UNIVERSIDAD DE SANTIAGO DE CHILE

FACULTAD DE INGENIERIA

DEPARTAMENTO DE INGENIERIA INFORMATICA

Laboratorio N°1: "Introducción a MATLAB"

Control y Simulación

Profesor: Gonzalo Acuña L.

Ayudantes: Alex Ahumada

Carlos Vilches

Fecha: 30-10-2013

ÍNDICE DE CONTENIDOS

1. Objetivos

Este primer Laboratorio de “Control y Simulación” tiene como objetivo principal acercar al alumno al manejo de MATLAB, con una introducción a este software de aplicación.

2. Intérprete de Comandos MATLAB

2.1 Representación de variables y vectores

Las variables se pueden representan por medio de letras, palabras, letras seguidas de números, básicamente como en cualquier tipo de lenguaje:

```
» a = 100                          /*variable*/
a =
   100

» poli1 = [1 2 3 120]              /*polinomio*/
poli1 =
    1     2     3     120

» x = [1:10]                       /*10 elementos equidistantes*/
x =
    1     2     3     4     5     6     7     8     9    10

» y = 5 * exp(x - 3) /*evaluación de vector x elemento a elemento en función y*/
y =
  1.0e+003 *
```

Columns 1 through 7

0.0007 0.0018 0.0050 0.0136 0.0369 0.1004 0.2730

Columns 8 through 10

0.74212.0171 5.4832

2.2 Representación de Matrices

Éstas 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:

- **Matriz traspuesta:**

  » traspuesta = matriz1’

Si matriz1 fuera compleja, la “ **’** ” hubiese entregado la traspuesta conjugada; para obtener la traspuesta utilice “ **.’** ”; los dos comandos son iguales si la matriz no es compleja.

- **Multiplicar matrices:**

  » mult = matriz1 * traspuesta (aquí es importante el orden de las matrices).

Es posible también invertir una matriz, sacar los valores propios, elevarla a una potencia (si la matriz es cuadrada), encontrar el polinomio característico, entre otros.

2.3 Representación de Polinomios

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

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

$\left[a_n \;\; a_{n-1} \;\; ..... \;\; a_1 \;\; a_0\right]$ vector del Polinomio

Ejemplo:

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. Por ejemplo la definición de la función: **y = sin (x)**, se debe realizar de la siguiente forma:

a) Definir los puntos que serán evaluados por la función, por lo tanto:

```
» x = -3:.5:3                    //Nota: También se puede utilizar   x = [-3 : 0.5 : 3]
 x =
 Columns 1 through 7
  -3.0000   -2.5000   -2.0000   -1.5000   -1.0000   -0.5000         0
 Columns 8 through 13
  -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.

b) La definición de la función:

```
» y = sin(x)
 y =
 Columns 1 through 7
  -0.1411   -0.5985   -0.9093   -0.9975   -0.8415   -0.4794         0
 Columns 8 through 13
   0.4794    0.8415    0.9975    0.9093    0.5985    0.1411
```

2.5 Representación Gráfica

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.

a) ***Ejemplo N° 1:*** Tomando la siguiente función para graficar, se debe hacer:

1. Definir el polinomio.

```
» f = [3 1 0 -8];
```

2. Definir los puntos a ser evaluados por la función

```
» x = -10:.05:10;
```

3. Evaluar los puntos en la función (Polinomio).

```
» y = polyval(f,x);
```

4. Graficar el polinomio.

```
» plot(x,y);
```

Resultando en el siguiente gráfico (figura):

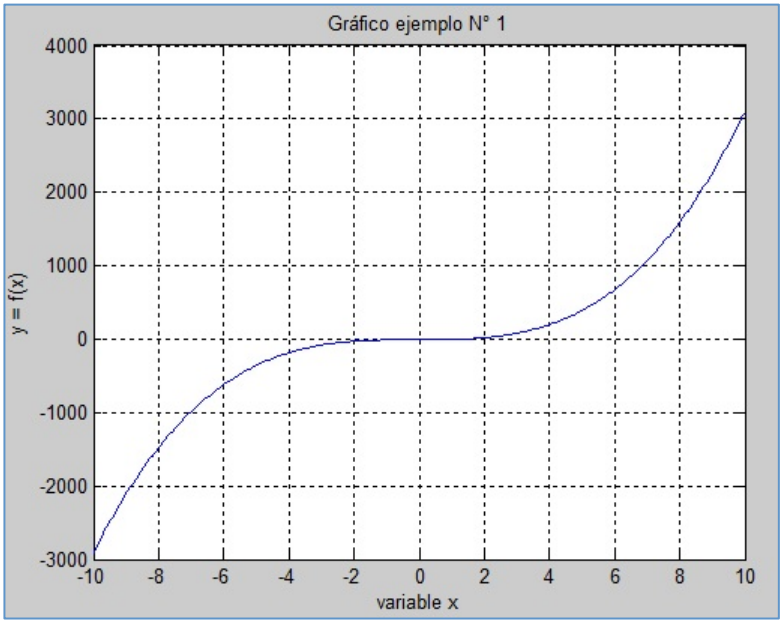

Figura 1*: Resultado ejemplo N° 1*.

Para graficar más de una función en una misma ventana, hay que tener cuidado que tengan la misma cantidad de puntos definidos, la instrucción es la siguiente.

» plot(x,y,x,z);

Donde **x** son los puntos ya definidos previamente e **y**, **z** son los puntos de dos funciones distintas, que han sido evaluadas con anterioridad en los puntos de x.

b) Ejemplo N° 2:

» z = 6 * x;

» plot(x,y,x,z);

Resultado:

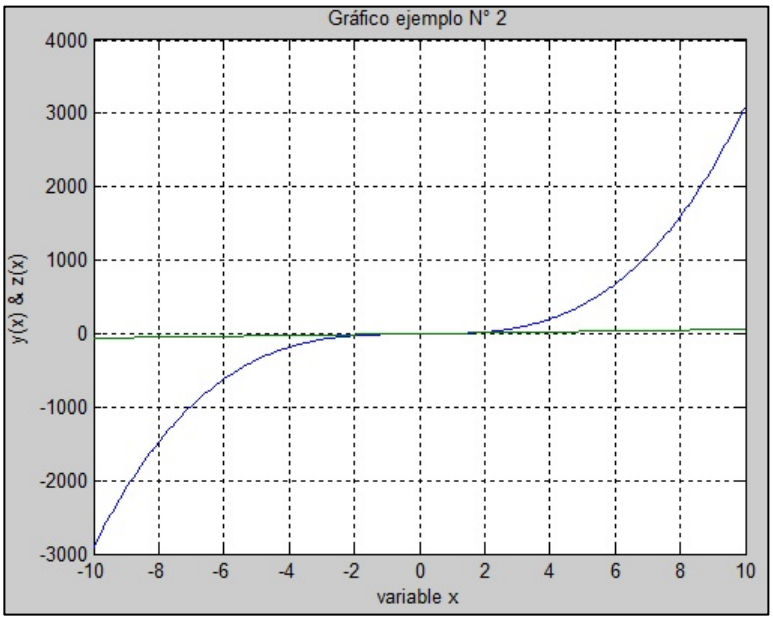

Figura 2*: Resultado ejemplo N° 2.*

c) Gráficas en escala logarítmica

Las gráficas en escalas logarítmicas se realizan mediante las siguientes instrucciones:

1. Definir los puntos a ser evaluados por la función.

» x = 0:.02:8;

2. Definir la función.

» y = 5 * exp(x-3);

3. Graficar la función en escala logarítmica.

» semilogy(x,y,'r*');

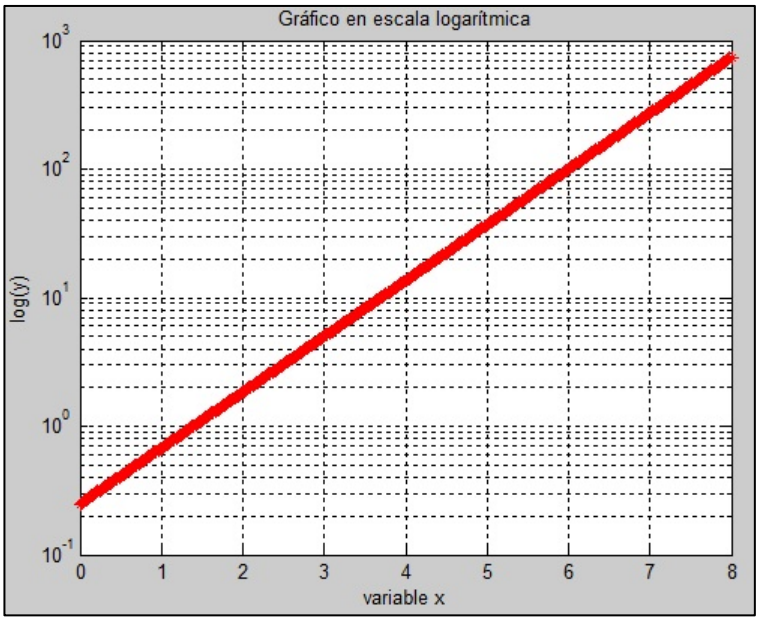

Figura 3*: Gráfico en escala logarítmica.*

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

» title(' Texto alusivo al título del grafico ');

» ylabel(' Texto alusivo al eje de las y ');

» xlabel(' Texto alusivo al eje de las x ');

» gtext(' Texto dentro del grafico ');

Nota: se debe ubicar la posición de este texto dentro del gráfico con el mouse.

» plot(x,y,'b*');

Esta opción genera un gráfico en azul con puntos con forma de asterisco (*). Así mismo se pueden utilizar otros colores y otros símbolos.

2.6 Programando bajo 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 necesario- sentencias de control de flujo, tales como *while*, *do*, *for*, *if*, *then*, *else* (todos terminan con *end*), etc.

Ejemplo:

```
     function [poly3] = sumapoly(poly1,poly2)
     %Comentario: función 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;
```

Este ejemplo permite sumar dos polinomios de distinto grado sin problemas.

NOTA: Para ser ejecutado este programa desde MATLAB debe encontrarse en el path.

1. Con el comando path se despliegan todos los directorios pertenecientes a éste:

» path

2. En caso de que su programa este guardado en un directorio que no pertenezca al path, se debe colocar:

» path (path, 'ruta del directorio donde se encuentra su archivo');

Ejemplo:

» path (path,'c:\');

Ejecución:

» nombre_archivo(pol1,pol2);

NOTA: Para ver el resultado se puede asignar la instrucción anterior a una variable o en su defecto colocar **ans** en la consola, ya que ésta es una variable temporal donde quedan los resultados de las funciones evaluadas.

2.7 Operadores Algebraicos

Se permiten todas las operaciones algebraicas comunes +, -, *, /, ^, teniendo cuidado cuando las expresiones a ser manejadas son de tipos o dimensión distinta.

Operadores:

+	Suma
-	Resta
*	Multiplicación
.*	Multiplicación de arreglo elemento a elemento
^	Potencia
.^	Potencia de arreglo elemento a elemento
\	División a la izquierda
/	División a la derecha
./	División de arreglo elemento a elemento
.	Punto decimal
%	Comentario
=	Asignación
==	Igualdad
< >	Operadores relacionales
&	AND
|	OR
~	NOT
xor	EXCLUSIVE OR

Instrucciones:

Whos:	despliega las variables activas con sus valores respectivos.
Clear:	borra todas las variables activas.
Whitebg:	coloca un fondo blanco a la figura activa.
Grid:	coloca una grilla en la figura activa.
Help:	proporciona ayuda a través de la consola.

3. Informe

Para el informe de Laboratorio se requieren los siguientes tópicos:

3.1 Primera Parte

a) Graficar por separado y en conjunto, las siguientes funciones:

$$a(x) = 7 * \log_5(2x - 3)$$

$$b(x) = sin(13 * \log_7(x + 2))$$

La primera en azul con '.', la segunda con 'x' y en rojo, tomando como puntos de *0* a 4 * *pi* con un espacio entre ellos de *0.01*.

b) Graficar en escala Normal y Logarítmica la siguiente función:

$$c(x) = 2x * e^{(5x-7)}$$

Colores y estilo a elección, cuadriculando la figura (grilla) y con puntos entre –30 y 30 (con espaciado de 0.05).

Nota: Todos los gráficos deben incluir grilla, título y etiquetas en los ejes; para los gráficos con más de una función se debe indicar dentro del gráfico a qué función corresponde. (5 gráficos en total)

3.2 Segunda parte

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

   IMPORTANTE: Para su resolución defina funciones que trabajen recursivamente. Tome en cuenta que la entrada de la función Newton-Raphson serán 3: *el polinomio* "[]", *n: número máximo de iteraciones* y *e: error a considerar*.

b) Crear un archivo ".m" que reciba como entrada un vector, y despliegue por pantalla el resultado de la suma de la raíz cuadrada de los 3 elementos de mayor valor (Manejar el ingreso erróneo en el número de elementos del vector).

Nota1: Si es necesario especifique las restricciones y supuestos que estime conveniente.

Nota2: Se debe realizar un control de errores (entrada, cantidad de elementos, etcétera).

3.3 Formato del Informe

a) Portada
b) Introducción Teórica (1 pág introducción, 2 páginas teoría)
c) Desarrollo Primera Parte: explicación del desarrollo de cada uno de los gráficos
d) Desarrollo de la segunda Parte: explicación de la implementación de cada uno de los algoritmos, código fuente (comentado y estructurado).
e) Manual de Uso, con tres ejemplos para cada algoritmo (segunda parte).
f) Conclusiones
g) Referencias (Formato APA)

Máximo de integrantes por grupo: 2

Fecha de entrega: miércoles 6 de Noviembre del 2013

Nota: La introducción teórica comprende los elementos de una introducción (motivación, objetivos, partes del informe, etc.) y un resumen del marco teórico asociado al tema de estudio, que en este caso es MATLAB.

Nota2: El informe debe ser escrito según formato tesis.

Importante:

1.- En caso de ser detectado, se penalizará con un descuento en la calificación final ante copias textuales desde la web y/o de otros trabajos (actuales o de semestres anteriores). Ante una reincidencia se aplicará el reglamento vigente.

2.- Los trabajos serán subidos a la plataforma moodle (usachvirtual), adjuntando el código desarrollado, en la fecha acordada.

3.- Se descontará puntos de la nota por la falta de algún ítem en el informe.

4.- Se descontarán puntos por mala presentación, redacción u ortografía.

5.- Se descontará 1 punto por hora de atraso luego de fijar la fecha de entrega.