# Modelación y Simulación
# Laboratorio 1
# Introducción a MATLAB

## 1 Objetivo

Este primer Laboratorio de Modelación y Simulación, tiene como objetivo principal acercar al alumno al manejo de MATLAB, con una introducción a este lenguaje interpretado.

## 2 Intérprete de comandos de MATLAB

### 2.1 Representación de variables y vectores

Estos se pueden representar por medio de letras, palabras, letras seguidas de números, básicamente como en cualquier tipo de lenguaje:

```
a = 100 variable
poli1 = [1 2 3 120] % polinomio
x = [1 2 3 4 5 6 7 8 9 10] % 10 elementos equidistantes
y = 5 * exp (x -3) % evaluacion de vector x elemento a elemento en y
```

### 2.2 Representación de matrices

Estas se ingresan al igual que un vector, donde las columnas van separadas por espacios y las filas por punto y coma.

```
matriz1 = [1 2 3; 4 5 6; 7 8 9]
matriz1 =
1 2 3
4 5 6
7 8 9
```

Si falta algún valor o no está bien dimensionada la matriz, ésta no quedará definida. Las matrices se pueden manipular de varias formas:

- Traspuesta:

```
traspuesta = matriz1'
traspuesta =
1 4 7
2 5 8
3 6 9
```

- Multiplicación de matrices:

```
mult = matriz1 * traspuesta % Es importante el orden de las matrices
mult =
14 32 50
32 77 122
50 122 194
```

## 2.3 Representación de polinomios

La representación de polinomios dentro de MATLAB se hace por medio de vectores, los cuales poseen la siguiente forma:

$$a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x^1 + a_0 \tag{1}$$

Por ejemplo para escribir el polinomio $f(x) = x^2 + 2x + 1$:

```
f = [1 2 1]
f =
1 2 1
```

## 2.4 Representación de funciones

La representación de una función cualquiera, es igual a la asignación de una variable, con el cuidado que la variable dependiente debe estar definida en sus puntos con anterioridad a la definición de la función. Es decir por ejemplo la definición de la función: y = sin x, se debe realizar de la siguiente forma:

```
x=-3:.5:3 % Nota: Se puede utilizar x = [-3:0.5:3]
x =
-3.0000 -2.5000 -2.0000 -1.5000 -1.0000 -0.5000 0 -0.5000 1.0000
1.5000 2.0000 2.5000 3.0000
```

La definición de estos puntos se realiza indicando la posición inicial, el intervalo de separación entre cada punto, y la posición final del intervalo.

Entonces la definición de la función:

```
y = sin ( x )
y =
-0.1411 -0.5985 -0.9093 -0.9975 -0.8415 -0.4794 0 0.4794 0.8415
0.9975 0.9093 0.5985 0.1411
```

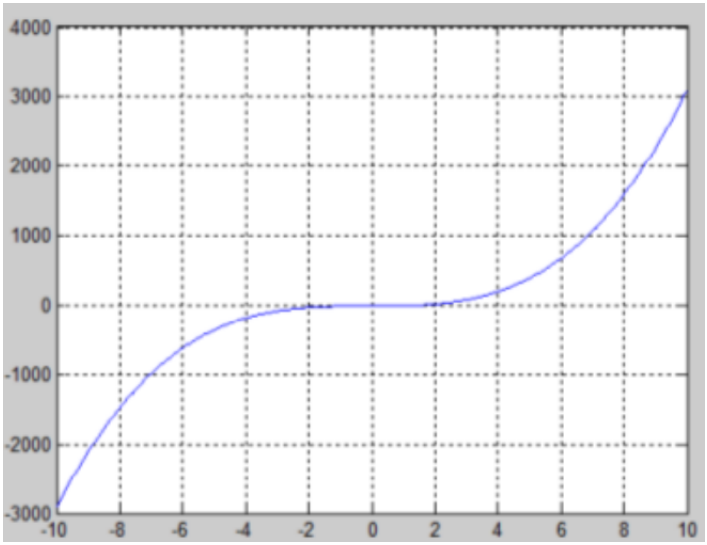
## 2.5 Representación de gráficos

Dentro de MATLAB, se pueden representar, tanto gráficos en dos como en tres dimensiones, para lo cual se define previamente los puntos de la función que se desea graficar, con el cuidado de que los puntos tanto de la variable dependiente e independiente sean de la misma dimensión.

- Ejemplo 1. Tomando la siguiente función para graficar, se realiza el siguiente proceso:

```
f = [3 1 0 -8]; %Se define el polinomio
x = -10:.05:10; %Se definen los puntos a ser evaluados
y = polyval (f , x ); %Se evaluan los puntos en la funcion
plot (x , y ); %Se grafica el polinomio
```

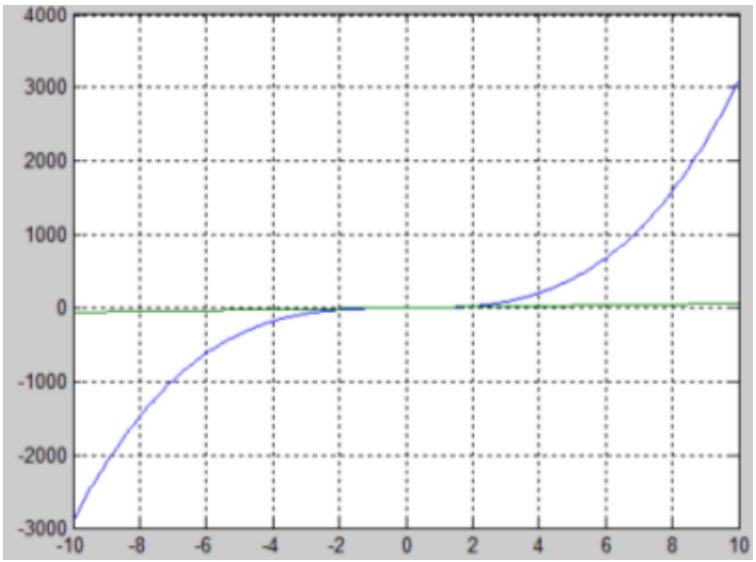
Resultando:




- Ejemplo 2

```
z = 6 * x;
plot (x , y , x , z );
```

Resultando:



### 2.5.1 Gráficas en escala logarítmica

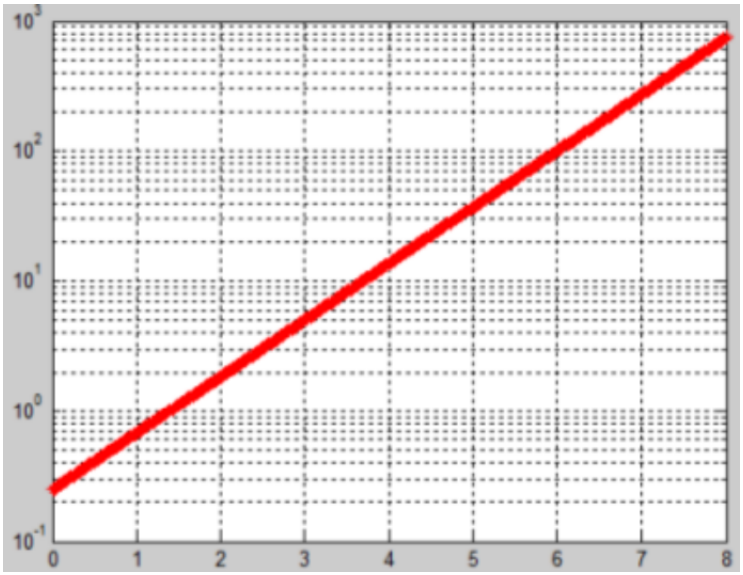
Las gráficas en escalas logarítmicas se realizan por medio de las siguientes funciones:

```
x = 0:.02:8; % Definir los puntos a ser evaluados por la funcion
y = 5 * exp (x -3); % Definir la funcion
semilogy (x , y , 'r * ' ); %Graficar la funcion en escala logaritmica
```

Para darle más utilidad a los gráficos, estos poseen ciertos comandos que hacen que estos sean más atractivos e ilustrativos, para nuestros fines, entre estos se encuentran los siguientes:

```
title ( ' Texto alusivo al titulo del grafico ' );
ylabel ( ' Texto alusivo al eje de las y ' );
xlabel ( ' Texto alusivo al eje de las x ' );
grid on %Muestra el grafico con grilla
plot (x , y , 'b * ' );
%plot genera un grafico en azul con puntos con forma de asterisco.
%Asi mismo se pueden utilizar otros colores y otros simbolos.
```

Resultando:



## 2.6 Programando en MATLAB

MATLAB permite programar funciones, para lo cual es necesario crear un archivo con el código fuente y extensión ".m", este archivo fuente contiene las instrucciones necesarias, existiendo, la posibilidad de utilizar si fuese necesarios sentencias de control de flujo, tales como while, do, for , if then else, (todos terminan con end), etc.

Ejemplo:

```
function [ poly3 ] = sumapoly ( poly1 , poly2 )
%funcion que recibe dos polinomios de cualquier orden y los suma
largo_poly1 = length ( poly1 );
largo_poly2 = length ( poly2 );
while ( largo_poly1 ~= largo_poly2 )
if ( largo_poly1 < largo_poly2 )
poly1 = [0 poly1 ];
largo_poly1 = length ( poly1 );
else
poly2 = [0 poly2 ];
largo_poly2 = length ( poly2 );
end
end
poly3 = poly1 + poly2 ;
```

Para ejecutar esta función desde MATLAB debe encontrarse en el path y el archivo.m debe llamarse igual que la función.

## 2.7 Operadores algebraicos

Se permiten todas las operaciones algebraicas comunes +, -, *, /, con el cuidado de que cuando las expresiones a ser manejadas son de tipos distintos (por ejemplo ya sea tanto por la dimensión de un polinomio, como por sumar una matriz a un polinomio, etc.), se deben tener en cuenta este tipo de salvedades.

Instrucciones importantes:

1. whos: despliega las variables activas con sus valores respectivos.
2. clear: borra todas las variables activas.
3. help: Proporciona ayuda a través de la consola de cada uno de los comandos.

# 3 Informe

Este informe se divide en dos partes, en donde se evalúan distintos aspectos de programación en MATLAB.

## 3.1 Primera parte

- Graficar por separado y en conjunto, las siguientes funciones (3 gráficos):

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

$$b(x) = cos(4(log_6(2x + 3))) + sen(2(log_6(2x + 5))) \tag{3}$$

  La primera función debe ser graficada en verde con +, la segunda con * y en rojo, en el intervalo [0, $10\pi$] con un espacio entre ellos de 0.01.

- Graficar en escala normal y logarítmica la siguiente función (2 gráficos):

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

  Con colores y estilo a elección, cuadriculando la figura (grilla) en el intervalo [–10, 10] con espaciado de 0.05. Haga una pequeña comparación de los gráficos generados, indicando posibles ventajas y desventajas de cada una de las escalas usadas.

- **Nota:** Todos los gráficos deben incluir título y etiquetas en los ejes. Para los gráficos con más de una función se debe indicar dentro del gráfico a qué función corresponde cada línea (5 gráficos en total).

## 3.2 Segunda parte

- Implementar el algoritmo de Newton-Raphson y que entregue como resultado una raíz de una función dada o una aproximación de ella. Considere que la función será de una sola variable, y que será dada por consola en formato de polinomio.

  **IMPORTANTE:** Para su resolución defina funciones que trabajen recursivamente. Tome en cuenta que las entradas de la función Newton Raphson serán 4: el polinomio, el número máximo de iteraciones, el error a considerar y el valor inicial del algoritmo.

- 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.

## 3.3 Formato del informe

Puede realizar su informe con Microsoft Word, LaTeX o un editor de texto de su preferencia. El informe debe contener:

1. Portada.
2. Introducción.
3. Marco teórico.
4. Desarrollo de la Primera Parte: Explicación del desarrollo de cada uno de los gráficos.
5. Desarrollo de la Segunda Parte: Explicación de la implementación de cada uno de los algoritmos.
6. Manual de uso con tres ejemplos a lo menos de la segunda parte.
7. Conclusiones.
8. Referencias (Formato APA).

**El informe debe ser escrito según formato memoria.**
**El código fuente debe estar correcta y completamente comentado.**
**Los laboratorios son en parejas.**
**Entrega informe: viernes 7 de abril de 2017 (23:55).**
**¡Éxito!**