# UNIVERSIDAD DE SANTIAGO DE CHILE

## FACULTAD DE INGENIERÍA

## Departamento de Informática



# Modelación y Simulación

# Laboratorio 1


**Juan Arredondo**

**Flavio Ramos**

Profesor: Gonzalo Acuña

Ayudante: Alan Barahona



Santiago – Chile

2021

# TABLA DE CONTENIDO

# ÍNDICE DE TABLAS

# ÍNDICE DE ILUSTRACIONES

# CAPÍTULO 1. INTRODUCCIÓN

En el presente documento se presentan los resultados obtenidos de la implementación del laboratorio uno del curso de Modelación y Simulación, en el cual se trabajan algunos aspectos del lenguaje de programación Matlab mediante la implementación de gráficos y funciones que hacen uso de características del lenguaje.

## 1.1 OBJETIVOS

### 1.1.1 Objetivo general

Acercamiento al lenguaje de programación Matlab, mediante la realización de diferentes actividades, involucrando el desarrollo de gráficos y funciones, con el uso de las herramientas integradas del lenguaje.

### 1.1.2 Objetivos específicos

- Graficar funciones en diferentes escalas usando MatLab.
- Implementar algoritmo de Newton-Raphson para un polinomio de una variable.
- Implementar función en la cual se manejen ingresos erroneos de parámetros.

## 1.2 ORGANIZACIÓN DEL DOCUMENTO

Este documento se divide en tres partes principales, la primera consta de graficar diferentes funciones en distintas escalas haciendo uso de las características de Matlab. Por otro lado, la segunda parte consta de la implementación del algoritmo de Newton-Raphson, donde la resolución de funciones debe ser de forma recursiva. La tercera parte, debe crear una función

que dependiendo la entrada logre retornar la desviación estándar, donde no se puede hacer uso de las funciones que entrega Matlab.

Finalmente, se debe crear un manual con ejemplos de uso de los programas implementados, en que se demuestre de forma sencilla el funcionamiento.

# CAPÍTULO 2. MÁRCO TEÓRICO

## 2.1 POLINOMIOS

En la formulación abstracta de problemas matemáticos se recurre con frecuencia a las llamadas expresiones algebraicas, que no son sino un conjunto de letras y números unidos entre sí por signos aritméticos. Los polinomios son casos especiales de expresiones algebraicas donde las variables o indeterminadas aparecen siempre elevadas a un exponente positivo y entero. Además, se denomina polinomio a una expresión algebraica que está definida como la suma de cada uno de los términos de éste, los cuales corresponden a monomios.

### 2.1.1 Elementos de un polinomio

Los términos de un polinomio tienen diferentes elemenos que los identifican y diferencian de otros, para la expresión general de un polinomio *P(x)* de una sola variable:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_2 x^2 + a_1 x + a_0$$

Los elementos del polinomio son:

- **Coeficientes**: Son los valores constantes $a_i$ con $i = 0, 1, 2, ...., n,$ donde el llamado coeficiente principal (denotado por $a_n$) acompaña a la variable de mayor exponente, mientras que el que no tiene variable asociada es llamado término independiente (denotado por $a_0$).

- **Variables**: En este caso la $x$, la cual representa un valor genérico cualquiera y que es asignado al momento de evaluar un polinomio.

- **Exponentes**: Son los que eleva la variable, indicando el grado del polinomio.

### 2.1.2 Grado del polinomio

Para un monomio, el grado es la suma de todos los exponentes de las variables, para el caso de un polinomio, el grado está representado por el monomio de mayor grado, es decir, el mayor exponente al que se encuentra elevada la variable, existiendo polinomios desde grado $0$ a grado $n$.

## 2.2 MÉTODO DE NEWTON-RAPHSON

El método de Newton-Raphson permite hallar una raíz de una ecuación no lineal siempre y cuando se parta de una buena estimación inicial de la misma. El esquema iterativo de Newton puede derivarse del desarrollo de Taylor de la función alrededor de la estimación inicial, de la siguiente forma:

$$f(x) = 0 = f(x_0) + f'(x_0)(x - x0) + O(h^2)$$

Ahora bien, la recta tangente a la función, que pasa por el punto [$x_0, f(x_0)$], se encuentra definida por la siguiente expresión:

$$g(x) = f(x_0) + f'(x_0)(x - x0)$$

Se denomina $x_1$ a la intersección $g(x)$ con el eje de las abscisas ( es decir, la raíz de $g(x)$), resolviendo dicha ecuación se obtiene la siguiente expresión:

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

Y generalizando este esquema de aproximaciones sucesivas a la raíz, se obtiene:

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

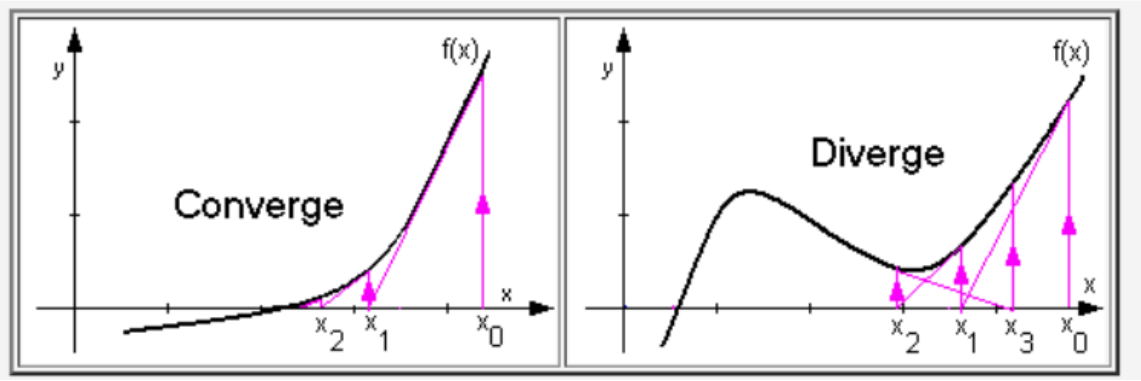


Figura 2.1: Ejemplo gráfico Newton-Raphson

Para que qsegurarse de que el método converge deben cumplirse ciertas condiciones. En la figura 2.1 se puede apreciar como aún partiendo de un punto cercano a la raíz buscada, en un caso el método converge y en el otro no.

# CAPÍTULO 3. DESARROLLO PRIMERA PARTE

La primera parte de este laboratorio consiste en graficar una serie de funciones haciendo uso de las herramientas que provee Matlab. A continuación se presenta el desarrollo de esta primera parte, junto con los resultados obtenidos.

## 3.1 GRÁFICOS SIMPLES Y COMPUESTOS

En esta sección se grafican las funciones:

$$a(x) = 6log_4(5x + 15) - log_2(2x)$$

$$b(x) = sin(6log_{10}(2x + 9)) + cos(2ln(x + 16))$$

Con dominio en el intervalo $[0, 15\pi]$. Para esto se requiere un gráfico para cada función, además de un tercer gráfico que muestre ambas funciones.

El resultado se presenta a continuación:

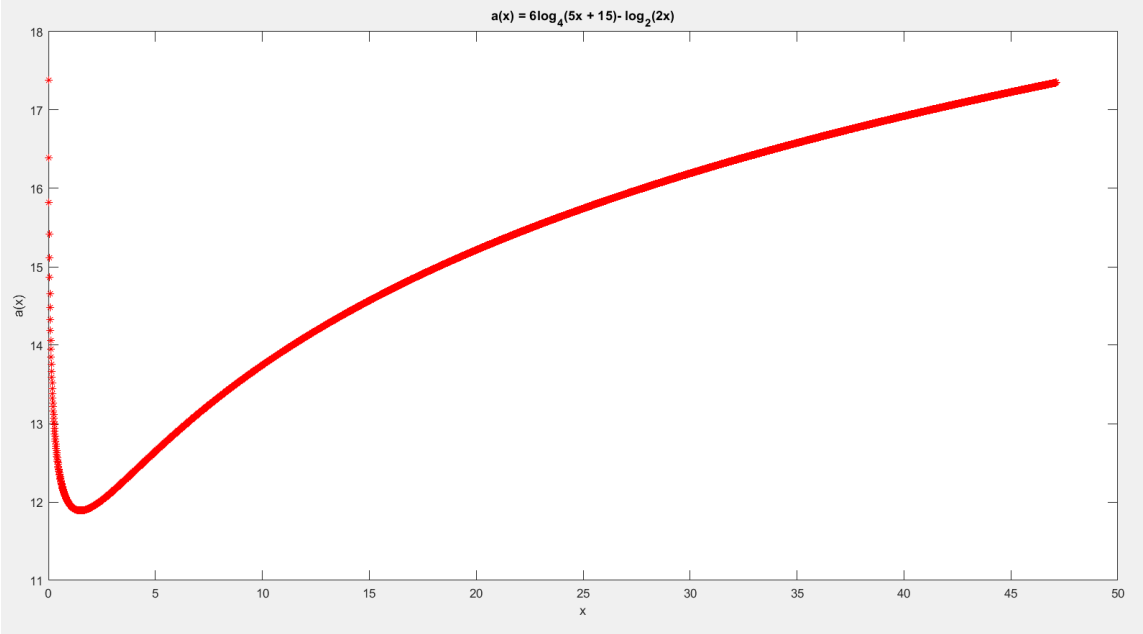


Figura 3.1: Gráfico de la función $a(x) = 6log_4(5x + 15) - log_2(2x)$

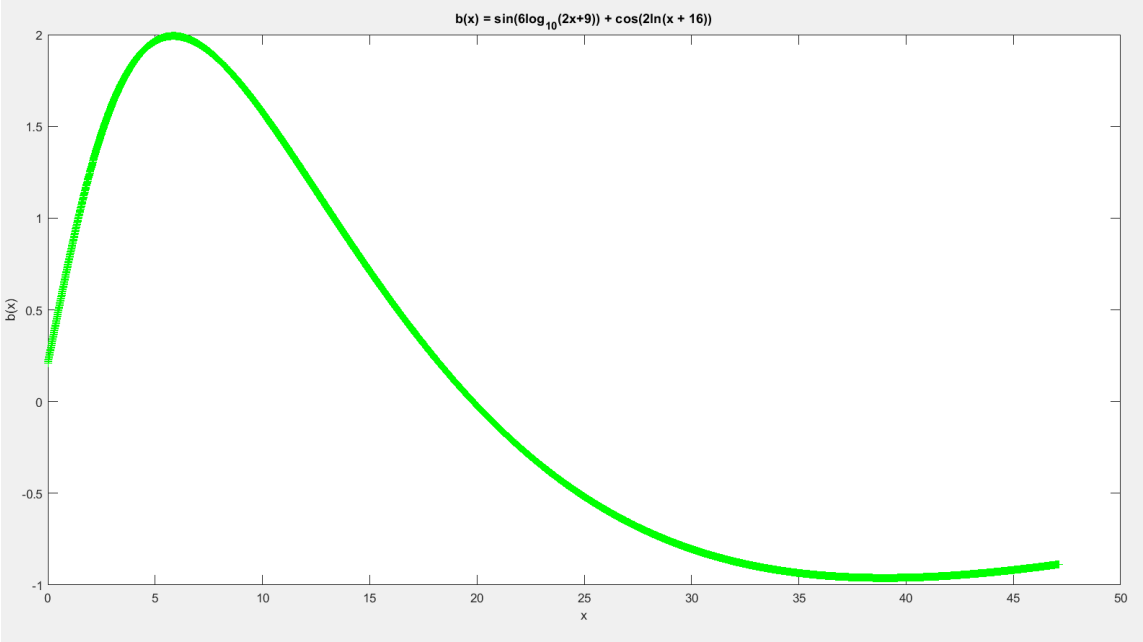


Figura 3.2: Gráfico de la función $b(x) = sin(6log_{10}(2x + 9)) + cos(2ln(x + 16))$

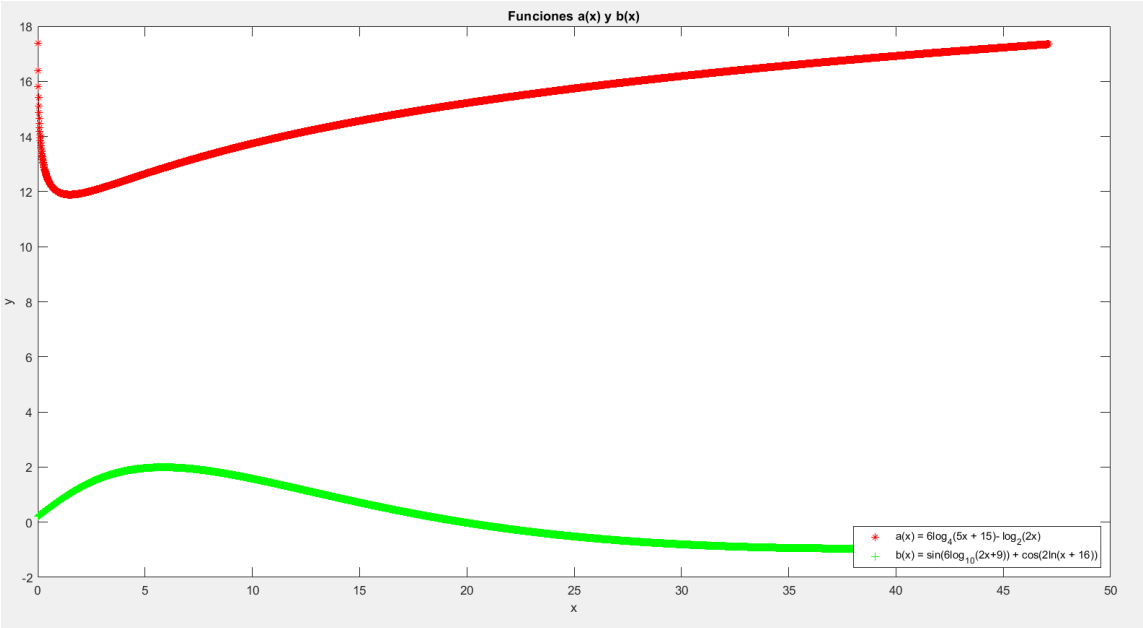


Figura 3.3: Gráfico de las funciones $a(x)$ y $b(x)$ en conjunto.

## 3.2 ESCALAS GRÁFICAS

En la segunda sección se grafica la función

$$c(x) = 2e^{2x+3} \; ; [-10, 10]$$

Usando escala normal y logaritmica en el eje Y, haciendo una comparación entre ambos gráficos.

El resultado obtenido fue el siguiente:

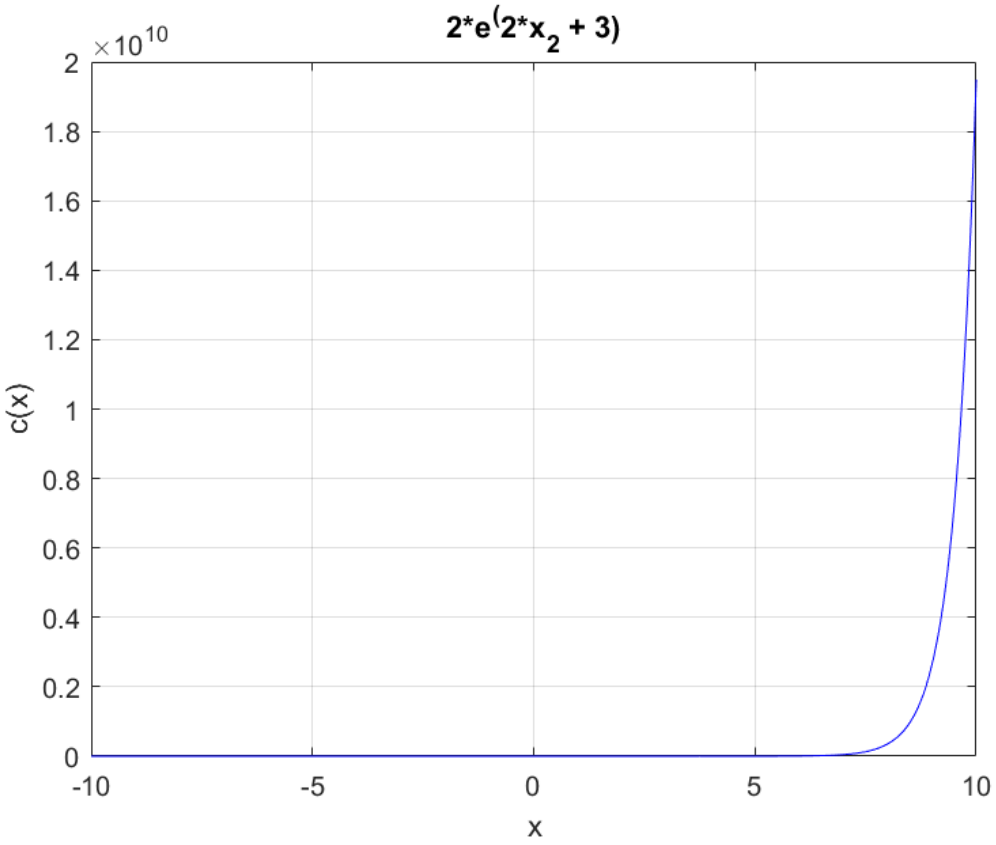


Figura 3.4: Gráfico $c(x)$, escala normal.

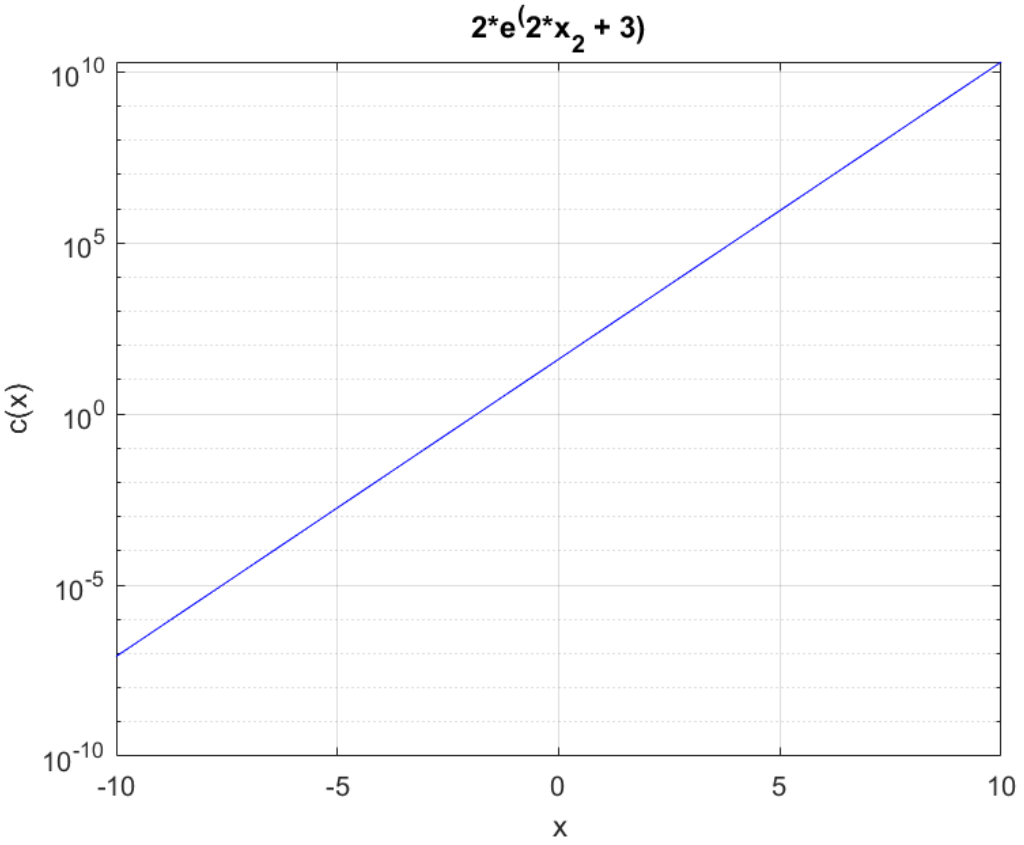


Figura 3.5: Gráfico $c(x)$, escala logaritmica.

Como se puede ver, la diferencia del gráfico es significativa. En el primer gráfico, los grandes cambios en el valor de la función a medida que aumenta $x$, opacan los valores más pequeños. Sin embargo, se aprecia de mejor forma el comportamiento exponencial de la función, siendo fácilmente comparable con funciones polinómicas. El segundo gráfico, al contrario, normaliza los cambios de la función, presentando una figura la cual sirve para comparar facilmente la magnitud de los valores que toma una función.

# CAPÍTULO 4. DESARROLLO SEGUNDA PARTE

## 4.1 ALGORITMO DE DESVIACIÓN ESTÁNDAR.

El algoritmo para el calculo de la desviación estándar recibe como entrada un arreglo de datos y retornará como salida la desviaciń estándar del conjunto de datos del arreglo.

Los pasos que sigue el algoritmo son como sigue:

1. Calcular la cantidad de elementos del arreglo de datos.

2. Sumar los elementos del arreglo y dividir por su cantidad para obtener la media.

3. Obtener las desviaciones de todos los datos, se elevan al cuadrado y se suman, obteniendo asi $\sum_{i=1}^{n} |x_i + \overline{x}|^2$

4. Se calcula la raiz cuadrada de la expresión anterior dividida en la cantidad de datos.

5. se retorna el valor de la desviación estándar.

```
1    % Función que permite obtener la desviación estandar de un arreglo de datos
2    % Entrada: Arreglo de datos
3    % Salida: Desviación Estandar
4    function [deviation] = std_dev(array)
5
6    % Se obtiene la cantidad de datos del arreglo
7    n = length(array);
8
9    % Se obtiene la media de los datos del arreglo
10   poly_mean = (sum(array) / n);
11
12   % Se obtiene la suma de las desviaciones cuadradas
13   square_sum = sum( abs( (array - poly_mean) ).^ 2 );
14
15   % Se calcula la desviación estándar
16   deviation = sqrt(square_sum / n);
17   end
```

Figura 4.1: Implementación del cálculo de la desviación estándar.

## 4.2 ALGORITMO DE NEWTON RAPHSON RECURSIVO.

El algoritmo para el calculo de la raiz de un polinomio a través del método de Newton Raphson recibe como entrada; el polinomio, la cantidad máxima de iteraciones a realizar, el error considerado para encontrar la raíz y el valor inicial del algoritmo. Su salida es la raiz del polinomio encontrada, en caso de no cumplir con el error considerado entregará el mejor acercamiento que se haya logrado realizar.

Los pasos que sigue el algoritmo son como sigue:

1. Se define el valor inicial entregado al algoritmo como posible raiz.

2. Se obtiene la derivada del polinomio dado.

3. Se obtiene la evaluación del polinomio en el valor inicial.

4. Si aún no se llega al tope de iteraciones se sigue el proceso. En caso contrario se indica la situación por consola y se termina el proceso.

5. Si la evaluación del polinomio en el valor inicial es mayor al error considerado se sigue el proceso. En caso contrario se indica por consola que se ha encontrado una raíz que cumple con el error considerado y se termina el proceso.

6. Se evalúa la derivada del polinomio en el valor inicial.

7. Se genera el valor para la siguiente recursión.

8. Se hace el llamado recursivo teniendo como entrada; el mismo polinomio, el número de iteraciones disminuido en uno, el mismo error, y el nuevo valor inicial calculado en el punto anterior.

```
function [root] = newton_raphson(poly, iter, error, initial)
% Se define el valor inicial como posible raiz
root = initial;
% Se obtiene la derivada del polinomio
derivate = polyder(poly);
% Se obtiene la evaluación del polinomio en el valor inicial
current = polyval(poly, initial);
% Se sigue el proceso mientras no se llegue al limite de iteraciones
if(iter > 0)
    % Se sigue el proceso si la evaluación actual del polinomio no cumple
    % con el error solicitado
    if(abs(current) > error)
        % Se obtiene la evaluación de la derivada del polinomio en el valor
        % inicial
        derivate_current = polyval(derivate, initial);
        % Se calcula el valor inicial de la siguiente recursión
        new_value = initial - (current / derivate_current);
        % Se hace el llamado recursivo con el nuevo valor e iteracion - 1
        root = newton_raphson(poly, (iter - 1), error, new_value);
    % Si la evaluación del polinomio cumple con el error solicitado se
    % informa a través de la consola
    else
        fprintf("La raiz del polinomio es %d.\n", initial);
    end
% Si se llega al limite de iteraciones sin llegar a una raiz que cumpla con
% el error solicitado se informa por consola y se entrega la ultima raiz
% calculada
else
    fprintf("Se han concluido las iteraciones sin encontrar una raiz que cumpla con el error.\n");
    fprintf("El ultimo valor encontrado ha sido %d.\n", initial);
end
end
```

Figura 4.2: Implementación del algoritmo de Newton Raphson recursivo.

# CAPÍTULO 5. CONCLUSIONES

Matlab es un lenguaje de programación que entrega múltiples herramientas que son útiles para el desarrollo de experiencias de todo tipo, pero dada su naturaleza se puede decir que su enfoque se centra en el área científica para el desarrollo de gráficos y eficiencia en operaciones matriciales y cálculo diferencial e integral.

Para la experiencia actual se hizo uso de este lenguaje para generar gráficos, explorando diferentes posibilidades en la confección de éstos, sus características, así como diferentes escalas para representar las funciones. Además, se implementó el método de Newton-Raphson utilizado para hallar una raíz real de un polinomio, en la cual se desarrolló una función recursiva que obtiene de forma aproximada esta raíz dado un error ingresado por parámetro. Finalmente, en la tercera parte se implementó la desviación estándar, solo usando la lógica del lenguaje y no sus funcionalidades.

Los aprendizajes obtenidos a partir de la experiencia del uso de Matlab serán de utilidad para los laboratorios posteriores, donde será necesario desarrollar programas más complejos, enfocados en la modelación de sistemas y representación de éstos.

# CAPÍTULO 6. ANEXOS

## 6.1 MANUAL DE USUARIO

A continuación se entregan las instrucciones de uso para las diferentes implementaciones realizada, entre esto se incluye el ingreso de parámetros para el funcionamiento del programa, resultados esperados y ejemplos de uso.

### 6.1.1 Primera parte

La primera parte del laboratorio entrega los gráficos de forma inmediata luego de haber ejecutado el script, este caso entrega cinco figuras correspondientes a los gráficos solicitados en el enunciado.

### 6.1.2 Segunda parte

#### *6.1.2.1 Desviación estándar*

Esta implementación es la que realiza la operación de la desviación estándar sobre un vector ingresado por teclado, el vector deseado debe ser ingresado como la representación de éstos en Matlab, es decir, los valores del vector deben ser ingresados entre corchetes ([]) y separados por espacios o comas, un ejemplo de un vector a ingresar puede ser el siguiente:

$$[2, 4, 3, 2, 4, 10]$$

También se debe tener en cuenta que al momento de ingresar el vector solo deberá contener números o el programa no se ejecutará correctamente.

```
>>
>> parte_2_1
Ingrese el arreglo: [ 2 3 1 3 2 2 10 2  3 42 2 32 1 3]
La desviación estandar del arreglo es 1.228488e+01.
>>
```

Figura 6.1: Ejemplo de uso de desviación estándar en MatLab

#### *6.1.2.2 Newton-Raphson*

Esta parte de la implementación consta de la implementación del método Newton-Raphson para obtener una raíz aproximada a un polinomio, en este caso se tiene un script para poder ingresar por teclado los parámetros necesarios, los cuales son: El polinomio como su representación en MatLab (como vector), el número máximo de iteraciones, el error y el punto inicial para comenzar a iterar. En la figura 6.2 se puede observar un ejemplo de uso, donde se busca la raíz del polinomio $4X^2 + 5X + 10$ con un máximo de 500 iteraciones, un error de 0.005 y un punto inicial de 0.

```
>> parte_2_2
Ingrese el polinomio: [4 5 10]
Ingrese el limite de iteraciones: 500
Ingrese el error máximo de la raiz a encontrar: 0.005
Ingrese el valor inicial a utilizar: 0
Se han concluido las iteraciones sin encontrar una raiz que cumpla con el error.
El ultimo valor encontrado ha sido -1.829699e+00.
>>
```

Figura 6.2: Ejemplo de uso de Newton-Raphson con función $4x^2 + 5x + 10$

La figura 6.3 muestra la resolución del polinomio $x^3 - 7x^2 + 5sin(2\pi)x - 6$ con un máximo de 500 operaciones, un error de 0.00001 y valor inicial de 9.

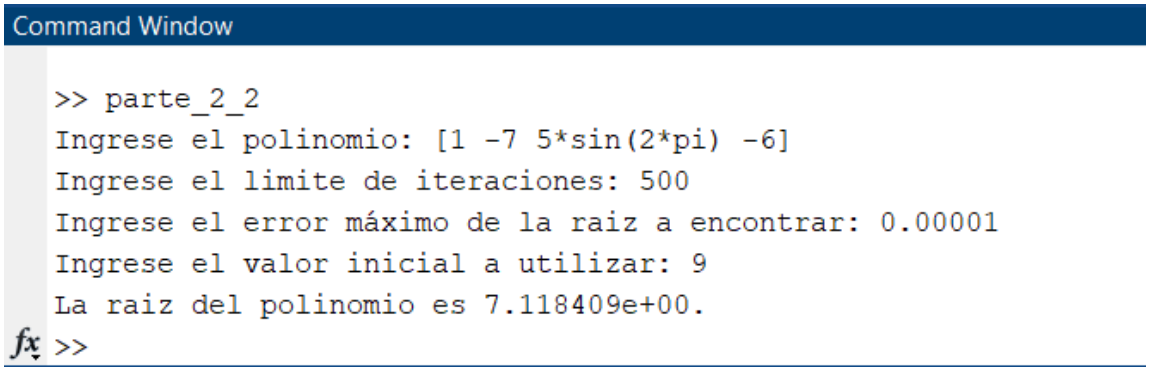


Figura 6.3: Ejemplo de uso de Newton-Raphson con función $x^3 - 7x^2 + 5sin(2\pi)x - 6$

La figura 6.4 muestra la resolución del polinomio $-3x^2 + 8x - 5$, primero con un máximo de iteraciones deliberadamente bajo donde se puede ver la salida del algoritmo en caso de no encontrar la raíz del polinomio. Luego se muestra la ejecución con un mayor máximo de iteraciones donde si se logra encontrar la raíz del polinomio.

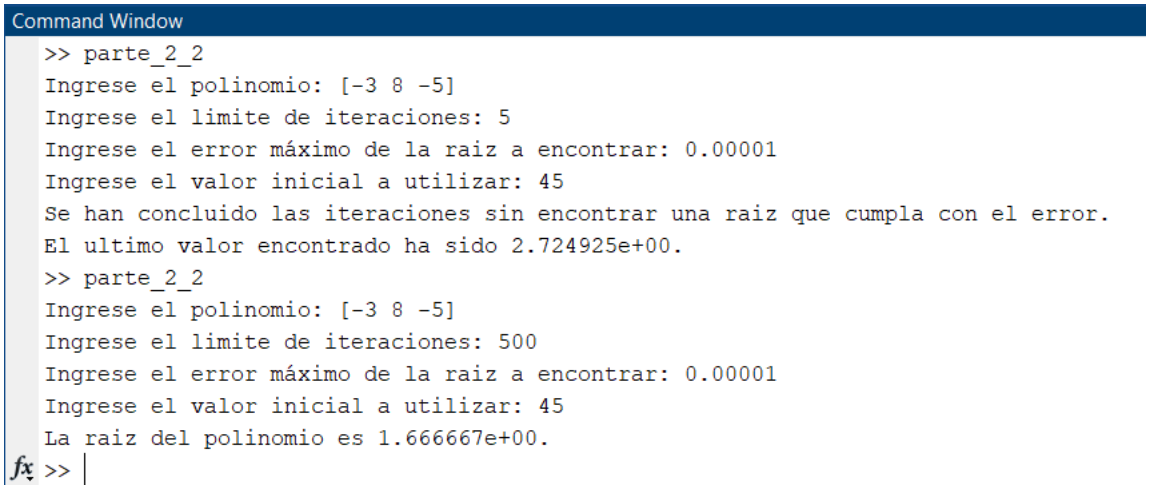


Figura 6.4: Ejemplo de uso de Newton-Raphson con función $-3x^2 + 8x - 5$

# BIBLIOGRAFÍA


- Pablo de Nápoli. (2014). Polinomios. 2021, de Universidad de Buenos Aires Sitio web: http://mate.dm.uba.ar/ pdenapo/apuntes-algebraI/polinomios.pdf
- MathWorks. (2013). Solving a Nonlinear Equation using Newton-Raphson Method. 2021, de MathWorks Sitio web: https://la.mathworks.com/matlabcentral/answers/107508-solving-a-nonlinear-equation-using-newton-raphson-method