

## **PRÁCTICA 1: INTRODUCCIÓN AL SOFTWARE MATLAB**

### **1. Descripción de la práctica**

En esta práctica se desarrollará un tutorial para comenzar a utilizar el software Matlab y las distintas herramientas que éste nos ofrece. Al realizar esta práctica se pretende que el estudiante pueda emplear el programa para: Reconocer y emplear la ventana de comandos, el editor y el espacio de trabajo del programa, nombrar y declarar distintos tipos de variables, realizar operaciones entre matrices/vectores y acceder a cada uno de sus elementos, usar algunas de las funciones del programa y entender su sintaxis.

### **2. Objetivos**

- Familiarizar al estudiante con la interfaz del programa.
- Aprender a nombrar e introducir diferentes tipos de variables.
- Aprender a manipular las variables de forma matricial.
- Emplear las funciones del programa para realizar operaciones matemáticas, algebraicas.

### **3. Procedimiento**

#### *Entorno de trabajo*

Al ejecutar el software Matlab se abre una interfaz con los elementos mostrados en la figura 1.

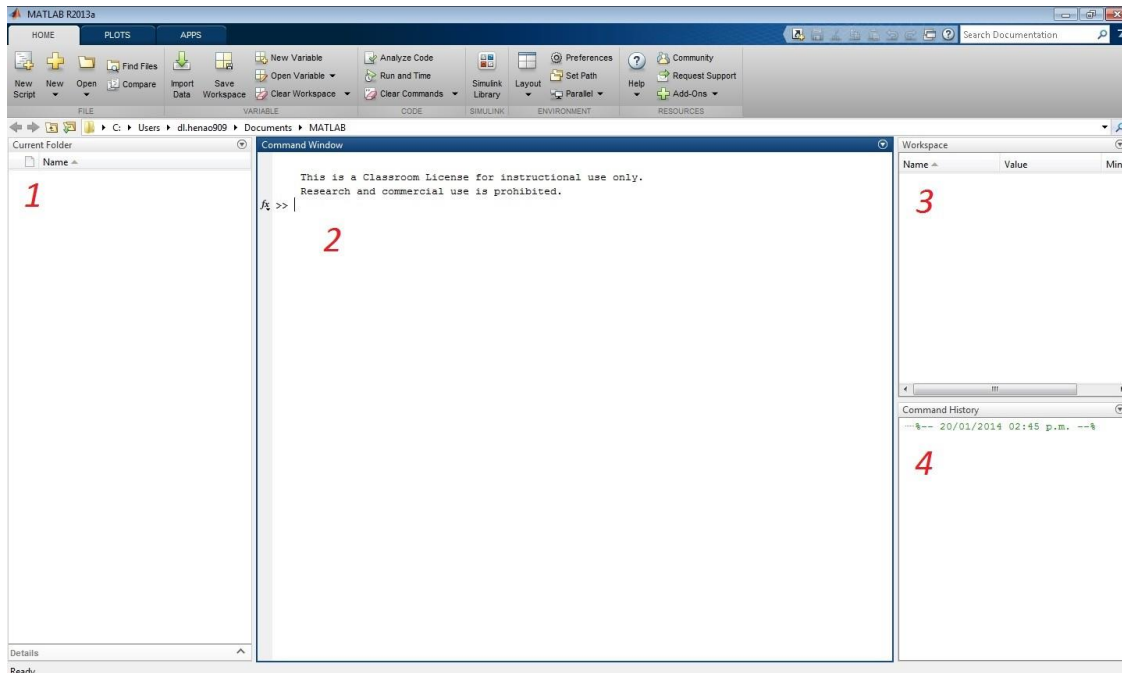


Figura 1. Interfaz de Matlab

Los cuales según la numeración de la figura 1:

**Directorio:** El directorio (ubicación) es donde se almacenan las variables y los archivos generados por Matlab, este puede ser seleccionado y modificado por el usuario. Es importante tener en cuenta en el momento de ejecutar un “Script” que éste debe estar guardado en la ubicación seleccionada, de otra manera el programa no lo encontrará.

**Ventana de comandos:** Desde allí se pueden introducir diferentes variables y ejecutar todo tipo de comandos del lenguaje Matlab (Ej. plot (x,y), sin (x), a = sym ('a'))

**Espacio de trabajo:** En éste se listan las variables insertadas al programa e información importante de cada una de ellas como el nombre, el valor, el tipo o el tamaño de cada una de ellas.

**Historial:** Allí se guardan los comandos empleados durante las diferentes sesiones de trabajo.

Cada una de estas ventanas puede ser movida, reemplazada o personalizada dentro y fuera del entorno, cambiando así la interfaz (como se muestra en la figura 2), en este tutorial se trabajará con la configuración por defecto del programa.

## Ejercicio 1

Modifique y explore la interfaz del programa. En cualquier momento para reestablecer la configuración por defecto en la pestaña: HOME-Layout-Default. (figura 3)

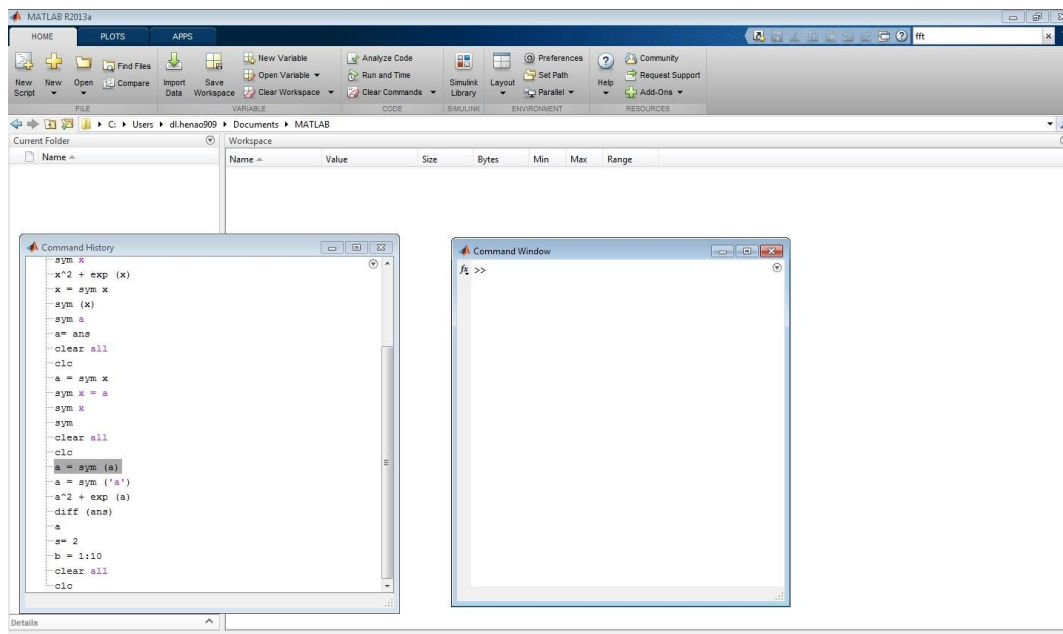


Figura 2: Interfaz personalizada. Ventanas de comandos y de historial desancladas, espacio de trabajo personalizado con columnas adicionales.

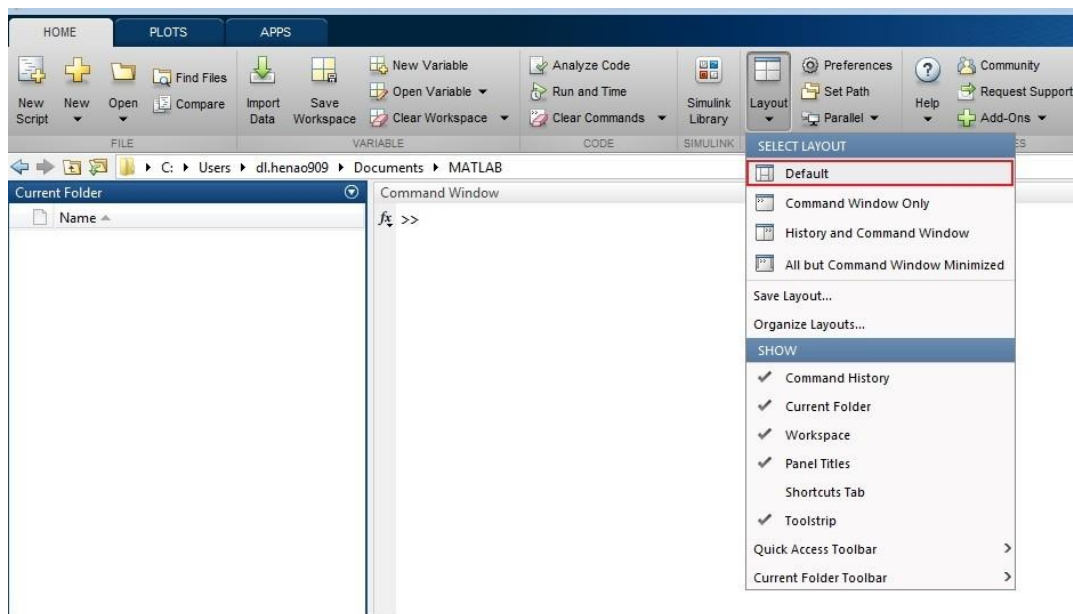


Figura 3: Reestablecer la configuración inicial.

## Variables

Matlab emplea matrices porque con ellas se pueden describir múltiples comportamientos de una forma flexible y eficiente. Las variables en Matlab y sus operaciones deben ser pensadas como un arreglo matricial, siendo un elemento (como un vector o un escalar) una parte de la matriz.

### Nombres de las variables

Las variables se nombran empleando una cadena de caracteres sin espacios que obligatoriamente comienzan por una letra seguida de un conjunto de letras, dígitos o una combinación de éstos. Por ejemplo, f1, tiempo, mpiston, mresor3 y t son nombres de variables válidos mientras que: FU 1, tiem.po y 1ma no lo son.

Matlab permite diferenciar entre letras mayúsculas y minúsculas por lo que las variables A y a son diferentes.

## Ejercicio 2

Cree una variable que contenga el número entero “2” y nómbrala “var1” esto se logra introduciendo en la ventana de comandos:

```
var1 = 2
```

Note que en el espacio de trabajo ahora se muestra la variable var1 junto con su información.

Ahora en la ventana de comandos digite:

```
Var1 = 3
```

Dado que Matlab diferencia entre mayúsculas y minúsculas, ahora en el espacio de trabajo se muestran dos variables, “var1” y “Var1”.

Finalmente digite:

```
var1= 200
```

En este caso como el nombre de la variable es exactamente igual a la definida en el paso 1 Matlab reasigna el valor de esta por el último definido, 200.

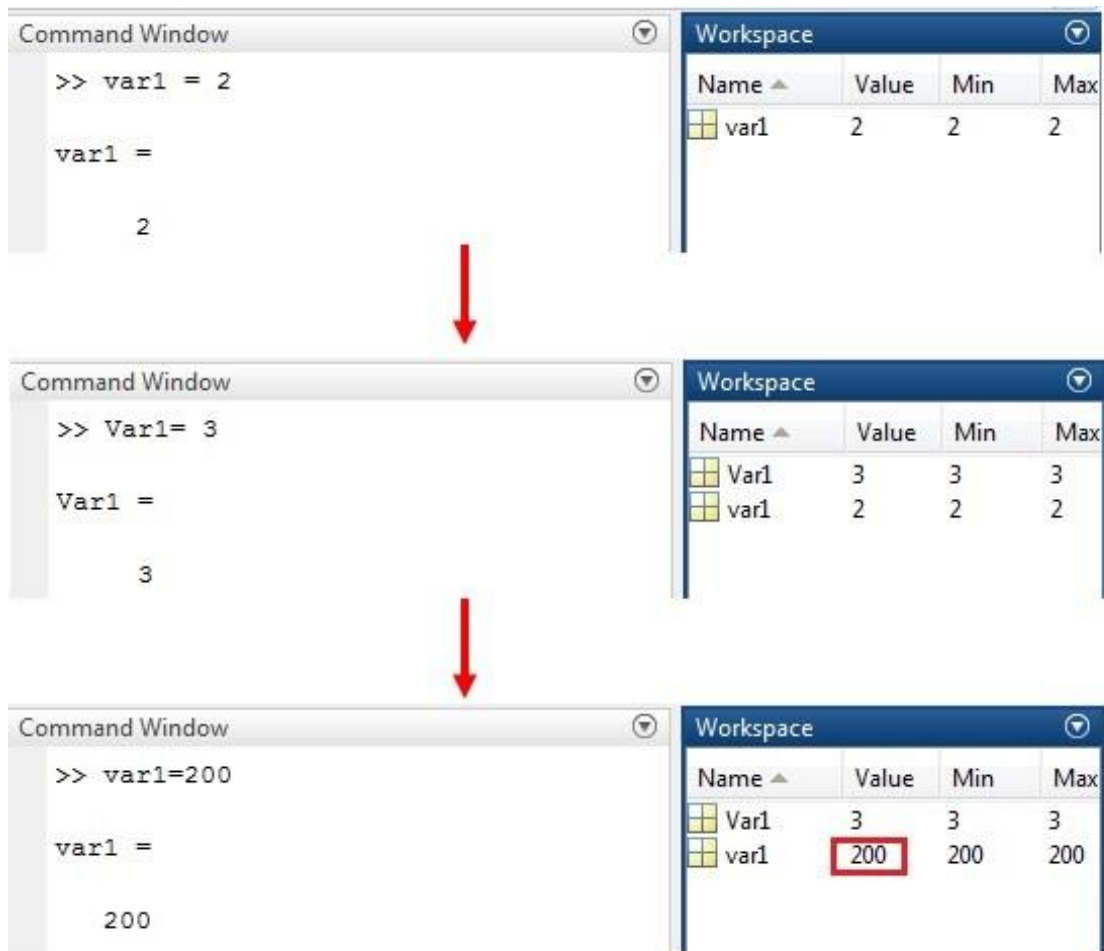


Figure 4: Ejercicio 2, nombres de las variables.

### Tipos de variables

Matlab no sólo admite variables numéricas, también se pueden introducir variables lógicas, simbólicas, cadenas de texto, estructuras, celdas, entre otras. Cada uno de estos tipos se emplea según la aplicación requerida.

### **Ejercicio 3**

Realice una limpieza de la ventana de comandos y las variables almacenadas en los ejercicios anteriores, esto se logra digitando en la ventana de comandos:

*clear all clc*

Ahora en la ventana de comandos ingrese diferentes tipos de variables así:

```
a= 1  
b= 1.5  
c= (-1)^0.5 ó power(-1, 0.5)  
d = 'Hola'  
e= true(1,1)  
f= cell(2,2)  
g= sym ('g')
```

3. En el espacio de trabajo seleccione Workspace Actions-Choose Columns-Class (figura 5)

Ahora en el espacio de trabajo se añadió una columna con información adicional del tipo de variable, observe que las variables a, b y c son numéricas (independientemente si son números enteros, decimales o imaginarios) la variable d es una cadena de caracteres, e es lógica, f es de tipo celda y finalmente la variable g es simbólica (figura 6).

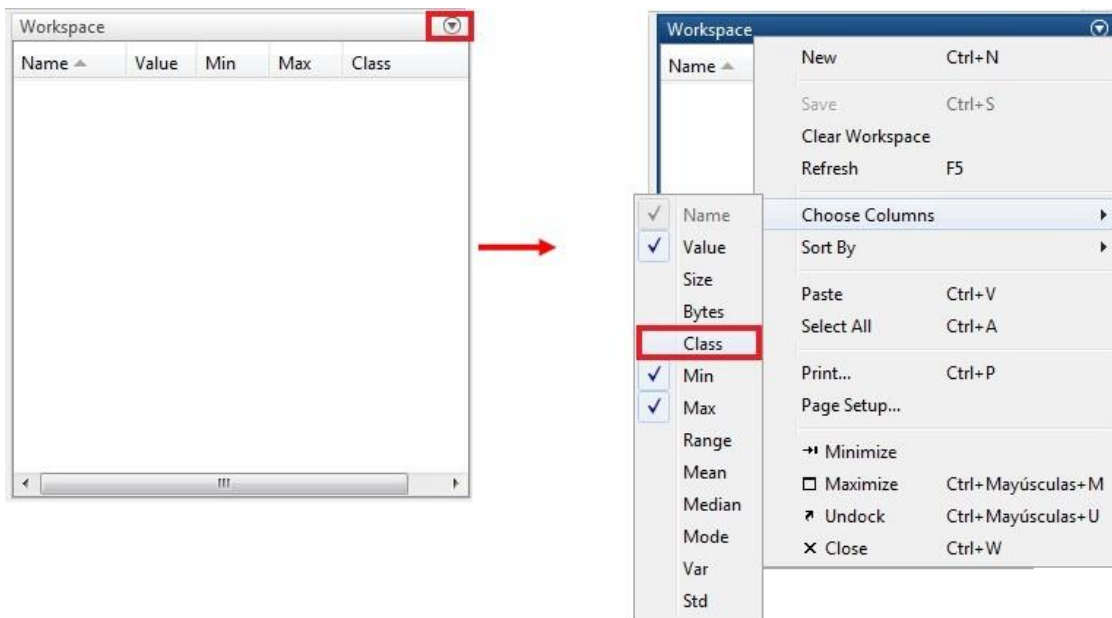


Figure 5: Añadir una columna de información al espacio de trabajo.

Name	Value	Min	Max	Class
a	1	1	1	double
b	1.5000	1.5000	1.5000	double
c	0.0000 + ...	0.0000...	0.0000...	double (complex)
d	'Hola'			char
e	1			logical
f	<2x2 cell>			cell
g	<1x1 sym>			sym

Figure 6: Columna clase en el espacio de trabajo.

Ahora emplearemos algunas de las variables declaradas en el ejercicio anterior para explorar sus aplicaciones.

#### Ejercicio 4

Podemos realizar diferentes operaciones entre las variables numéricas por ejemplo digite en su ventana de comandos

$a+b$   
 $b+c$   
 $c^2$

Y observe los resultados.

Podemos obtener la derivada de una función haciendo uso de la variable simbólica g, así:

Cree la función

$$f = g^2 + \exp(g)$$

Derive la función empleando el comando “diff”

$\text{diff}(f)$

## Vectores, Matrices

### Crear vectores, matrices

Como se había mencionado anteriormente, Matlab permite guardar variables en forma matricial y acceder a cada uno de estos elementos. En esta sección, aprenderemos a introducir un vector y una matriz de diferentes formas.

### Ejercicio 5

Introduzca un vector fila con los números del 0 al 9 con el nombre A digitando:

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

También puede realizar esta operación reemplazando los espacios con comas:

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

En caso que requiriéramos construir un vector de mayores dimensiones, por ejemplo con los números del 0 al 100, introducir número por número sería un proceso inadecuado para estos casos Matlab admite la notación 0:1:100, de esta manera se construye un vector con los números del 0 al 100 con un paso (incremento) de 1 es decir 0,1,2,3... 100. Si escribiéramos 0:2:100 el incremento sería de 2 y los elementos del vector serían 0,2,4... 100.

El paso por defecto que emplea Matlab es 1 por lo que si no definimos el paso y simplemente escribimos `A= 0:100` el programa empleará nuevamente un incremento de 1 y obtendremos el vector 0,1,2,3... 100.

Pruebe esta notación para obtener un vector que contenga los números de 0 al 100 con un incremento de 0.05, digite en la ventana de comandos

```
nuevo= 0:0.05:100
```

Observe que se creó el vector `nuevo = [0,0.05, 0.10, 0.15... 100]` el cual contiene 2001 elementos y al igual que todas las variables, además de guardarse en el espacio de trabajo se muestra en la ventana de comandos. Cuando no queremos que se muestren todos los elementos en la ventana de comandos podemos añadir un “;” al final de cada instrucción. Realice una limpieza de la ventana de comandos:

```
clc
```

Ahora digite:



`C= 0:1000;`

Observe como se almacena el vector C en el espacio de trabajo más no se muestra en la ventana de comandos.

Ahora introduciremos una matriz de 3x3 con el nombre B que contenga los números del 1 al 9:

Para crear una matriz se añaden los vectores fila uno a uno separados con un “;” así

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

O bien:

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

### Acceder a los elementos

Podemos acceder individualmente a cada uno de los elementos de un vector o matriz al igual que a cada columna o fila de esta.

### **Ejercicio 6**

Acceda al quinto elemento del vector A, escribiendo `A (5)`

Note que la respuesta que da el programa es 4 pues el valor 4 es el elemento número 5 del vector.

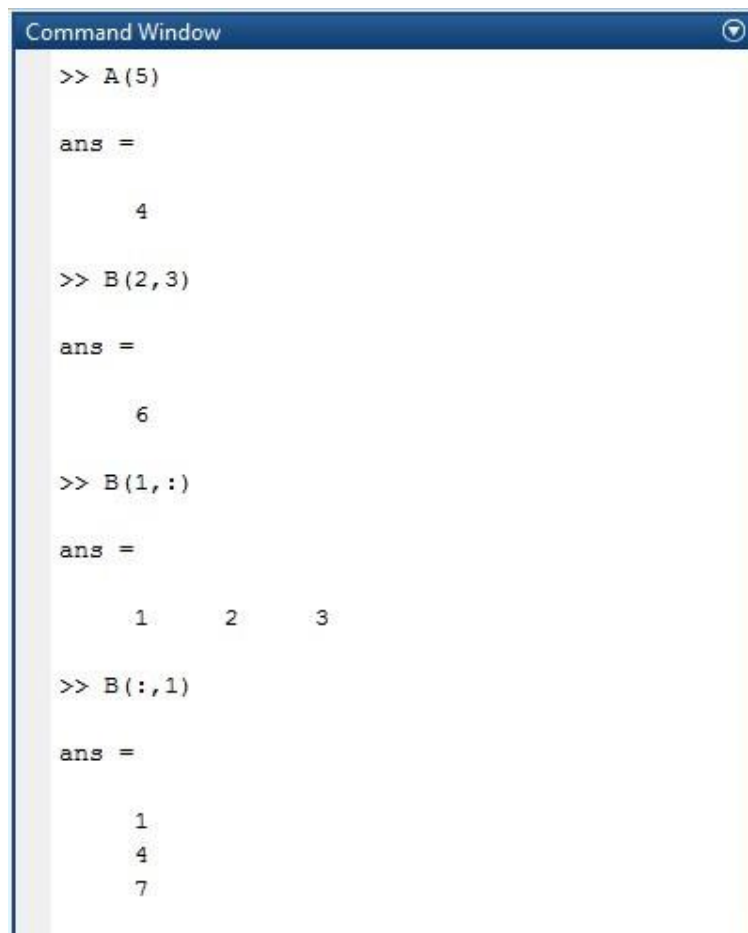
Ahora acceda al elemento en la fila 2, columna 3 de la matriz B escribiendo: `B(2,3)`

Matlab muestra el número 6 que es el elemento ubicado en la fila 2 columna 3 de la matriz.

Ahora liste toda la primera fila de la matriz B, digitando

$B(1,:)$

De manera similar, liste ahora la primera columna de la matriz B, digitando  $B(:,1)$



```
Command Window
>> A(5)
ans =
    4
>> B(2,3)
ans =
    6
>> B(1,:)
ans =
    1    2    3
>> B(:,1)
ans =
    1
    4
    7
```

Figure 7: Ejercicios 5 y 6

También puede declarar nuevas variables empleando esta notación, por ejemplo:  $t = B(1,:)$  crearía un nuevo vector llamado t que contiene los elementos 1,4 y 7.

## Operaciones y funciones de Matlab

Para poder operar entre matrices y vectores de Matlab se deben tener en cuenta las dimensiones correctas. Si multiplica una matriz A de 3x3 por una matriz B de 3x2 se obtendrá una matriz de 3x2, sin embargo si multiplica la misma matriz B por la matriz A no se podrá realizar la operación pues el número de columnas de B no coincide con el número de columnas de A y Matlab mostrará un error de dimensiones como se ilustra en la figura 8.

```
Command Window
>> A = [0 1 2; 0 1 2; 0 1 2]

A =

     0     1     2
     0     1     2
     0     1     2

>> B = [2 3;4,5;2 3]

B =

     2     3
     4     5
     2     3

>> A*B

ans =

     8    11
     8    11
     8    11

>> B*A
Error using *
Inner matrix dimensions must agree.

fx >> |
```

Figure 8: Multiplicación entre matrices

El operador punto se añade colocando un punto antes del operador y se emplea para indicarle a Matlab que se requiere que realice la operación solicitada en cada uno de los elementos de la matriz y así evitar errores de dimensionamiento.

## Ejercicio 7

Realice una limpieza de su ventana de comandos y espacio de trabajo, cree luego una matriz M de 3x3:

```
clear all clc  
M= [2 1 3; 1 7 2; 0 3 2]
```

Ahora realice las operaciones

```
M^2  
M.^2
```

Como se puede observar en los resultados  $M^2$  realiza la operación  $M*M$  es decir es una multiplicación de 2 matrices de 3x3, mientras que  $M.^2$  eleva cada uno de los elementos de la matriz M al cuadrado.

En este caso como la M es una matriz cuadrada  $M^2$  no se genera ningún error de dimensionamiento, sin embargo si por ejemplo A fuera una matriz 2x3 y no empleamos el operador punto el programa intentará multiplicar una matriz 2x3 por otra matriz 2x3 por lo que no se podrá realizar la operación como se muestra en la figura 9.