertas condiciones y es eficientes para ecuaciones no lineales (como lo polinomios de prueba que se mostraron en el manual de usuario). Sin embrago, es necesario conocer la primera derivada de la función, lo que para algunos casos puede ser complejo o lento calcular esta, lo que puede provocar que el desarrollo del algoritmo sea mas lento, al ir calculando muchas derivadas complejas. Considerar que al ser un algoritmo recursivo, si es que las iteraciones son muchas o si el error permitido es muy pequeño, puede consumir mucho los recursos del computador, por lo tanto tendrá algún límite.

Finalmente podemos concluir que los objetivos del laboratorio fueron cumplidos con éxito, ya que se logró representar gráficos en MATLAB, estas incluyen funciones en escala normal y logarítmica. Se aprendió a implementar un algoritmo de búsqueda de raíces como lo es Newton-Raphson, el cual se realizó de forma recursiva, junto con implementar un algoritmo para obtener una solución en base a determinados valores de un vector.

Bibliografía

Villanueva, W. D. (1998). *Newton-Raphson*.