INTRODUCCIÓN A MATLAB

Sebastián Domínguez Rivera

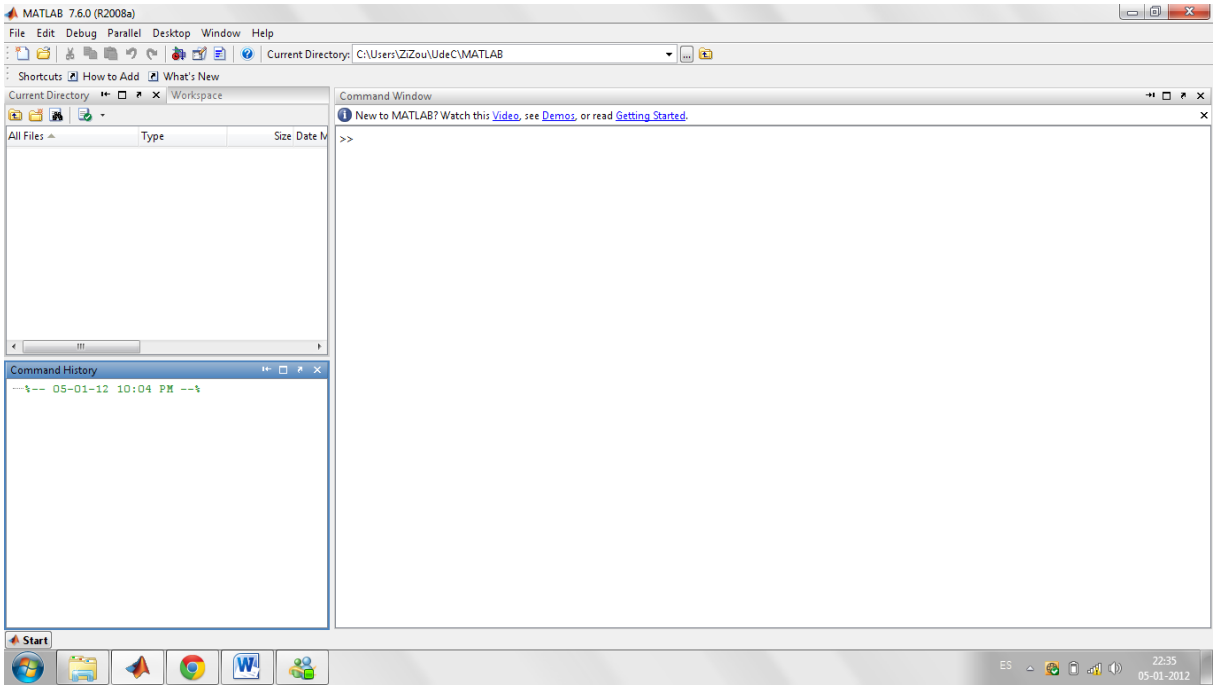
Universidad de Concepción, Chile

Concepción, 05 de enero de 2012

Contents

1 Introduciéndonos en el entorno MATLAB

Matlab es un software que posee una variada gama de aplicaciones que hacen de él un muy buen instrumento de trabajo para fines científicos y académicos. También está determinado por partes visibles que conforman la ventana de Matlab, las cuales son, por defecto, las ventanas de *Command Window*, *Command History*, *Workspace* y *Current Directory*, además, de las aplicaciones como los son *File*, *Edit*, *Debug*, *Parallel*, *Desktop*, *Window* y *Help* y las herramientas básicas que nos entrega Matlab por defecto que aparecen debajo de las aplicaciones antes mencionadas.



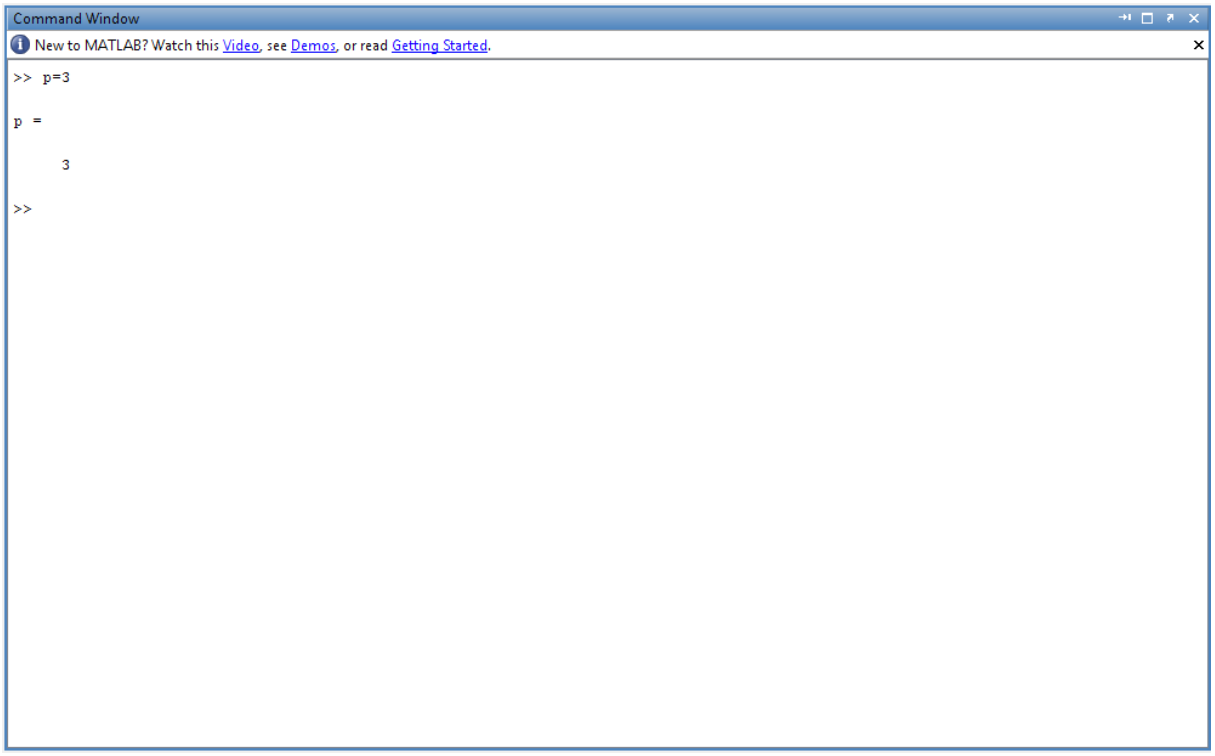
1.1 La Ventana *Command Window*

Esta ventana de Matlab es una de las ventanas más importantes para trabajar. Es donde el usuario ingresa las *entradas*, que son los valores o variables que recibe el programa, y recibe su respectiva *salida*, la cual será definida como la respuesta a una entrada dada. Por ejemplo, al ingresar un número en esta ventana, para que Matlab la reconozca y grabe, debemos apretar *enter*, con lo que Matlab guardará tal entrada y la denominará con el nombre de `ans`. También podemos ponerle nombre a alguna entrada que queramos ingresar a Matlab. Por ejemplo, si queremos ingresar el número 3 y definirla con la letra `p`, entonces debemos escribir lo siguiente en la *Command Window*

```
>> p=3
```

Al presionar enter se obtiene la respuesta

```
p =

     3
```

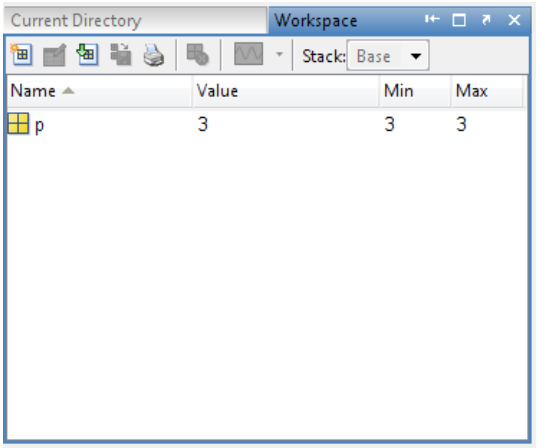


Muy frecuentemente se quieren ingresar una gran catidad de entradas mediante la *Command Window*, por lo que la respuesta, como la mostrada anteriormente, ocupa mucho espacio, lo que la mayoría de las veces es innecesario. Una solución a esto, es usar el *punto y coma* al final de cada ingreso de cada entrada, es decir

```
>> p=3;
```

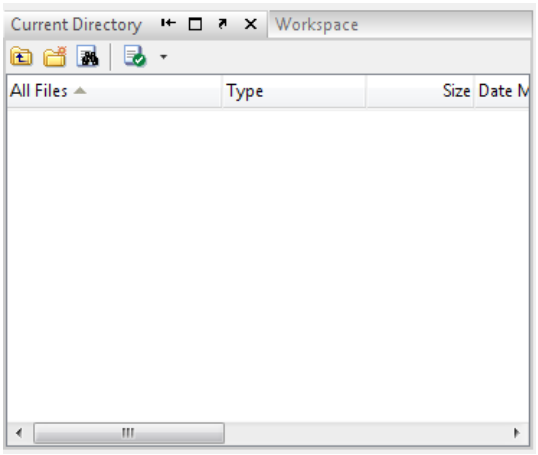
1.2 La Ventana *Workspace*

La principal función de esta ventana es la de mostrar las variables guardadas desde la *Command Window*, mostrando información adicional de las entradas guardadas.



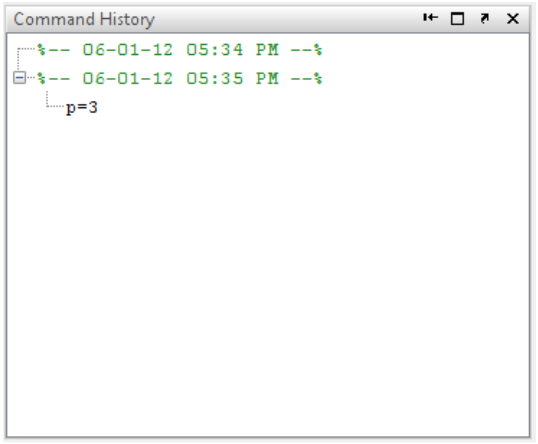
1.3 La Ventana *Current Directory*

En esta parte, se indica el lugar del disco duro en el cual Matlab guarda los archivos creados durante el uso del software. Por defecto, esta dirección está dirigida a una carpeta llamada Matlab que se aloja en *Mis Documentos*, al momento de la instalación del programa.



1.4 La Ventana *Command History*

Esta ventana nos muestra las acciones que se graban durante el uso del software, ordenadas cronológicamente, y almacenando también las entradas y salidas que se generan durante el uso de Matlab. Tal información aparece ordenada por días. Además, el programa graba la hora y día en la cual el programa se inicio.



1.5 Aplicaciones y Herramientas Básicas

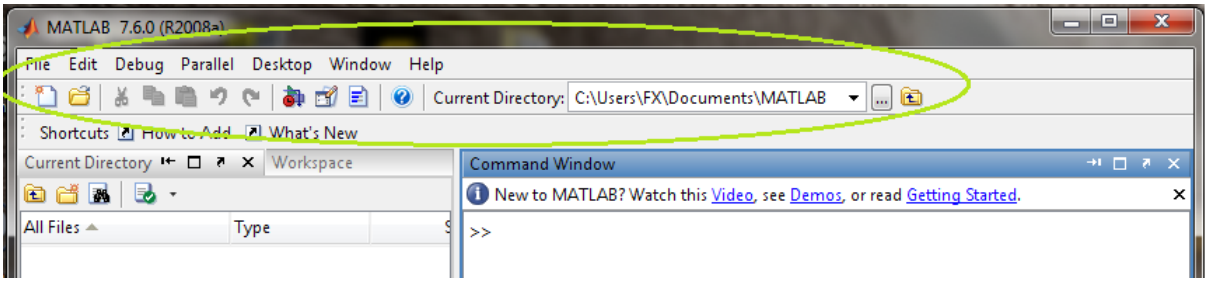
MATLAB posee muchas aplicaciones y herramientas visibles en la ventana de MATLAB por defecto. Son muy útiles al momento de realizar programas en la ventana *Command Window* o desde el *editor* de MATLAB. Es más, esta última aplicación tiene un espacio en el encabezado de la ventana da MATLAB, junto con, por ejemplo *open file*, *cut*, *paste*, *copy*, ect., que son de las más usadas y útiles por los usuarios de MATLAB de nivel básico. También en esta parte es posible cambiar el directorio en el cual se está trabajando y en donde se guardarán los programas creados en el editor (en el capítulo 5 se mostrará cómo relaizar y guardar dichos programas), si por ejemplo lo quisieramos cambiar a algún otro lugar de nuestra eleccó. En Windows, al momento de instalar MATLAB (la instalación de MATLAB no se abordará en este curso), el isntalador crea una carpeta de directorio llamada MATLAB en la carpeta Mis Documentos.

Otras dos aplicaciones muy importantes lo son *file* y *help*. La primera nos sirve, entre muchas cosas, para guardar programas, historial, abrir programas creados antiguamente. La segunda nos

llevará a un entorno en donde podemos encontrar una pequea guía de cómo usar cada comando que trae por defecto el programa MATLAB. Otra opción de usar esta aplicación, es usando el comando *help* en la ventana *Command Window*. Por ejemplo si quieramos usa un comando llamado `introamatlab` (este comando no existe en MATLAB, sólo fue usado como un ejemplo en este caso), debemos escribir

```
>> help introamatlab
>> INTROAMATLAB(x,y,z) is a algorithm ...
```

Apareciendo mas abajo una descripicón y ejemplos de cómo usar tal comando.



2 Matrices y vectores en Matlab

Matlab es un software que trabaja mediante entradas llamadas matrices. En matemáticas una matriz que un ordenamiento de números, ordenados por filas y columnas como sigue

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} = (a_{ij}) \tag{1}$$

Es decir, esta matriz A posee m filas y n columnas. Analogamente, un vector será un ordenamiento de números con una columna y varias filas, es decir,

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} \tag{2}$$

Observar de lo anterior, que x posee m columnas y una fila. Es decir, que un vector no es nada más que sólo una matriz con $n = 1$.

También, de la definición de la matriz A, podemos definir su matriz traspuesta A^T, que es cambiar filas por columnas, por

$$A^T = \begin{pmatrix} a_{11} & \cdots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{mn} \end{pmatrix} = (a_{ji}) \tag{3}$$

Así, llamaremos a x, definido en (2), como vector columna y a x^T como vector fila. Finalmente observar que un número (escalar) a es una matriz de una fila y una columna. Es decir, que

$$a = (a) \tag{4}$$

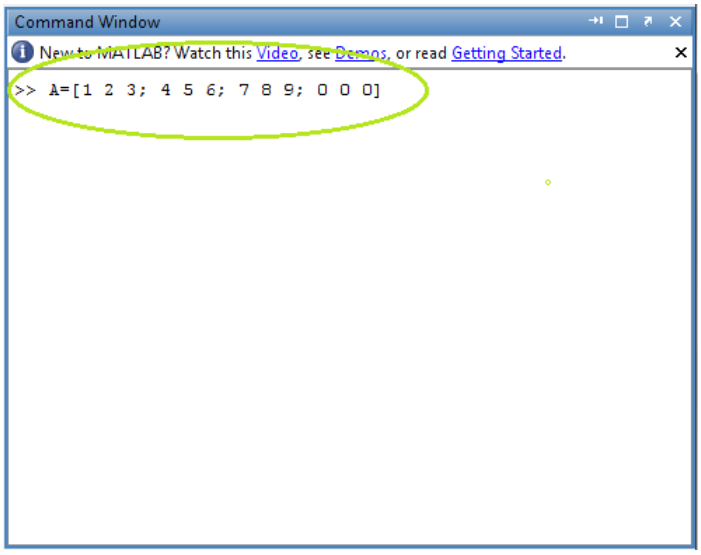
2.1 Ingresando Matrices y vectores mediante la ventana *Command Window*

Al querer ingresar una matriz a Matlab, debemos de tener en cuenta la estructura de la matriz misma, es decir, cuantas filas y columnas tiene la matriz. Si por ejemplo queremos ingresar la matriz

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 0 & 0 & 0 \end{pmatrix} \tag{5}$$

Debemos escribir lo siguiente

```
>> A=[1 2 3; 4 5 6; 7 8 9; 0 0 0]
```



Podemos notar que los *punto y coma* dentro de los corchetes cuadrados delimitan cada fila, mientras el espacio entre cada número, delimita cada columna.

Otra opción es ingresarla de la siguiente forma

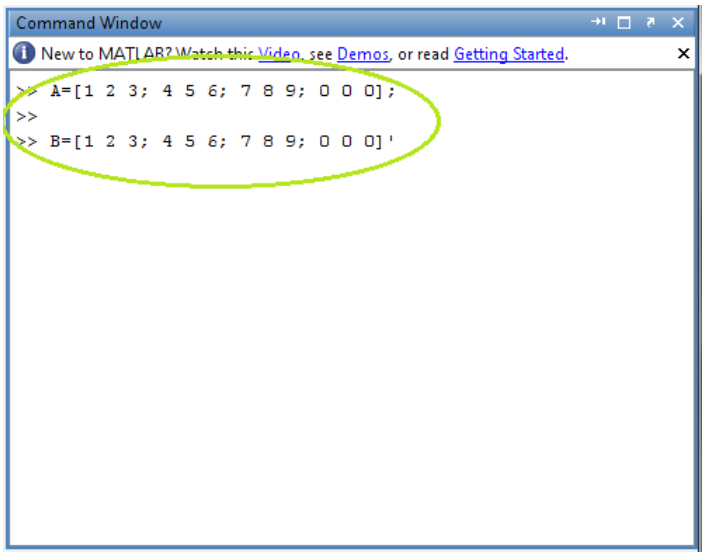
```
>> A=[1 2 3; 4 5 6; 7 8 9; 0 0 0];
```

Lo que nos entregará como respuesta

```
>> A=[1 2 3; 4 5 6; 7 8 9; 0 0 0];
>>
```

Para encontrar la matriz traspuesta en Matlab, de por ejemplo nuestra matriz A anterior escribirmos

```
>> B=[1 2 3; 4 5 6; 7 8 9; 0 0 0]'
```



Si queremos ingresar el vector columna

$$x = \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix} \tag{6}$$

Entonces escribimos

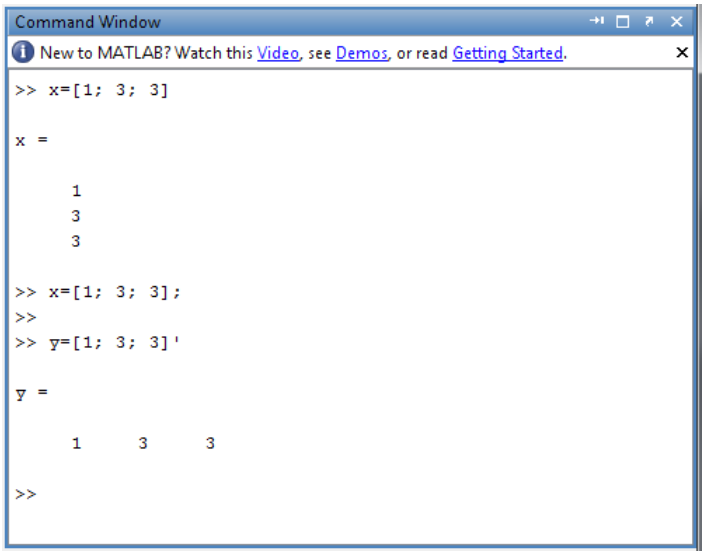
```
>> x=[1; 3; 3]
```

O también,

```
>> x=[1; 3; 3];
```

Y su respectivo vector fila sería

```
>> y=[1; 3; 3]'
```

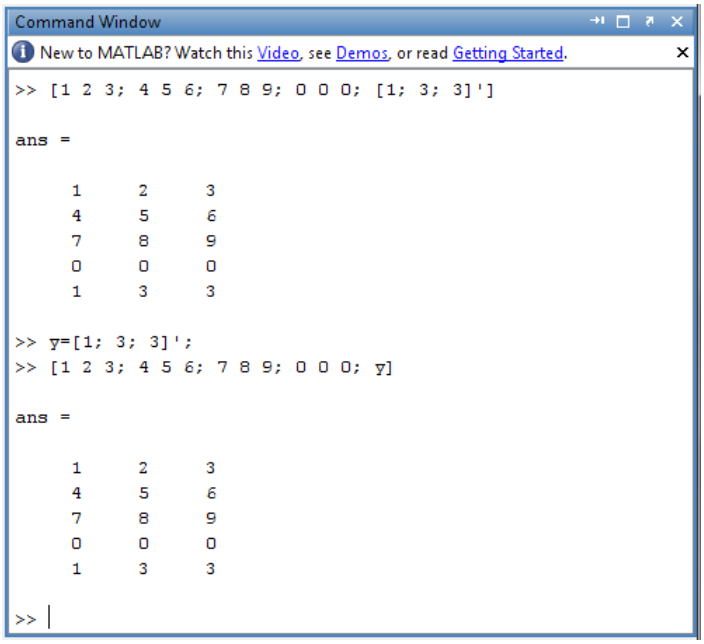


Si además quisieramos agregar el vector fila y como última fila a la matriz A, esta nueva matriz puede ser ingresada como sigue

```
>> [1 2 3; 4 5 6; 7 8 9; 0 0 0; [1; 3; 3]']
```

O bien,

```
>> y=[1; 3; 3]';
>> [1 2 3; 4 5 6; 7 8 9; 0 0 0; y]
```



2.2 Operaciones Básicas

2.2.1 Suma y multiplicación usuales

Entre matrices hay dos operaciones básicas válidas: la suma, multipicación y multipicación por escalar. En Matlab también podemos hacer estas dos operaciones entre matrices siempre y cuando sea posible, pues se sabe que, por ejemplo, las matrices

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 0 & 0 & 0 \end{pmatrix}, \; x = \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix} \tag{7}$$

Pueden multiplicarse como Ax pero no sumarse ni multiplicarse como xA.

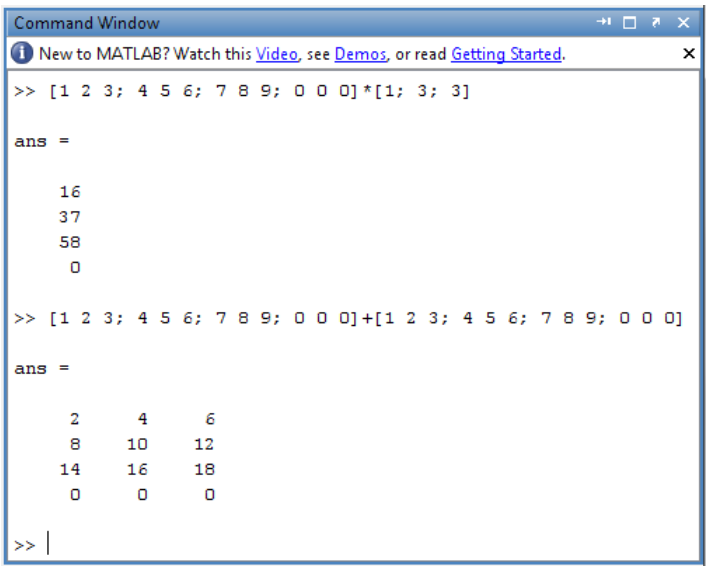
Es por esto, que al momento de querer sumar matrices debemos tomar en cuenta que todas las matrices que se esten sumando tengan las misma cantidad de filas y columnas entre ellas, y que al hacer la multipiclación AB entre dos matrices A y B cualesquieras, la cantidad de columnas de A debe ser igual a la cantidad de filas de B.

Siguiendo con el ejemplo anterior, para hacer la multipicación Ax en Matlab, debemos ingresar lo siguiente en la ventana *Command Window*

```
>> [1 2 3; 4 5 6; 7 8 9; 0 0 0]*[1; 3; 3]
```

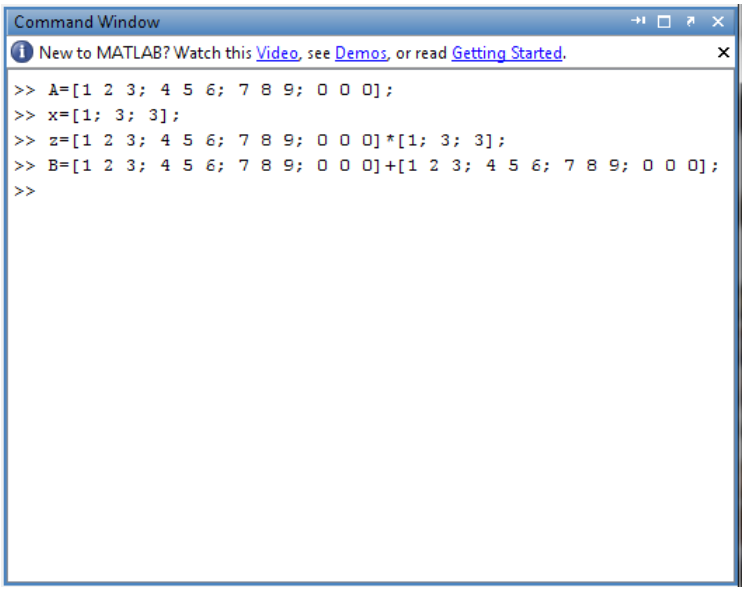
El signo * es la multiplicación usual de matrices. Por otro lado si quisieramos hacer la suma de matrices $A + A$ escribimos

```
>> [1 2 3; 4 5 6; 7 8 9; 0 0 0]+[1 2 3; 4 5 6; 7 8 9; 0 0 0]
```



Otra opción, aplicando el la abreviación de Matlab, *punto y coma* al final de cada entrada, pueden ingresarse las operaciones anteriores de la siguiente forma

```
>> A=[1 2 3; 4 5 6; 7 8 9; 0 0 0];
>> x=[1; 3; 3];
>> z=[1 2 3; 4 5 6; 7 8 9; 0 0 0]*[1; 3; 3];
>> B=[1 2 3; 4 5 6; 7 8 9; 0 0 0]+[1 2 3; 4 5 6; 7 8 9; 0 0 0];
>>
```



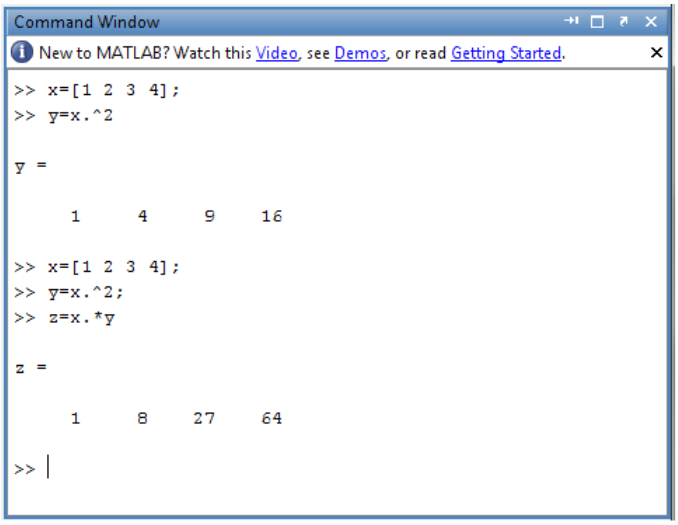
2.2.2 Exponente, multiplicación y división de los elementos de una matriz

Muchas veces es útil y necesario el calcular, por ejemplo la n-ésima potencia de cada elemento de una matriz o vector. Esto también es posible hacerlo en Matlab mediante la siguiente escritura

```
>> x=[1 2 3 4];
>> y=x.^2

y =

   1 4 9 16

>>
```

En el caso anterior se elevó al cuadrado cada elemento del vector x. También es posible hacer lo siguiente

```
>> x=[1 2 3 4];
>> y=x.^2;
>> z=x.*y
y =

   1 8 27 64

>>
```

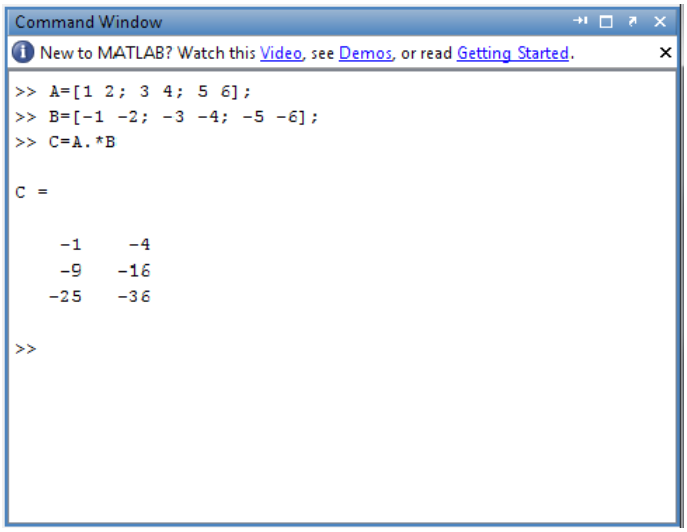


Por otro lado, también podemos multiplicar elemento a elemento cada matriz, diferentemente a la multiplicación usual de matrices. Si por ejemplo quisieramos hacer la multipicación elemento a elemento de las matrices

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 3 \\ 5 & 6 \end{pmatrix}, B = \begin{pmatrix} -1 & -2 \\ -3 & -4 \\ -5 & -6 \end{pmatrix}$$

En MATLAB ingresamos

```
>> A=[1 2; 3 4; 5 6];
>> B=[-1 -2; -3 -4; -5 -6];
>> C=A.*B

C =

   -1  -4
   -9 -16
  -25 -36

>>
```

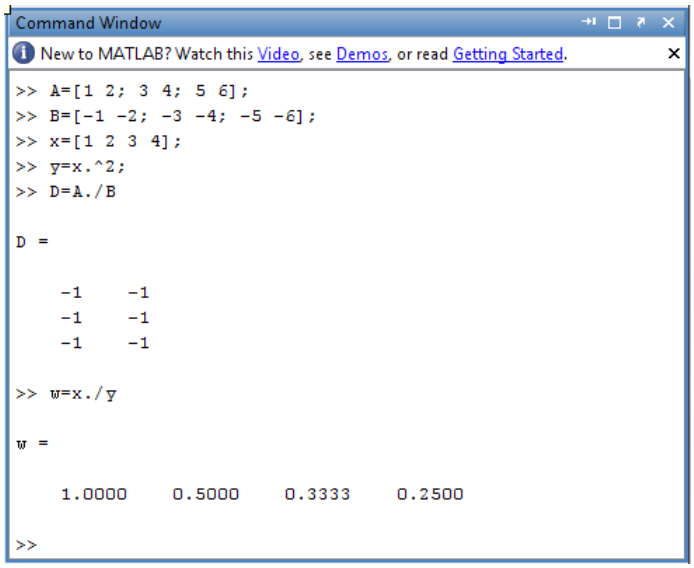


Observar que la multiplicación usual de matrices AB en este caso no es válida, pero si la multiplicación elemento a elemento.

Análogamente, podemos hacer la división elemento a elemento entre dos matrices, siempre que la matriz que divida a la otra no tenga entradas nulas. Tomando las matrices A y B como antes escribimos

```
>> A=[1 2; 3 4; 5 6];
```

```
>> B=[-1 -2; -3 -4; -5 -6];
>> x=[1 2 3 4];
>> y=x.^2;
>> D=A./B
>> w=x./y
>>
```



3 Funciones de una variable

Una función de números reales es una aplicación $f : Dom(f) \subseteq \mathbb{R} \to \mathbb{R}$, donde $Dom(f)$ se llama dominio de f, que lo definiremos como los valores posibles que pueda tomar f, que llamaremos preimágenes, y denotaremos por $Rec(f) \subseteq \mathbb{R}$ al conjunto de los valores que entrega nuestra función f, que llamaremos recorrido. A los elementos del recorrido de f los llamaremos imágenes f.

Ejemplo 1. La función $f : \mathbb{R} \to \mathbb{R}$ dada por

$$f(x) = x^2$$

Tiene como dominio a $\mathbb{R}$ y recorrido a $[0, +\infty)$, es decir, que $Dom(f) = \mathbb{R},\ Rec(f) = [0, +\infty)$.

3.1 Cómo ingresar funciones de una variable en MATLAB?

Observar de lo dicho anteriormente, que las funciones dependen de una variable independiente, denotada por x usualmente, que genera un valor, que llamaremos variable dependiente, denotada por y. En tal caso, escribimos

$$y = f(x)$$

Tomando en cuenta lo anterior, es que ingresar una función de una variable en MATLAB es relativamente simple. Para ello seguimos los siguientes pasos. Supongamos que queremos ingresar la función definida en el **Ejemplo 1.** mediante la ventana *Command Window*. Luego escribimos,

```
>> x=[-1;0.01;1];
>>
```

El vector ingresado anteriormente tiene como valor mínimo a -1 y valor máximo a 1. Los dos *puntos y comas* y el número 0.01 significan que el vector x se generará desde el valor -1 (valor mínimo) con un salto de 0.01 hasta el valor máximo 1. Es decir, que -1 será el primer elemento de x, -0.99 será el segundo y así sucesivamente, hasta llegar al antepenúltimo que será 0.98, el penúltimo 0.99 y el último será 1.

Después de esto generamos la variable y como la función $f(x) = x^2,\ x \in [-1, 1]$ como sigue,

```
>> y=x.^2;
>>
```

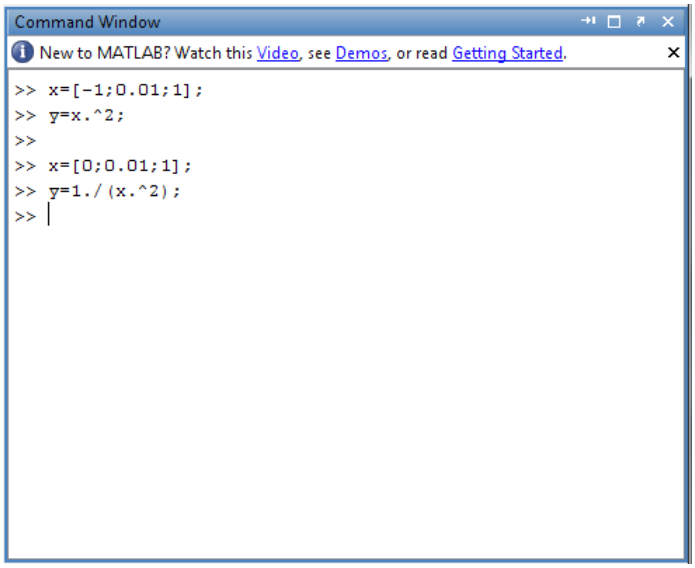
Como se explicó en el capítulo anterior.

Una dificultad es si se quisiera agregar la función

$$g(x) = \frac{1}{f(x)},\ f(x) \neq 0$$

En tal caso, haciendo una analogía a la división elemento a elemento entre vectores en MATLAB, si $f(x) = x^2$, entonces ingresamos

```
>> x=[0;0.01;1];
>> y=1./(x.^2);
>>
```



3.2 Algunas funciones de una variable de MATLAB

Muchas funciones muy útiles y frecuentemente usadas están ingresadas en MATLAB por defecto. La siguiente tabla muestra alguna de las mas conocidas.

$f(x)$	Comando en MATLAB
$\sqrt{x}$	`sqrt(x)`
$\cos(x)$	`cos(x)`
$\text{sen}(x)$	`sin(x)`
$\tan(x)$	`tan(x)`
$\arctan(x)$	`atan(x)`
e^x	`exp(x)`
$\ln(x)$	`log(x)`
$\log(x)$	`log10(x)`
$\cosh(x)$	`cosh(x)`
$\sinh(x)$	`sinh(x)`
$\tanh(x)$	`tanh(x)`

Table 1: Comandos de funciones de una variable en MATLAB.

4 Creando gráficos de funciones de una variable en MATLAB: El comando *plot*

Junto con ingresar funciones a MATLAB, también es posible graficar tales funciones aplicando comandos ya existentes en MATLAB.

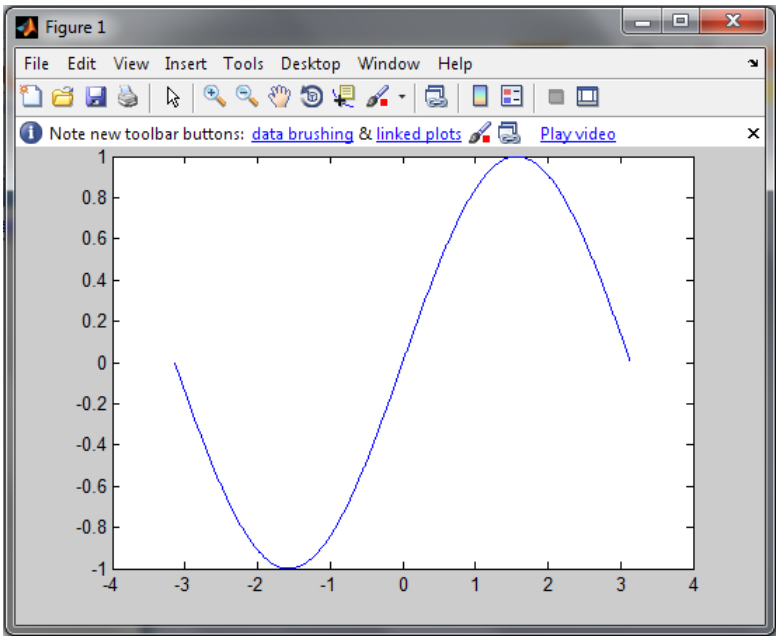
El comando *plot* es uno de los más usados para representar funciones de una variable ingresadas en MATLAB. Su uso es muy simple y sólo se reduce a ingresar el comando `plot`. Primeramente debemos ingresar las variables independiente x y dependiente y a matlab. Supongamos que queremos ingresar la función

$$f(x) = \sin(x),\ x \in [-\pi, \pi]$$

Entonces ingresamos

```
>> x=[-pi:0.01:pi];
>> y=sin(x);
>> plot(x,y)
```

El orden de escribir las variables en el comando plot es muy importante, pues por defecto este comando toma como variable independiente, a la variable ingresada en la primera posición, y como variable dependiente, a la variable ingresada en la segunda posición.



Observar que el color de la gráfica de la función $\sin(x)$ es azul. Este es el color que aplica por defecto el comando `plot`. Para poder cambiar el color de la gráfica sólo debemos hacer el cambio

```
>> plot(x,y,'r')
```

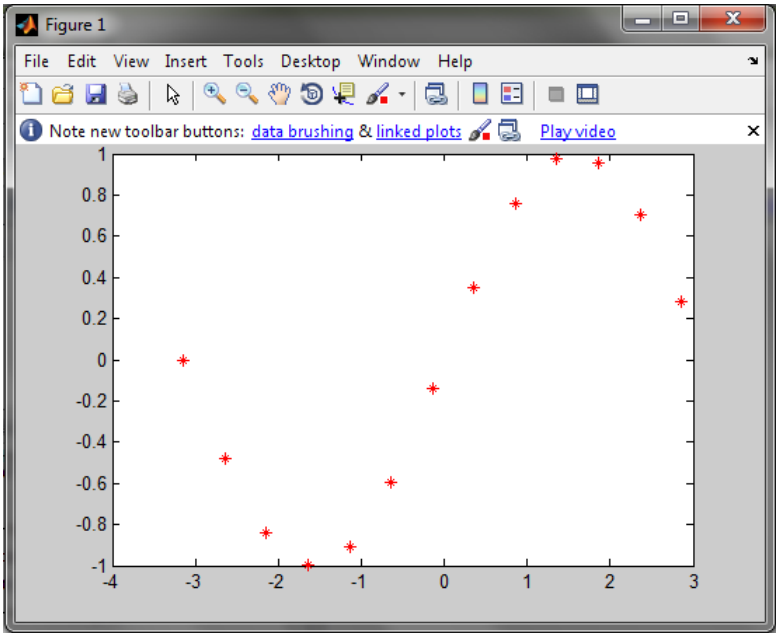
La letra `r` significa que la gráfica aparecerá de color rojo.

También es posible cambiar la forma en la que el comando dibuja la función. Por defecto, `plot` une dos puntos por una línea recta continua. También podemos hacer un cambio para que, por

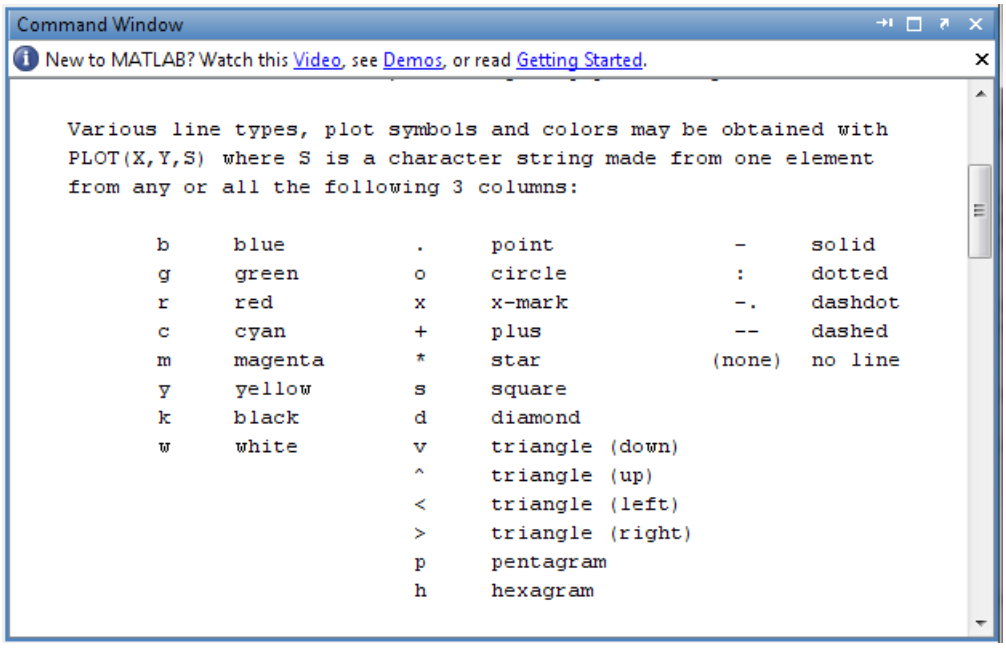
ejemplo, el plot una dos puntos por una línea punteada, o que por cada punto dibuje alguna forma que contiene este comando. Por ejemplo si escribimos

```
>> x=[-pi;0.5:pi];
>> y=sin(x);
>> plot(x,y,'*r')
```

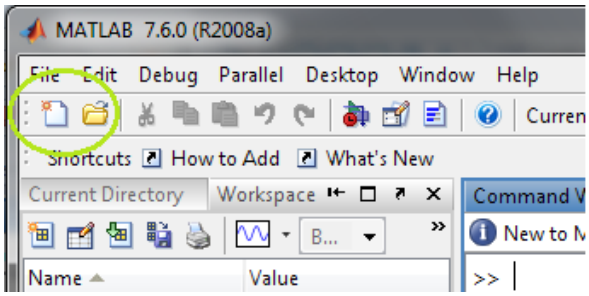
La gráfica será de color rojo, y en cada punto (x, y) se dibujará un $*$.



La siguiente figura muestra todas las opciones disponibles antes descritas.



5 Trabajando con el editor de MATLAB



Hasta ahora hemos visto cómo usar diferentes comandos usando la ventana *Command Window*. Otra buena aplicación en MATLAB es el *editor*. Es una herramienta muy utilizada en la programación más compleja y extensa. También pueden generarse matrices, funciones, gráficas y diversas salidas con esta aplicación. Podemos ingresar al editor desde las aplicaciones que se exhiben en la ventana de MATLAB, vista en el capítulo 1.

Dentro del editor mientras generamos un programa podemos comentar lo que estamos haciendo, sin que estos comentarios afecten el resultado del programa. Para ello es que para comentar un paso o definición de alguna variable, definir lo que el programa o una línea de este hará, es que se escribe al inicio de la línea que queremos comentar el signo %, y para dividir nuestro programa en diferentes secciones y agregarles un título, aplicamos el signo %%.

5.1 Comandos

En el editor de MATLAB, existen mucho comandos usandos frecuentemente y que sirven como base para la generación de progrmas básicos y complejos, dependiendo de lo que se quiera programar.

Todo comando en el editor tiene su encabezado, que lleva el nombre del comando y alguna condición que se deba cumplir, y un término, que siempre se escribe como `end`. El uso de un comando tiene entonces la siguiente estructura

```
Definiciones previas que necesitara el comando (si es necesario);

Comando (condicion)
Pasos que seguira el comando;
end
```

Una vez concluida la escritura de nuestro programa en el editor, debemos guardar el programa para poder usarlo. Para eso exhibimos las siguientes figuras.
Luego para poder ocupar nuestro programa, sólo debemos ingresar en la ventana *Command Window* el nombre de nuestro programa, si se trata de un *Rutero*, o el nombre del programa más las variables que necesita para poder operar nuestro programa, si se trata de una *Function*.

5.1.1 Comando lenght

Un comando muy básico pero no menos importante el el comando *lenght*, le cual determina el largo de un vector definido antes. Es muy útil al momento de por ejemplo querer utilizar el comando *for* o el comando *if* para generar funciones, entre otros. Si por ejemplo tenemos ingresado el vector `x=[1;0.5;10];`, entonces usando el comando *lenght* podemos definir como m el largo del vector, que será el número de elementos que tenga el vector, es decir, que `m=lenght(x);`.

5.1.2 Comando while

Este comando nos permite generar una relación recursiva entre distintas variables que obedecen a una condición definida antes. Por ejemplo si quisieramos llenar un vector vacío, que en MATLAB se ingresa como `x=[];`, de números naturales desde 1 hasta un n dado, escribimos en el editor

```
x=[];
n=10;
k=1;
while(k<=n)
x(k)=k;
k=k+1;
end
```

Un elemento muy importante de este comando es el de renovar el *contador*, que lo definimos como la variable del comando, que en el caso anterior es k. Cada cálculo que realiza el comando se llama *iteración*. En cada interación notar que el contador en el caso del comando *while* debe ir renovándose, según la finalidad que el programa tenga.

5.1.3 Comando for

Es un comando muy usado. Tiene la misma finalidad que el comando *while*, pero su uso es mucho más simple pero también más limitado. La condición de este comando es el de que un contador varíe entre dos números, según la finalidad del programa. En este caso, no es necesario renovar el contador, pues el comando lo hace automáticamente, una vez realizada la iteración correspondiente.

```
x=[];
n=10;
for k=1:n
x(k)=k;
end
```

Cuando escribimos $k = 1 : n$, el contador varía desde 1 hasta n de uno en uno, es decir, que toma los siguientes valores

$$1, 2, 3, \ldots, n-1, n$$

5.1.4 Comando if, else y elseif

Estos comando son útiles cuando se necesita construir, por ejemplo un vector o matriz, alguna función definidas por partes.

Ejemplo 2. Consideremos la función $f : [0, 2] \to \mathbb{R}$ y la matriz $A = (a_{ij})$ definidas por

$$f(x) = \begin{cases} 1, \text{ si } 0 \leq x < 1 \\ x, \text{ si } 1 \leq x \leq 2 \end{cases}, \quad A = \begin{cases} 1, \text{ si } i > j \\ 0, \text{ si } i \leq j \end{cases}$$

Entonces para ingresar la función f y la matriz A usamos estos comandos.

5.1.5 Comando zeros, ones y rand

Son comandos muy usados al momento de por ejemplo resolver sistemas de ecuaciones, tema que no se tocará en este curso, o también para construir matrices, vectores y/o funciones. Los comandos *zeros* y *ones* tienen como entrada el largo del vector de ceros y unos que necesitemos, respectivamente. Por otro lado, el comando *rand* también tiene como entrada el largo del vector que necesitemos construir, pero cada elemento de este vector posee un número comprendido entre 0 y 1, pero en forma aleatoria. La siguente figura muestra vectores generados con estos comandos en la ventana *Command Window*.

5.2 Rutero

Escencialemente un *rutero* no es más que una secuencia de condiciones que se deben cumplir dados valores iniciales particulares, es decir, que un rutero no depende de variables externas al propio programa. Por ejemplo si quisieramos graficar una función, como la del ejemplo 1 del capítulo 3, podemos crear un rutero para que haga esta tarea, de la siguiente forma

```
%% RUTERO %%
x=[-1:0.01:1];
y=x.^2;
plot(x,y,'r')
%% FIN RUTERO %%
```

Para poder utilizar el rutero anterior, sólo debemos guardarlo. Esta vez lo guardemos con el nombre de plot1. Así, para utilizarlo usamos la ventana *Command Window* y escribimos

```
>> plot1
>>
```

Y aparecerá el plot de la función $f(x) = x^2$. Otra forma dde poder probar nuestro rutero es utilizar la opción *asdasd* que aparece en la parte superior del editor, sin que tengamos que tengamos que cambiar al a ventana *Command Window*.

5.3 Function

Básicamente un rutero y una *function* cumplen el mismo objetivo, pero la gran diferencia entre ellas, es que la *function* requiere de variables externas para poder emplearse. Esto hace que este comando sea muy utilizado en la creación de programas mucho más elaborados y generales que un rutero, pues en la sección anterior vimos que un rutero nos podía servir para graficar la función $f(x) = x^2,\ x \in [-1, 1]$, mientras que para graficar cualquier función, podríamos crear una *function* que tenga como variable de entrada la función que queramos graficar y su variable independiente x. Como se vió en la sección de Comandos, cada uno de ellos comenzaba con su nombre, una condición y siempre terminaba con end. Para el comando *function*, es igual que a tales comandos. Su estructura sería

```
function (nombre de la funcion y de las variables externas)

cuerpo de la funcion

end
```

Por ejemplo si quisieramos crear una función que dado un vector x, multiplique cada uno de sus elementos podríamos hacer la siguiente *function*

```
function m=multi(x)
n=lenght(x);
m=1;
k=1;
while k<=n
      m=m*x(k);
      k=k+1;
end
end
```

Igual que para usar el rutero, debemos guardar el programa creado. En este caso, debemos guardar la function con el nombre que definimos la función misma, que en este caso es multi. Para poder utilizar nuestra función vamos a ventana Command Window y escribimos

```
>> x= [1 2 3 4];
>> multi(x)
```

6 Referencias

1.- *Aprenda MATLAB 7.0 como si estuviera en primero, Escuela politécnica de Ingenieros Industriales, Universidad Politéctnica de Madrid.*
2.- *Manual básico de MATLAB, Apoyo a investigación y Docencia, Servicios informáticos UCM.*