

Matlab y Octave

Módulo 1

Agenda

1. Matlab.
2. Pantalla inicial.
3. Ayuda.
4. Creación y asignación de variables.
5. Definición de vectores y matrices.
6. Operaciones matemáticas básicas.
7. Funciones matemáticas básicas.
8. Estructuras de control.
9. Scripts.
10. Comentarios e indentación inteligente.
11. Funciones.
12. Comandos varios.
13. *Octave*.

Matlab

- Es un programa desarrollado por MathWorks.
- Su nombre proviene de “*Matrix Laboratory*”.
- Es un lenguaje de alto nivel (interpretado) y un ambiente interactivo.
- Se utiliza ampliamente en instituciones académicas, centros de investigación y empresas alrededor del mundo.
- Existen versiones para Windows, Mac y Linux.
- Cuenta con una gran cantidad de librerías (conocidas como “*toolboxes*”) en áreas de ingeniería, ciencias básicas y finanzas.

Pantalla inicial de Matlab

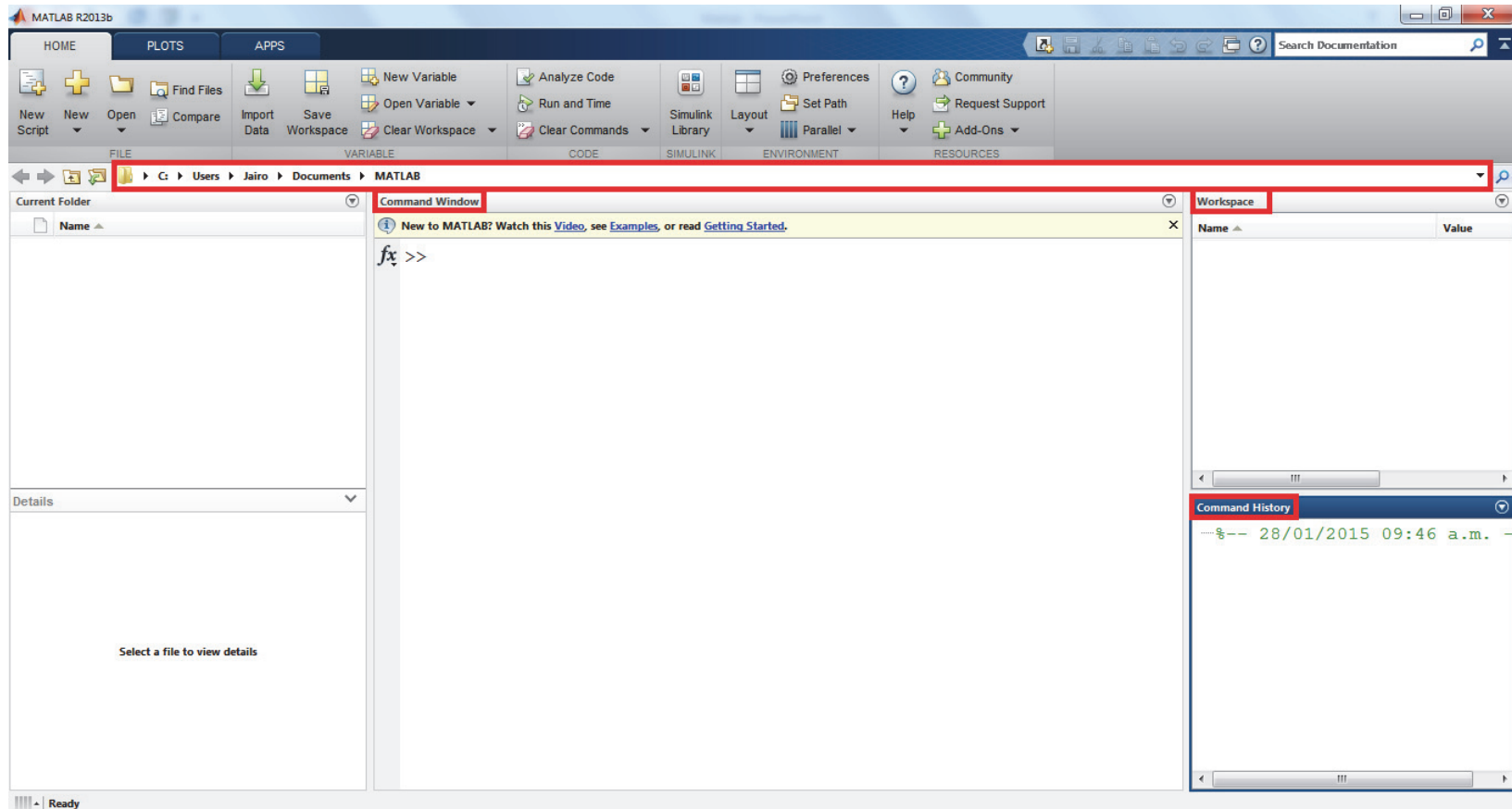


Figura 1: Current folder, command window, workspace, command history.

Cómo invocar la ayuda desde el entorno gráfico?

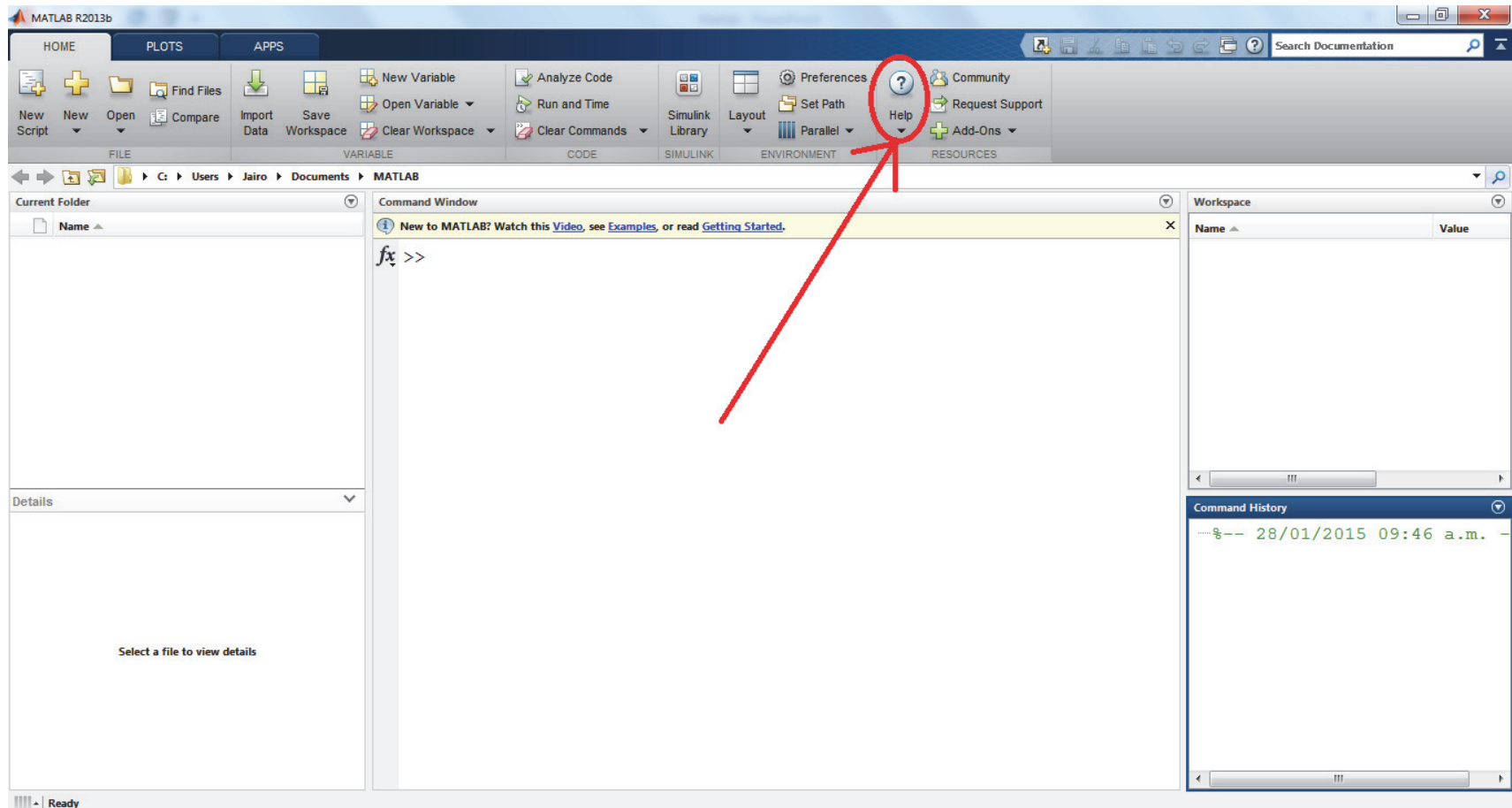


Figura 2: Ícono para invocar la ayuda gráfica.

Cómo invocar la ayuda en modo gráfico desde la consola?

- Escriba el comando *doc* seguido de lo que quiere buscar en la ventana de comandos. Ejemplo: *doc peaks*

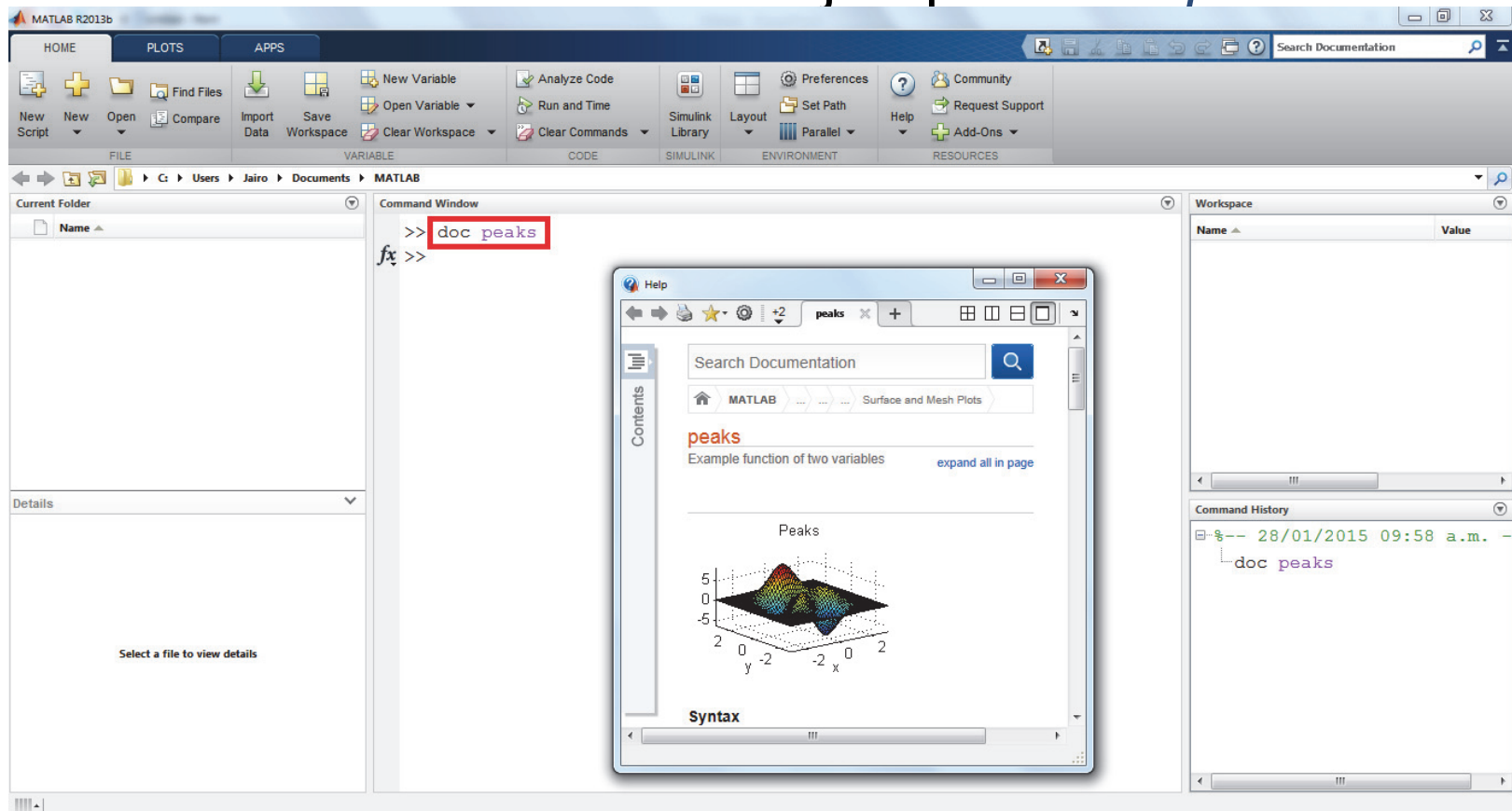


Figura 3: Ayuda gráfica desde la consola usando el comando *doc*.

Cómo invocar la ayuda en modo texto desde la consola?

- Escriba el comando *help* seguido de lo que quiere buscar en la ventana de comandos. Por ejemplo: *help function*

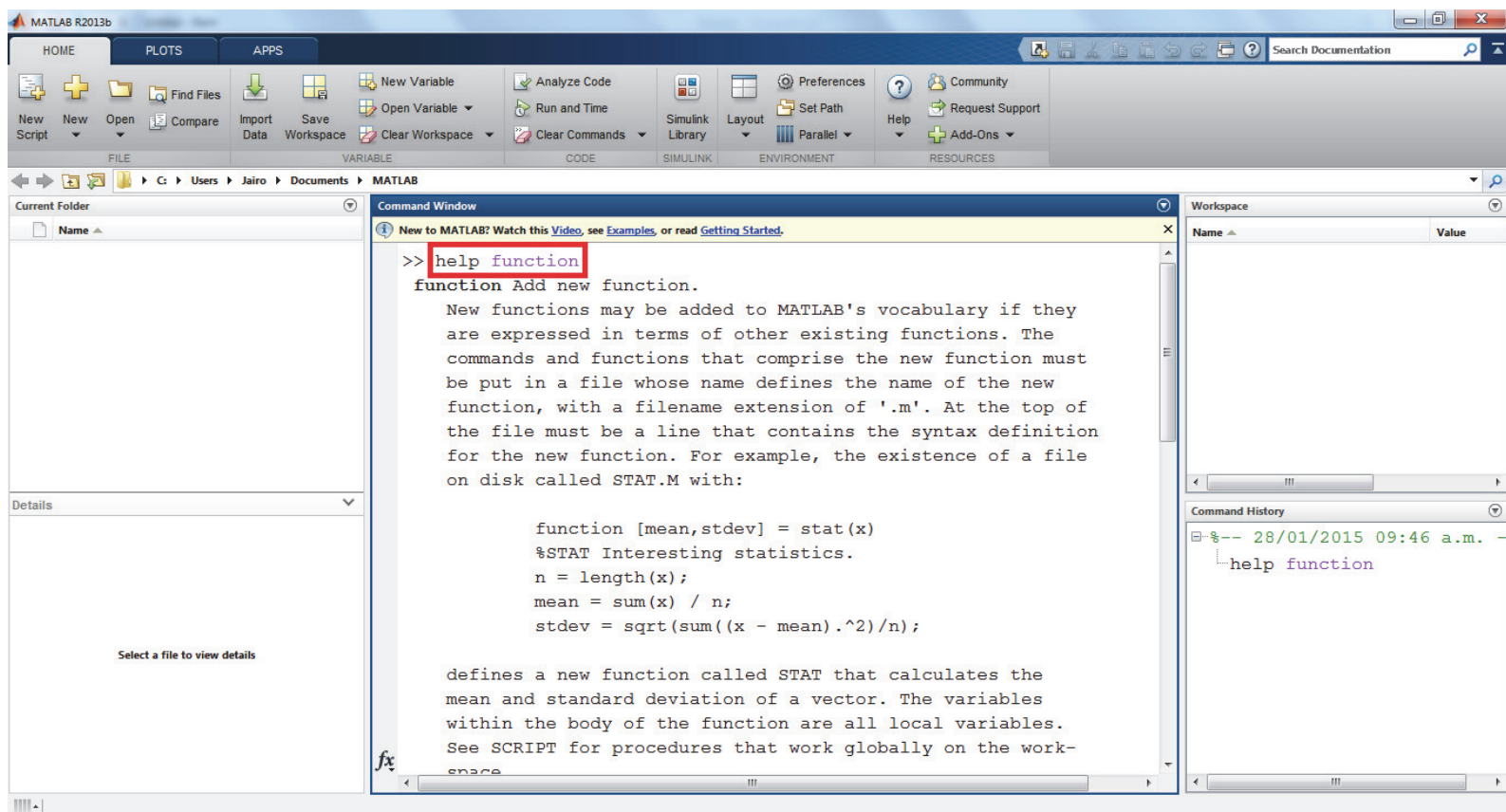


Figura 4: Ayuda textual por consola usando el comando *help*.

Cómo crear y asignar variables numéricas?

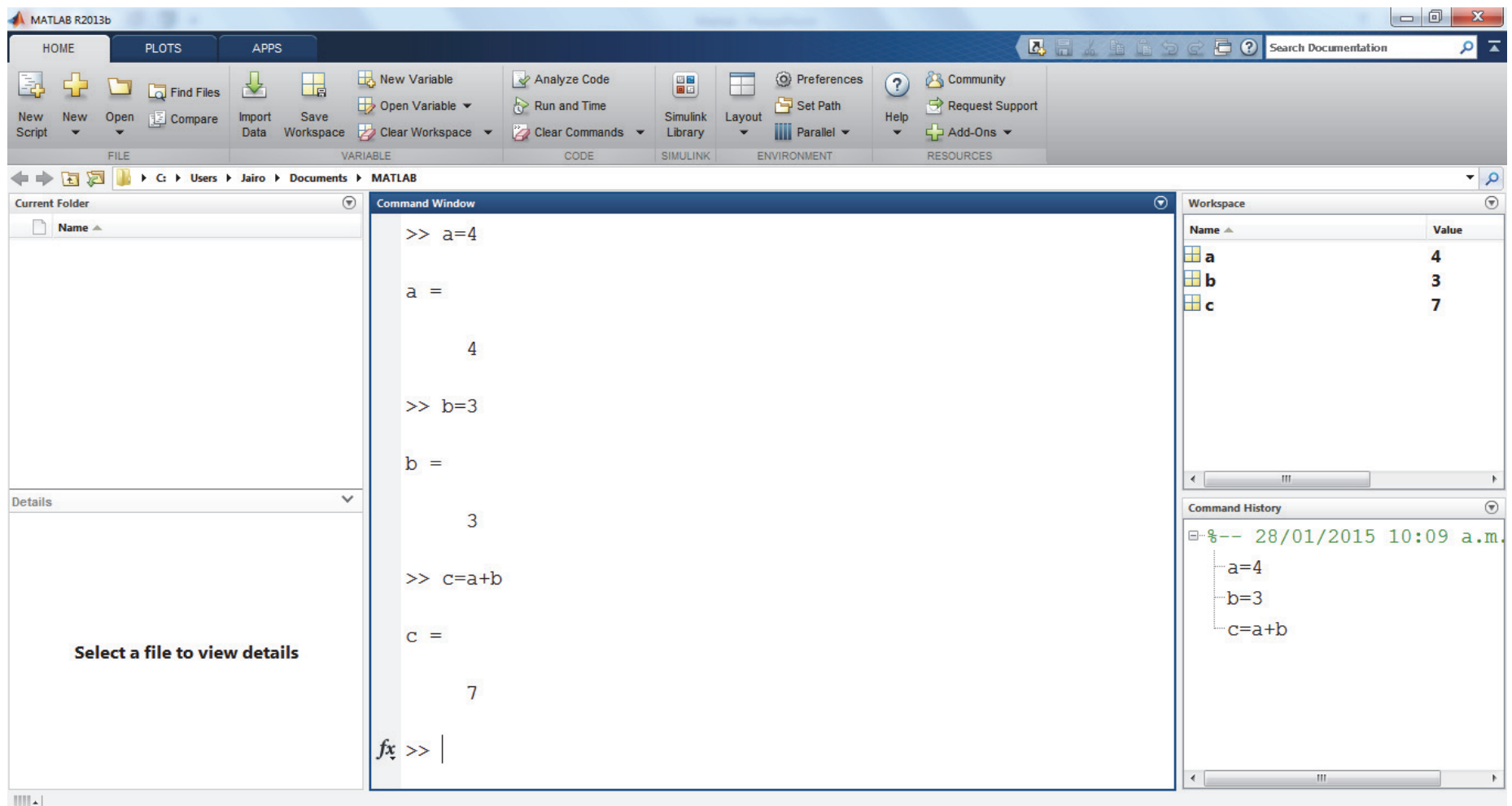


Figura 5: Asignación de variables. Note el “eco” que retorna Matlab en consola después de cada comando pues no se utilizó “;” al final del mismo.

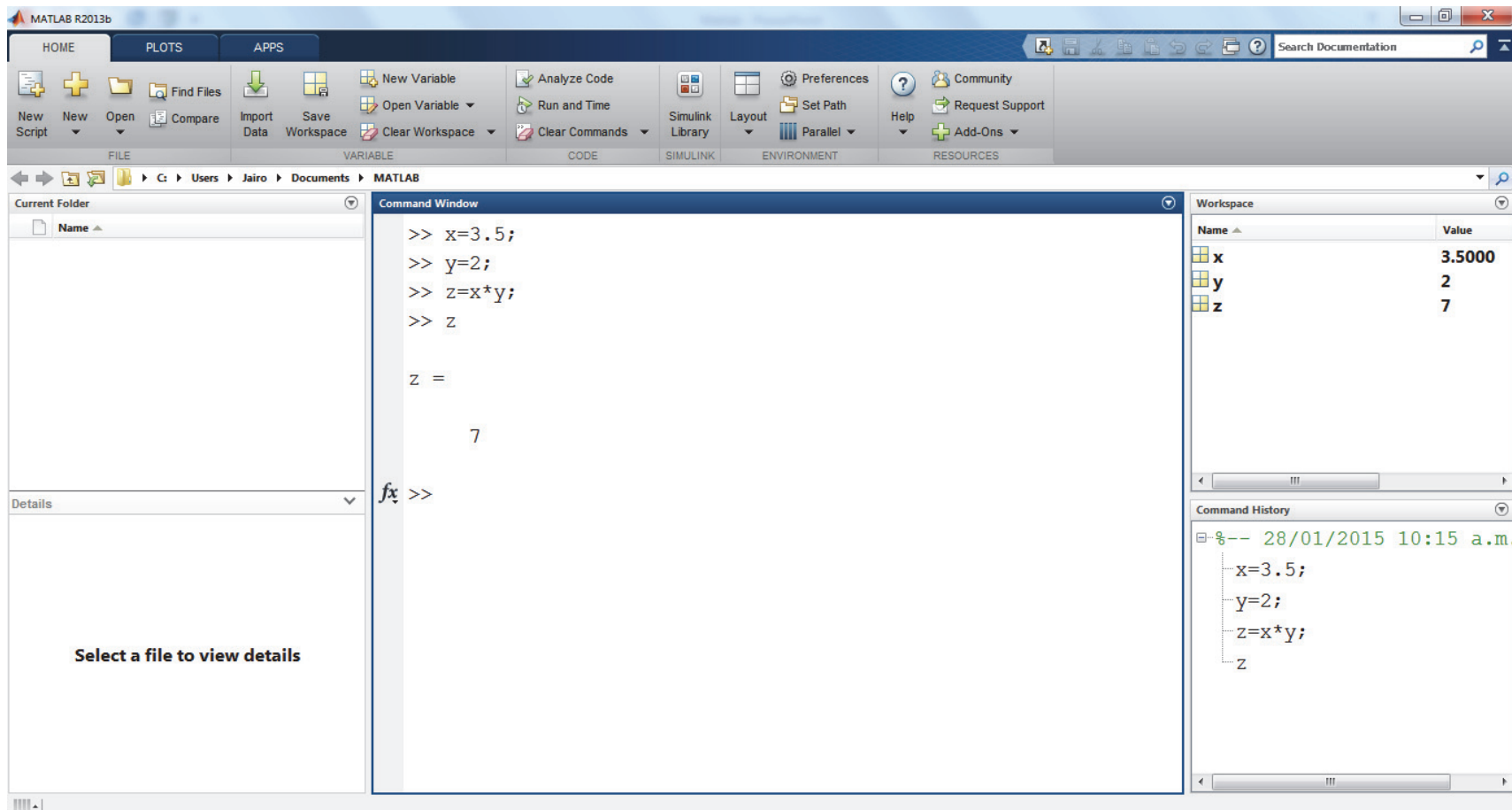


Figura 6: Asignación de variables. *Note que no es necesario indicar el tipo de la variable, por defecto Matlab la asume como double.*

Cómo crear caracteres o cadenas de caracteres y asignarlas a variables?

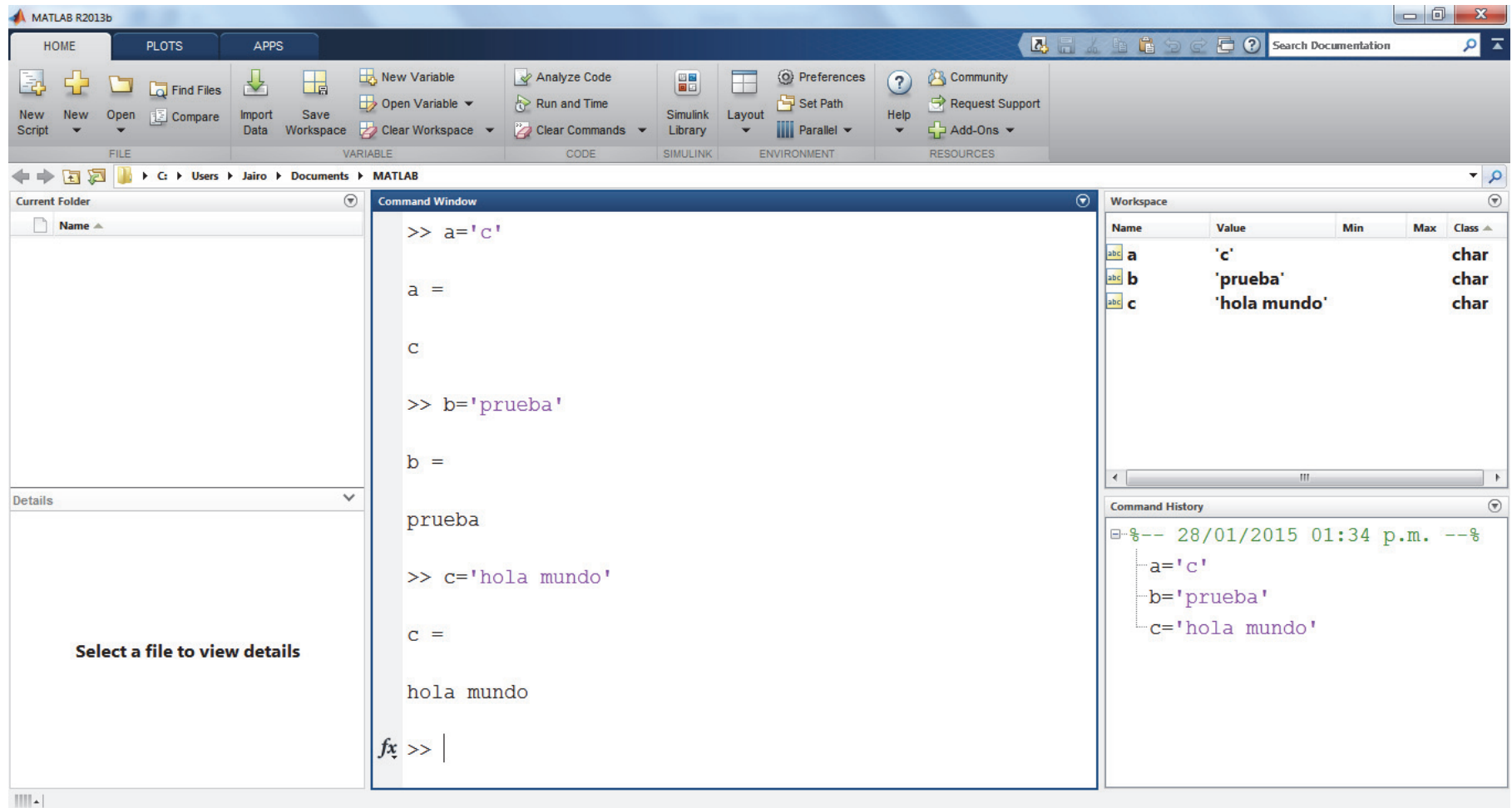


Figura 7: Asignación caracteres y cadenas de caracteres.

Observaciones

- No es obligatorio declarar la variable ni definir el tipo de datos antes de asignarle valores.
- Si usted no escribe un punto y coma (;) al finalizar el comando, Matlab mostrará el resultado en la ventana de comandos, esto se conoce como el eco.
- Mostrar resultados intermedios es útil para depurar los programas, pero hace que la ejecución sea más lenta, particularmente cuando se muestran en pantalla vectores o matrices de gran tamaño.

Cómo crear vectores fila?

```
>> a=[1 2 3 4]
```

%Usando espacios como separadores.

```
a =
```

```
1 2 3 4
```

```
>> b=[1,2,3,4]
```

%Usando comas como separadores.

```
b =
```

```
1 2 3 4
```

```
>> c=1:1:4
```

%Valor inicial:incremento:valor final.

```
c =
```

```
1 2 3 4
```

```
>> d=1:4
```

%Si no se especifica, se asume que el incremento es 1

```
d =
```

```
1 2 3 4
```

Figura 8: Creación de vectores fila.

Cómo crear vectores columna?

```
a=[1  
2  
3  
4] %Usando enter como separador.
```

```
a =  
1  
2  
3  
4
```

```
>> %Usando punto y coma como separador.
```

```
>> b=[1;2;3;4]
```

```
b =  
1  
2  
3  
4
```

```
>> d=(1:4).' %Transponiendo un vector fila.
```

```
d =
```

```
1  
2  
3  
4
```

```
>> c=(1:1:4).' %Transponiendo un vector fila.
```

```
c =
```

```
1  
2  
3  
4
```

Figura 9: Creación de vectores columna.

Cómo crear matrices?

```
>> a=[1,2;3,4]
```

a =

```
1 2
3 4
```

```
>> b=[1 2
      3 4]
```

b =

```
1 2
3 4
```

```
>> e=zeros(2,2)
```

e =

```
0 0
0 0
```

```
>> d=[1 2; 3 4]
```

d =

```
1 2
3 4
```

```
>> c=[1, 2
      3, 4]
```

c =

```
1 2
3 4
```

```
>> f=ones(2,2)
```

f =

```
1 1
1 1
```

En las funciones que retornan matrices, el primer argumento es el **número de filas** y el segundo es el **número de columnas**

Figura 10: Creación de matrices.

Cómo recuperar elementos de un vector?

- Si v se ha definido como un vector de la forma $v = [-5, 6, 8, 7, 1, 0, 3, 1]$
- Cómo recuperar el **segundo** elemento (es decir 6) y almacenarlo en la variable x ? se puede escribir:

$x = v(2)$

Sin embargo, es mejor acostumbrarse y escribir :

$x = v(1, 2)$

Donde queda explícito que se trata del elemento que se encuentra en la **primera** fila, y en la **segunda** columna.

- Suponga, que se ha definido un vector columna:

```
z=[4  
    7  
    3  
    8];
```

- Ahora se quiere recuperar el **cuarto** y **quinto** elemento de z y almacenarlos en un vector b. Para ello, se puede escribir

```
b=z(4:5)
```

Sin embargo, es mejor escribir:

```
b=z(4:5,1)
```

Donde queda explícito que se trata de los elementos que se encuentran en la **cuarta** y **quinta** fila de la **primera** columna.

Cómo recuperar elementos de una matriz?

- Los conceptos vistos para acceder a elementos de vectores fila y columna se extienden al caso de matrices.
- Por ejemplo, si se ha definido:

```
M=[4 5 6;  
    1 2 4;  
    0 3 3]
```

- Al escribir $M(2,1)$ se obtiene 1.
- Al escribir $M(2,3)$ se obtiene 4.
- Al escribir $M(1:2,2:3)$ se obtiene la matriz $\begin{bmatrix} 5 & 6 \\ 2 & 4 \end{bmatrix}$
- Al escribir $M(1,:)$ se obtiene el vector fila $[4 \ 5 \ 6]$

Cómo modificar los valores de un vector o de una matriz?

➤ Si se ha definido:

```
M=[4 5 6;  
    1 2 4;  
    0 3 3]
```

Y se escribe $M(1,2)=9$ se obtiene:

```
M=[4 9 6;  
    1 2 4;  
    0 3 3]
```

➤ Si se ha definido:

```
M=[4 5 6;  
    1 2 4;  
    0 3 3]
```

Y se escribe $M(2,:)=[2\ 4\ 8]$ se obtiene:

```
M=[4 5 6;  
    2 4 8;  
    0 3 3]
```

➤ Si se ha definido:

```
M=[4 5 6;  
    1 2 4;  
    0 3 3]
```

Y se escribe $M(1:2,2:3)=[1\ 0; 0\ 1]$ se obtiene:

```
M=[4 1 0;  
    1 0 1;  
    0 3 3]
```

Cómo determinar las dimensiones de un vector y de una matriz?

Suponga que A es una matriz. Al ejecutar el siguiente comando:

```
[m,n]=size(A);
```

La variable “m” contendrá el número de filas y “n” contendrá el número de columnas.

Suponga que v es un vector. Al ejecutar el siguiente comando:

```
e=length(v);
```

La variable “e” contendrá el número de elementos, pero NO se podrá determinar si ese valor corresponde al número de filas o de columnas.

Cuáles son las operaciones matemáticas, lógicas, relacionales y de arreglo básicas en Matlab?

Operadores matemáticos básicos

- Suma: $+$
- Resta: $-$
- Multiplicación $*$
- División: $/$
- Potencia: $^$
- Transpuesta T
- Compleja conjugada transpuesta H

Operadores matemáticos básicos de arreglo

Se aplican “elemento a elemento” entre vectores y matrices, o arreglos en general de igual tamaño.

- Multiplicación: .*
- División: ./
- Potencia: .^

Funciones matemáticas básicas

- Raíz cuadrada de x : `sqrt(x)`
- e^x : `exp(x)`
- $\log_{10}(x)$: `log10(x)`
- $\ln(x)$: `log(x)`
- $|x|$, si x es real: `abs(x)`
- $||x||$ si x es complejo: `abs(x)`
- seno(x) `sin(x)`
- coseno(x) `cos(x)`
- tangente(x) `tan(x)`
- *Mínimo de un vector v :* `min(v)`
- *Máximo de un vector v* `max(v)`
- *Desviación estándar de un vector v :* `std(v)`
- *Media aritmética de un vector v :* `mean(v)`
- *Mediana de un vector v :* `median(v)`
- *Inversa de una matriz M :* `inv(M)`
- *Pseudoinversa de una matriz M :* `pinv(M)`
- Otras funciones de interés incluye: `asin`, `acos`, `atan`, `atan2`,

Operadores relacionales

- Es igual a ==
- Es diferente de \neq
- Menor que <
- Mayor que >
- Menor o igual que <=
- Mayor o igual que >=

Operadores lógicos básicos

- AND `&&`
- OR `||`
- NOT `~`
- XOR(a,b) `xor(a,b)`

Caracteres especiales

- Dos puntos :
- Paréntesis y subíndices ()
- Corchetes []
- Llaves y subíndices { }
- Creación de manejadores de funciones @
- Punto decimal .
- Acceso a campo de estructura .
- Directorio padre ..
- Continuación ...
- Separador ,
- Punto y coma ;

- Comentario %
- Invoca un comando del sistema operativo !
- Asignación =
- Comilla simple '
- Concatenación horizontal [,]
- Concatenación vertical [;]

Estructuras de control

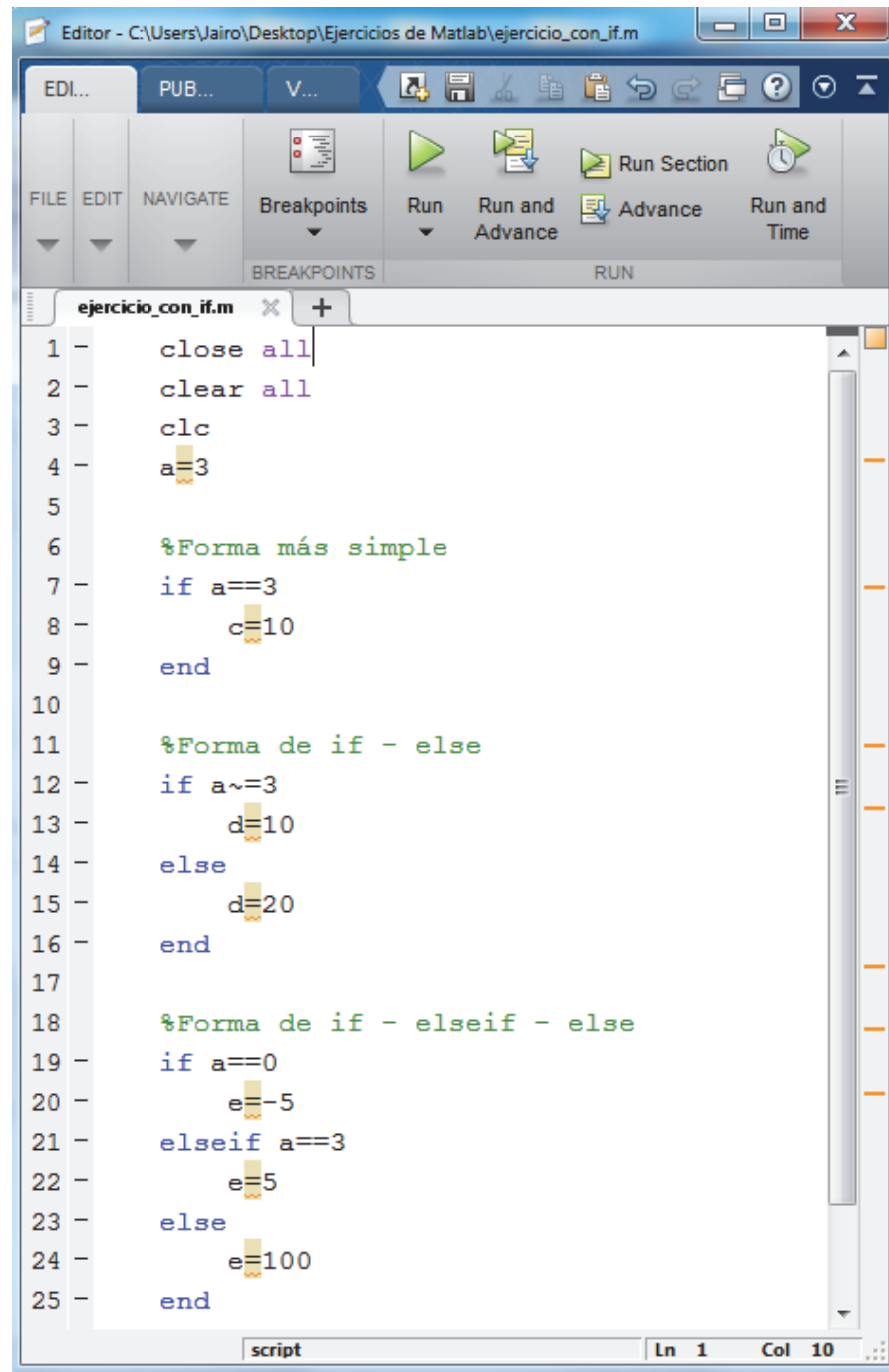


Figura 11: Ejemplo de uso de *if-else*.

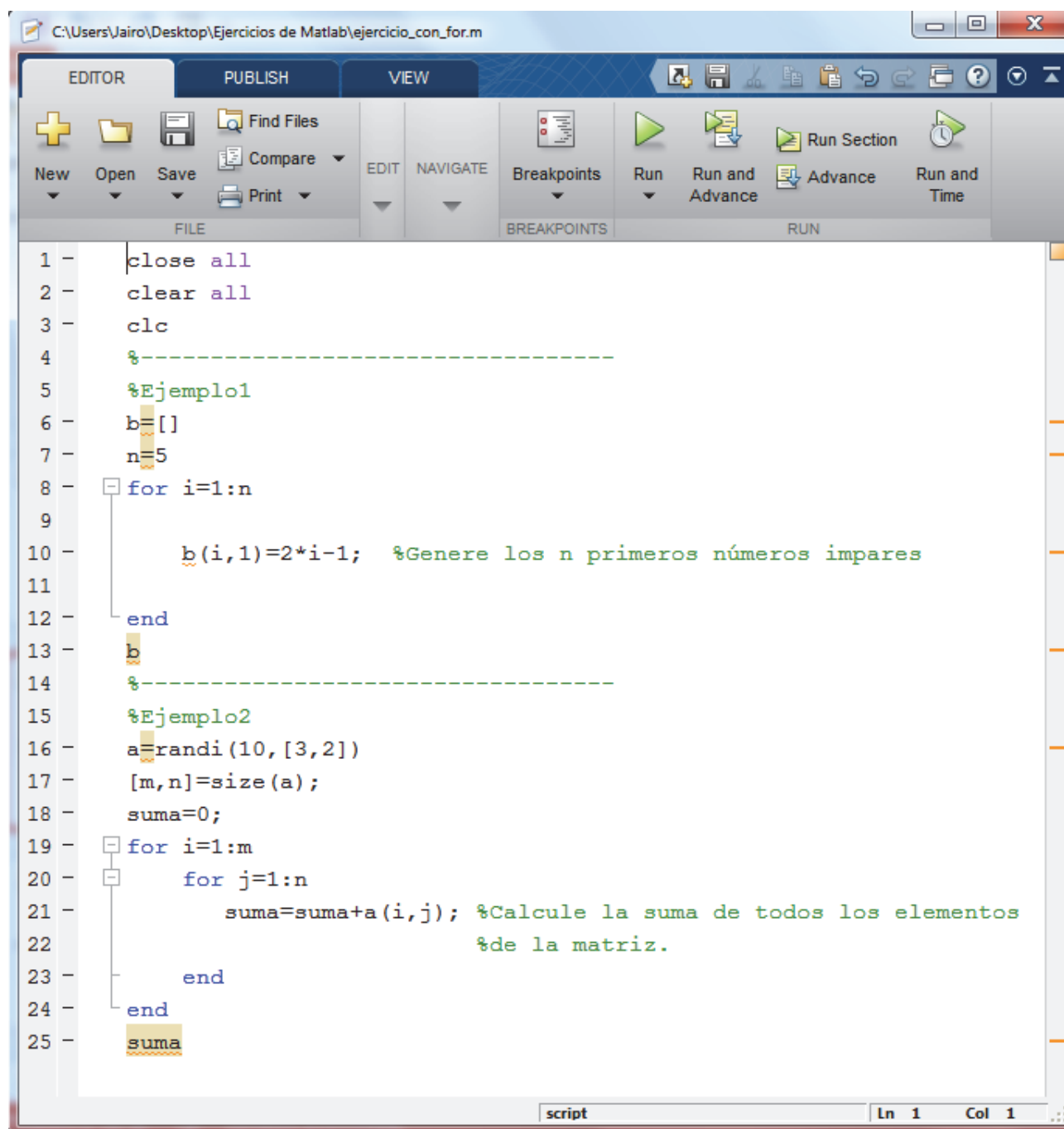
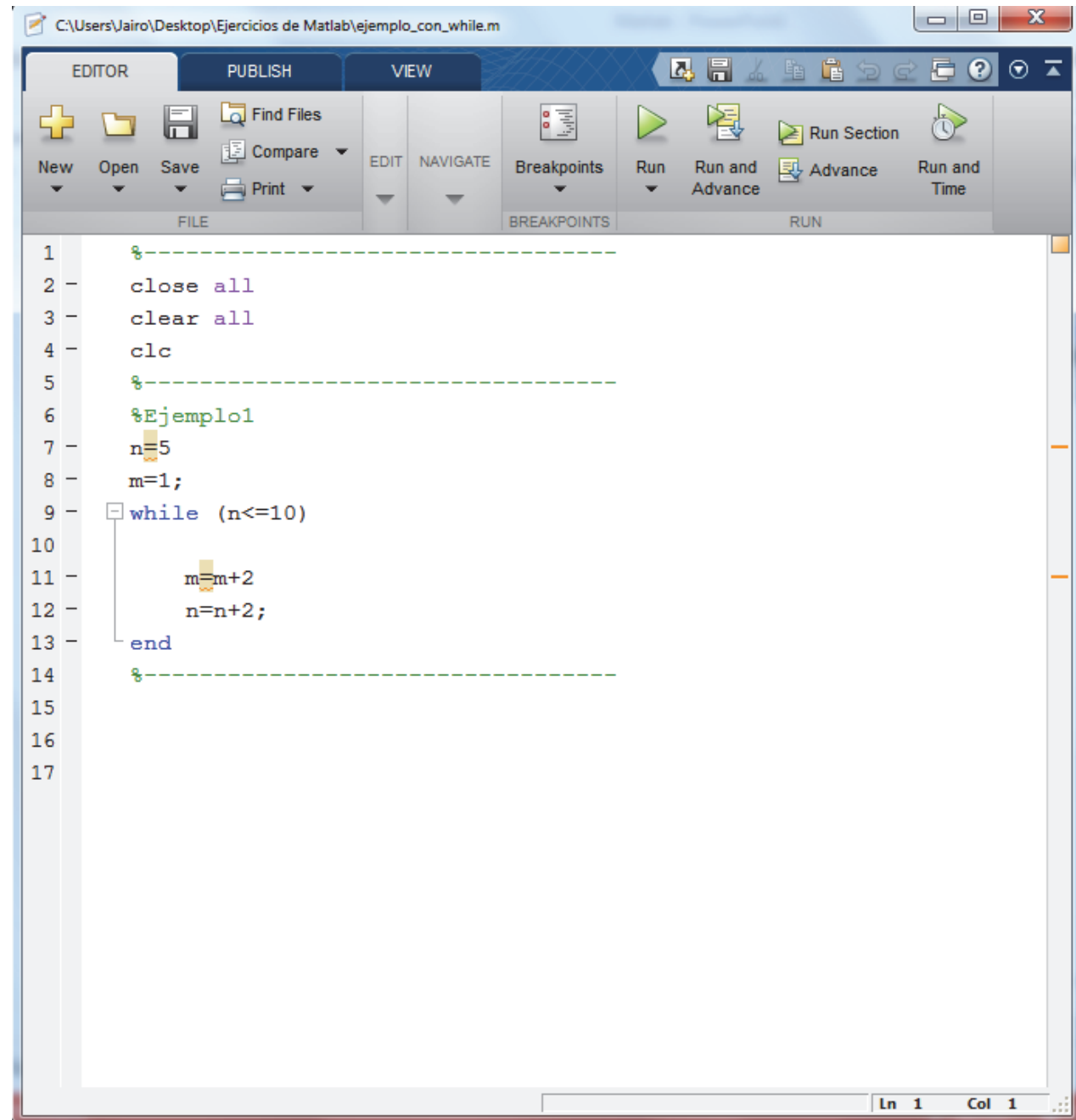


Figura 12: Ejemplo de uso del *for*.



The image shows the MATLAB Editor window with the file path `C:\Users\Jairo\Desktop\Ejercicios de Matlab\ejemplo_con_while.m`. The editor contains the following code:

```
1 %-----  
2 close all  
3 clear all  
4 clc  
5 %-----  
6 %Ejemplo1  
7 n=5  
8 m=1;  
9 while (n<=10)  
10     m=m+2  
11     n=n+2;  
12  
13 end  
14 %-----  
15  
16  
17
```

The status bar at the bottom right indicates `Ln 1 Col 1`.

Figura 13: Ejemplo de uso del *while*.

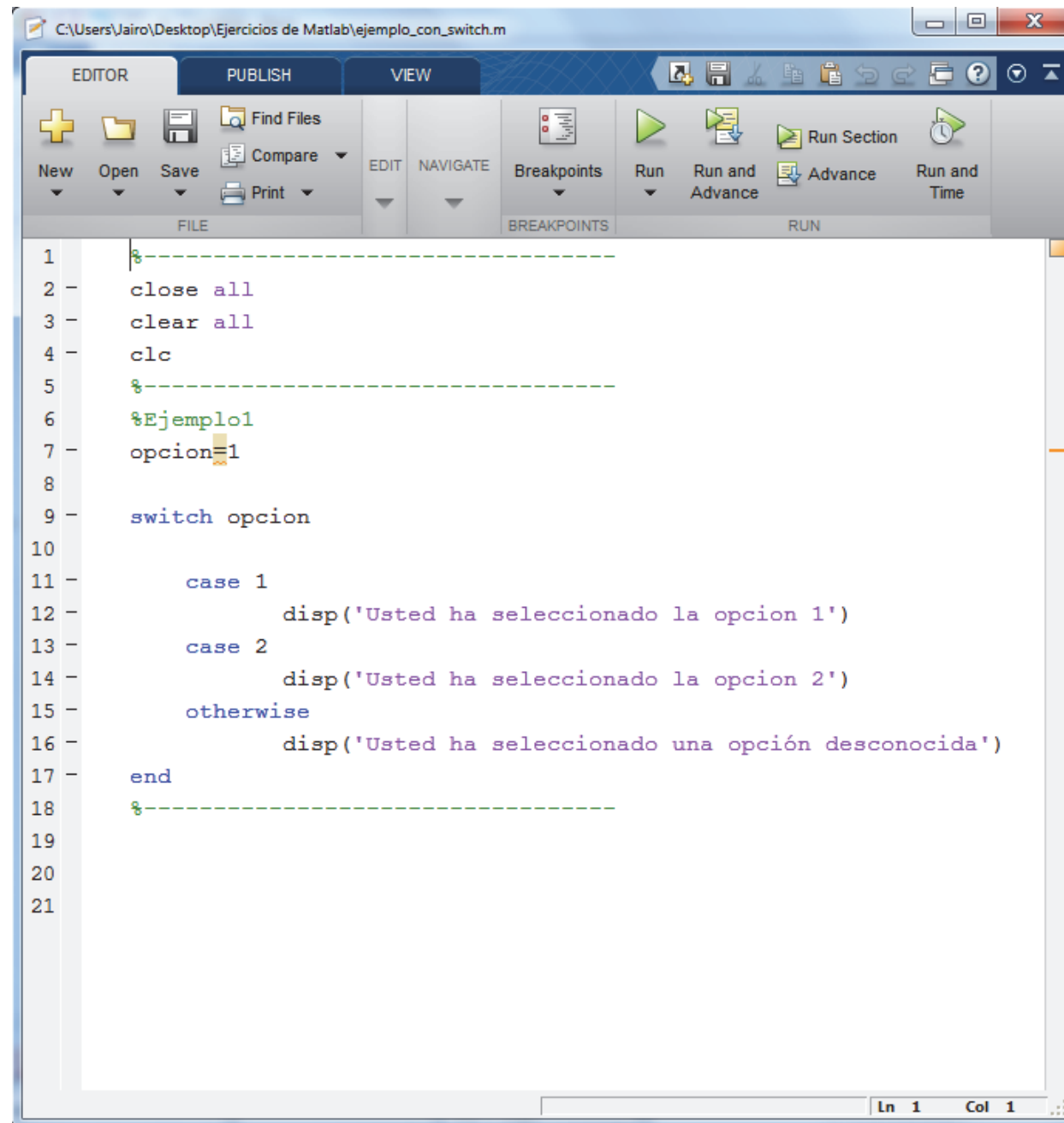



Figura 14: Ejemplo de uso del *switch*.

Scripts

- En Matlab los programas se almacenan en scripts.
- Un script es un archivo de texto plano con extensión **.m**
- Los scripts se pueden ejecutar desde la consola escribiendo el nombre del script y presionando enter o también se pueden ejecutar desde el editor de scripts haciendo click en el ícono 
- **Nota:** Recuerde que el “current folder” debe apuntar al directorio donde tiene su script, en caso contrario Matlab no podrá ejecutarlo.

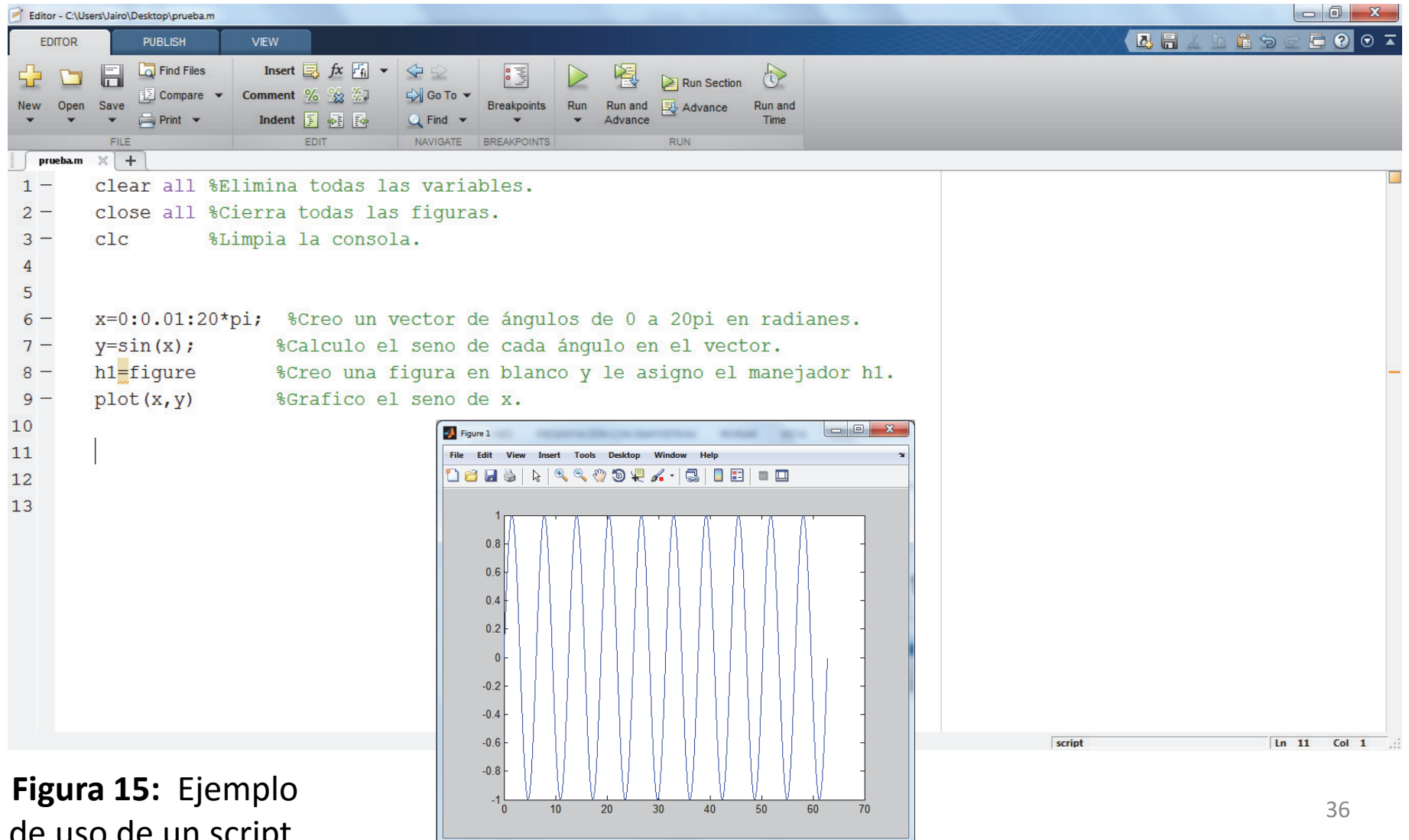


Figura 15: Ejemplo de uso de un script.

Comentarios y orden en los programas

- Cuando se selecciona una porción de texto en un script y se hace un click derecho con el mouse, aparecen opciones como *Comment* y *Uncomment*, que permiten comentar o remover los comentarios de la sección .
- *Smart Indent*, permite ajustar la tabulación de los diferentes comandos del programa de manera automática. Es útil para ajustar las buclas anidadas y las diferentes estructuras de control.

Funciones

- Una función se crea con la siguiente sintaxis:

```
function [salida1,salida2] = nombre_funcion(argumento1,argumento2)
```

- La función se debe almacenar en un archivo con el nombre exacto de la función correspondiente y con extensión .m
- La función puede recibir un número de argumentos de entrada fijo o variable y puede retornar un número de argumentos de salida fijo o variable.

Ejemplo

- Se crea una función que recibe como parámetro el radio “r” y retorna el área “a” y el perímetro “p” de una circunferencia. La función se llama **area_y_perimetro.m**

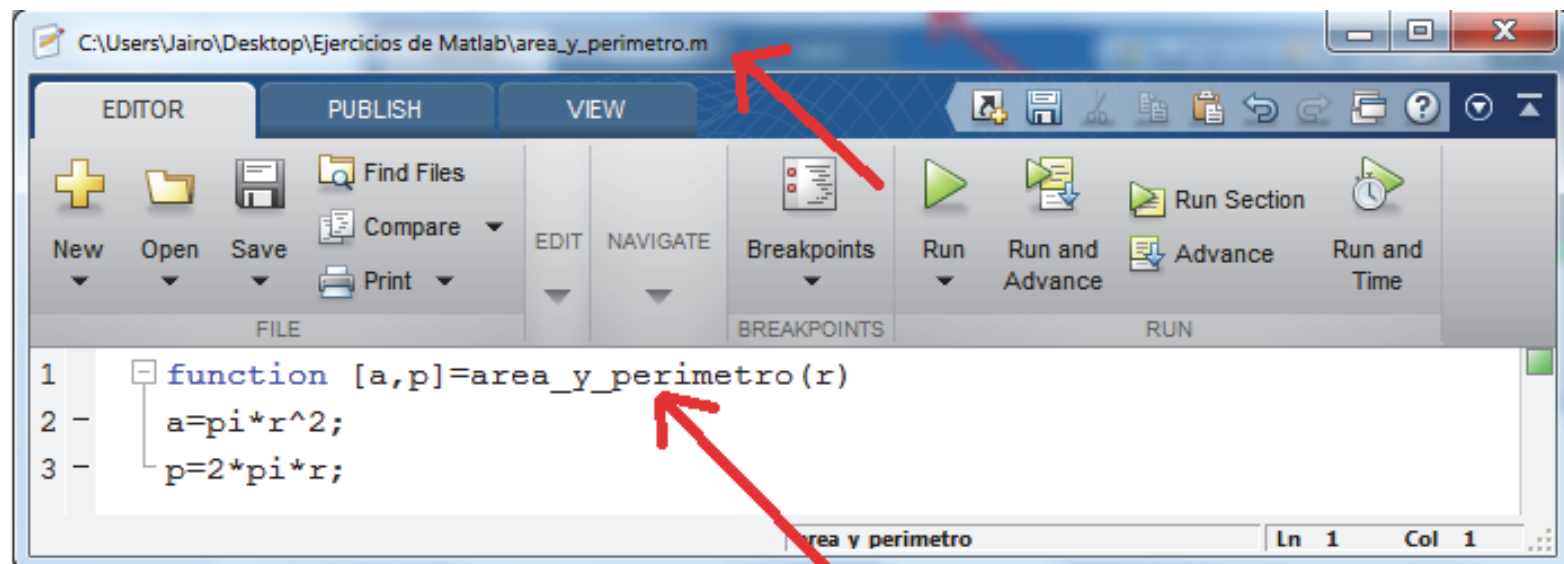


Figura 16: Ejemplo de la definición de una función.

- Se invoca la función desde la consola, desde un script u otra función:

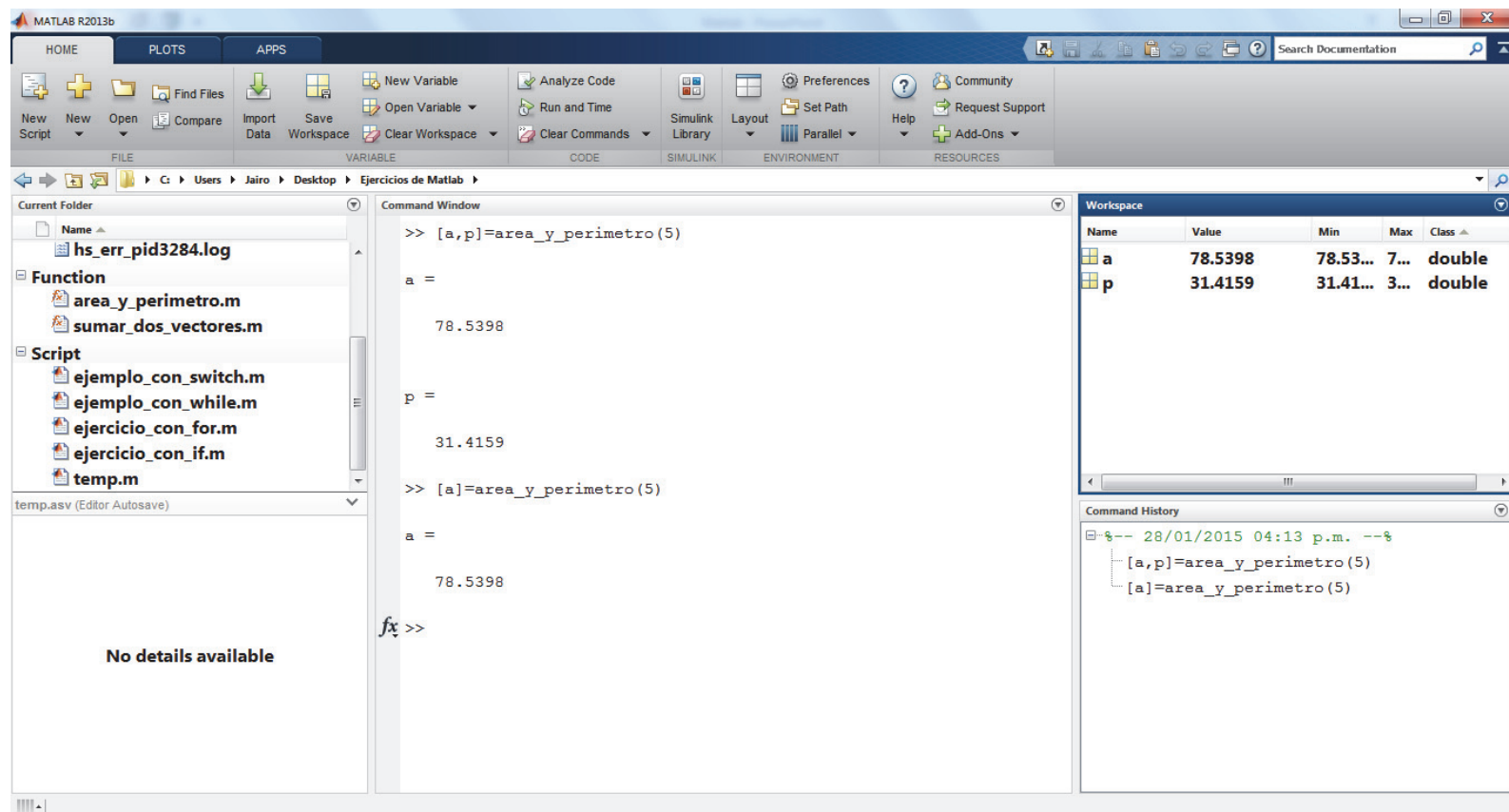


Figura 17: Ejemplo de uso de una función.

Comandos varios

- *clear variable*: borra una variable. Si se escribe *clear all* borra todas las variables.
- *clc*: limpia la consola.
- *close manejador*: permite cerrar una figura en particular. Si se escribe *close all* se cierran todas las ventanas abiertas.
- *who, whos*: permite obtener información sobre las variables que han sido definidas y sus detalles.
- *varargin, varargout*: Estos comandos permiten definir en las funciones, un número variable de argumentos de entrada y de salida respectivamente.

- *disp*: permite desplegar información en la consola.
- *reshape*: permite cambiar la forma de un arreglo, manteniendo el número total de elementos.
- *plot*: permite realizar gráficas en dos dimensiones.
- *surf*: permite desplegar gráficas en tres dimensiones.
- *fliplr*: refleja horizontalmente un arreglo.
- *flipud*: refleja verticalmente un arreglo.

- *save*: Permite guardar en disco duro las variables en un archivo de extensión .mat
- *load*: Permite cargar del disco duro las variables almacenadas en un archivo .mat
- *rand*: Permite generar números pseudo aleatorios tomados de una distribución uniforme entre 0 y 1.
- *randn*: Permite generar números pseudo aleatorios tomados de una distribución normal.
- *randi*: Permite generar números pseudo aleatorios enteros tomados de una distribución uniforme.

- *svd*: Permite encontrar la descomposición en valores singulares de una matriz.
- *eig*: Permite obtener los valores y vectores propios de una matriz (*eigenvalues* y *eigenvectors*).

Uso del comando *reshape*

```
%-----  
close all          %Cierra todas las ventanas.  
clear all          %Borra todas las variables del workspace.  
clc                %Limpia la consola.  
%-----  
A = [7 6 4  
     1 8 2]        %Crea una matriz A de 2 filas y 3 columnas.  
[m,n] = size(A)    %m: número de filas de A.  n: número de columnas de A.  
B = reshape(A,3,2) %B es una matriz de 3 filas y 2 columnas.  
C = reshape(A,6,1) %C es una matriz de 6 filas y 1 columna.  
D = reshape(A,1,6) %D es una matriz de 1 fila y 6 columnas.  
%-----
```

```
%Resultados en consola:  
A = 7     6     4  
    1     8     2  
m = 2  
n = 3  
B = 7     8  
    1     4  
    6     2  
C = 7  
    1  
    6  
    8  
    4  
    2  
D = 7     1     6     8     4     2
```

Figura 18: Ejemplos que muestran el uso del comando *reshape*. Note que el número de elementos de los arreglos antes y después no puede cambiar.

Uso del comando *plot*

```
%-----  
close all          %Cierra todas las ventanas.  
clear all          %Borra todas las variables del workspace.  
clc                %Limpia la consola.  
%-----  
  
x=0:0.5:15;        %Vector x.  
y1=2*x;            %Vector y1.  
y2=0.2*(x.^2);     %Vector y2.  
h1=figure          %Crea el manejador de la figura  
set(gcf,'color','white') %Define que el fondo sea blanco.  
plot(x,y1,'*','color','red'); %Grafique y1 vs x, usando como marcador  
                        %un * rojo.  
  
hold on            %Mantiene el gráfico actual y todas las  
                  %propiedades de la figura actual.  
  
plot(x,y2,'color','blue','LineWidth',2); %Grafique y2 vs x, usando una línea  
                                         %continua azul de grosor 2.  
  
hold off  
xlabel('Variable independiente x') %Texto en el eje "x".  
ylabel('Variable dependiente y')  %Texto en el eje "y".  
title('Gráfica y vs x')           %Título de la gráfica.  
grid on                           %Activa la grilla.  
%-----
```

Figura 19: Ejemplo para graficar usando el comando *plot*.

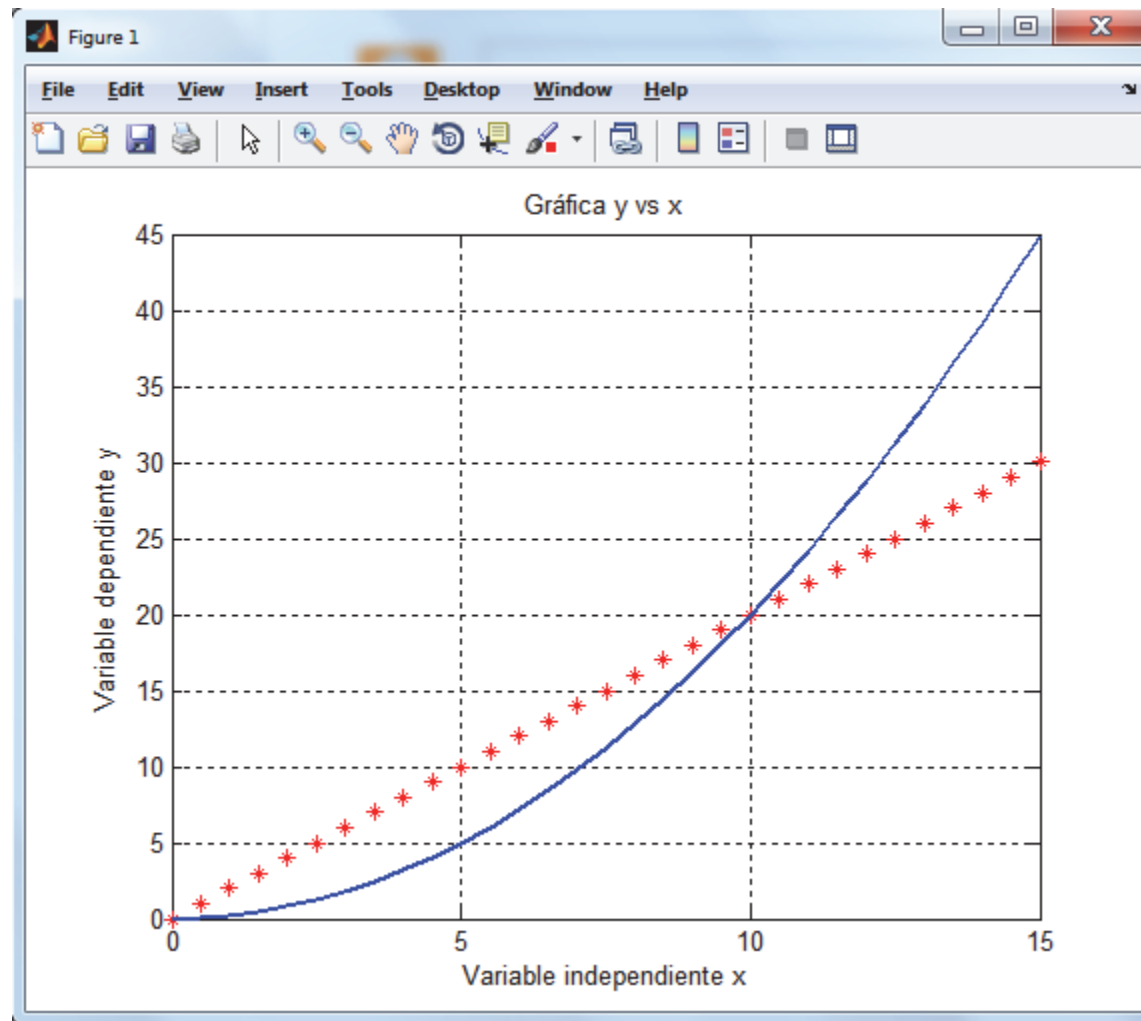


Figura 20: Gráfica resultante en Matlab.

Uso del comando *subplot*

```
close all          %Cierra todas las ventanas.
clear all          %Borra todas las variables del workspace.
clc                %Limpia la consola.
x=0:0.5:15;        %Vector x.
y1=2*x;            %Vector y1.
y2=0.2*(x.^2);     %Vector y2.
h1=figure          %Crea el manejador de la figura
set(gcf,'color','white') %Define que el fondo sea blanco.
%-----
subplot(2,1,1)      %Dividir la ventana en 2 filas y una columna
                    %y activar la primera ventana
plot(x,y1,'*','color','red'); %Grafique y1 vs x, usando como marcador
                               %un * rojo.
xlabel('x')          %Texto en el eje "x".
ylabel('y_1')        %Texto en el eje "y1".
title('Gráfica y_1 vs x','FontSize',14) %Título de la gráfica.
grid on              %Activa la grilla.
%-----
subplot(2,1,2)      %Activar la segunda ventana.
plot(x,y2,'color','blue','LineWidth',2); %Grafique y2 vs x, usando una línea
                                         %continua azul de grosor 2.
xlabel('x')          %Texto en el eje "x".
ylabel('y_2')        %Texto en el eje "y1".
title('Gráfica y_2 vs x','FontSize',14) %Título de la gráfica.
grid on              %Activa la grilla.
%-----
```

Figura 21: Ejemplo para manejar varias gráficas dentro de una figura usando el comando *subplot*.

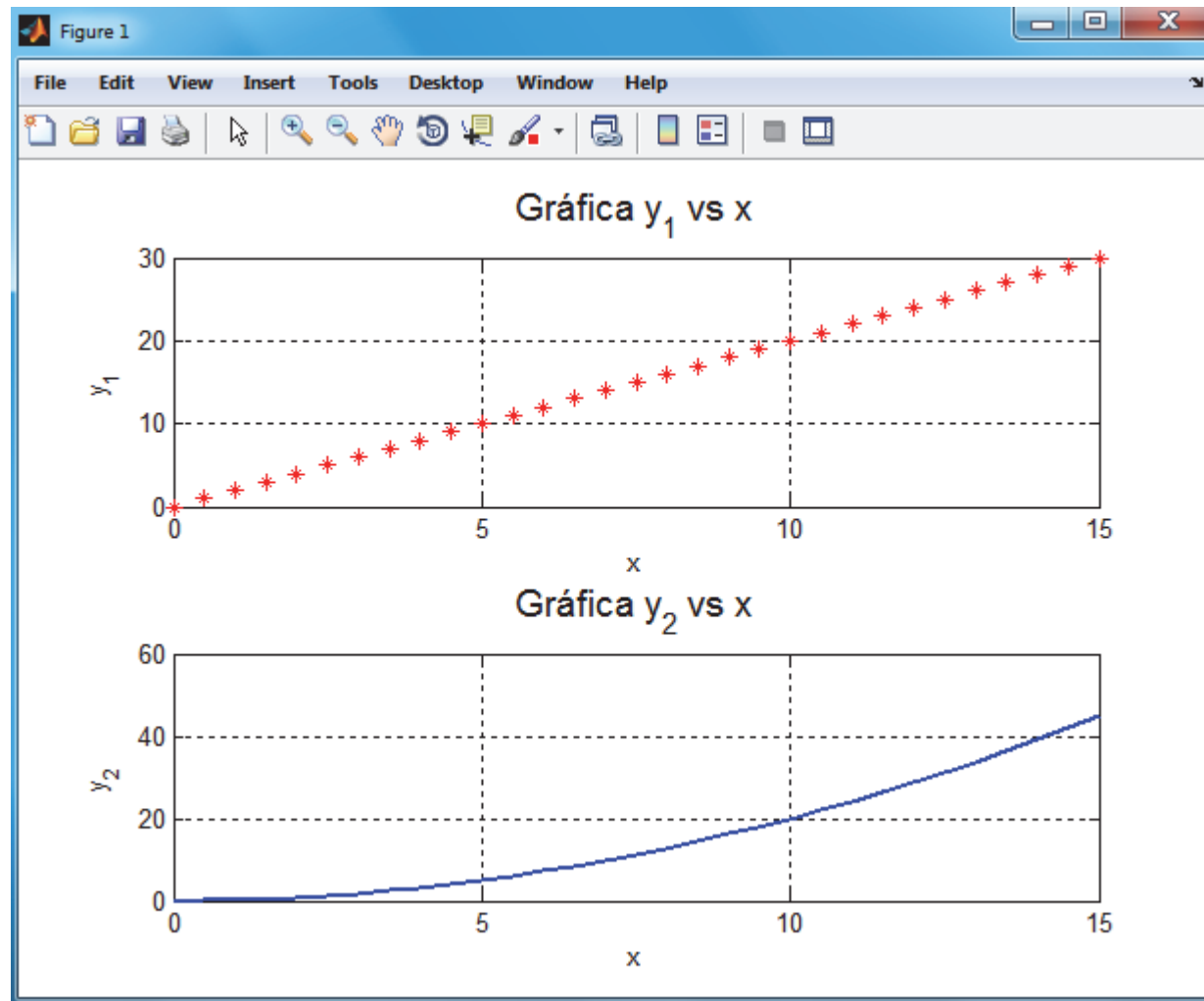


Figura 22: Gráfica resultante en Matlab.

Ejemplo con operaciones vectoriales

```
%-----  
close all      %Cierra todas las ventanas.  
clear all      %Borra todas las variables del workspace.  
clc            %Limpia la consola.  
%-----  
a=[1; 2; 3]    %a es un vector columna de 3 filas.  
b=[4; 5; 6]    %b es un vector columna de 3 filas.  
c=a.*b         %c va a ser un vector columna de 3 elementos.  
d=(a.')*b       %d va a ser un número escalar.  
e=(a)*(b.')     %e va a ser una matriz de 3 filas y 3 columnas.  
%-----
```

```
a =  
 1  
 2  
 3  
  
b =  
 4  
 5  
 6  
  
c =  
 4  
10  
18  
  
d = 32  
  
e =  
 4  5  6  
 8 10 12  
12 15 18
```

Figura 23: Ejemplos de diferentes operaciones vectoriales.

Ejemplo con operaciones matriciales

```
%-----  
close all           %Cierra todas las ventanas.  
clear all           %Borra todas las variables del workspace.  
clc                 %Limpia la consola.  
%-----  
A=[4 5 8;  
   2 1 9;  
   7 6 0]  
  
B=[1 0 0;  
   0 1 0;  
   0 0 1]  
  
C=[2 7 9;  
   1 4 8;  
   3 5 6]  
  
D=A*B  
  
E=A.*B  
  
F=A./C  
%-----
```

A =	4	5	8
	2	1	9
	7	6	0
B =	1	0	0
	0	1	0
	0	0	1
C =	2	7	9
	1	4	8
	3	5	6
D =	4	5	8
	2	1	9
	7	6	0
E =	4	0	0
	0	1	0
	0	0	0
F =	2.0000	0.7143	0.8889
	2.0000	0.2500	1.1250
	2.3333	1.2000	0

Figura 24: Ejemplos de diferentes operaciones matriciales.

Concatenación vertical y horizontal de vectores

```
close all
clear all
clc

%-----
a=[9 8 7 6]
b=[5 4 3 2]

c=[a,b]      %Concatenación horizontal de dos vectores fila.

g=[a;b]      %Concatenación vertical de dos vectores fila.

%-----
u=[1;
   2]
v=[3;
   4]

w=[u, v]      %Concatenación horizontal de dos vectores columna.
h=[u;v]      %Concatenación vertical de dos vectores columna.

%-----
```

Figura 25: Ejemplo de concatenación de vectores.

a =	9	8	7	6				
b =	5	4	3	2				
c =	9	8	7	6	5	4	3	2
g =	9	8	7	6				
	5	4	3	2				
u =	1							
	2							
v =	3							
	4							
w =	1	3						
	2	4						
h =	1							
	2							
	3							
	4							

Figura 26: Resultados del ejemplo de concatenación de vectores.

Concatenación vertical y horizontal de matrices

```
close all
clear all
clc
%-----
a=[1 1
   1 1
   1 1]

b=[2 2
   2 2
   2 2]

c=[a,b]    %Concatenación horizontal de dos matrices.

d=[a;b]    %Concatenación vertical de dos matrices.
%-----
```

Figura 27: Ejemplo de concatenación de matrices.

$$\begin{aligned}
 \mathbf{a} &= \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \\
 \mathbf{b} &= \begin{bmatrix} 2 & 2 \\ 2 & 2 \\ 2 & 2 \end{bmatrix} \\
 \mathbf{c} &= \begin{bmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \end{bmatrix} \\
 \mathbf{d} &= \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \\ 2 & 2 \\ 2 & 2 \end{bmatrix}
 \end{aligned}$$

Figura 28: Resultados del ejemplo de concatenación de matrices.

Cómo resolver sistemas de ecuaciones en Matlab?

```
close all
clear all
clc
%-----
%Cómo resolver el siguiente sistema de ecuaciones en Matlab?
%  x1  +2*x2=20
%  4*x1-x2  =35

%Primer paso: Una opción consiste en representar
%              el sistema de la forma M*X=B

%Segundo paso: Construir la matriz del sistema
M=[1  2;
   4 -1]

%Tercer paso: Construir el vector de valores independientes.
B=[20;
   35]

%Cuarto paso: Obtener el vector columna X=[x1; x2]
X=inv(M)*B

%Quinto paso: En caso de ser necesario recuperar las variables x1 y x2
x1=X(1,1)
x2=X(2,1)
%-----
```

$$\begin{array}{rcl} M & = & \begin{bmatrix} 1 & 2 \\ 4 & -1 \end{bmatrix} \\ B & = & \begin{bmatrix} 20 \\ 35 \end{bmatrix} \\ X & = & \begin{bmatrix} 10 \\ 5 \end{bmatrix} \\ x1 & = & 10 \\ x2 & = & 5 \end{array}$$

Figura 29: Ejemplo de solución de un sistema de ecuaciones lineales.

Ejemplo de uso del comando *save*

```
close all
clear all
clc

a=14.2;
b=[4 5; 6 7];

%Para guardar TODAS las variables disponibles en el workspace,
%en un archivo con nombre todas_las_variables.mat
%en la ruta C:\MiCarpetaDeEjemplo\
%debe escribir:
save('C:\MiCarpetaDeEjemplo\todas_las_variables.mat');

c=[1 2 ; 3 4];
d=20.7;

%Para guardar ÚNICAMENTE las variables "c" y "d"
%en un archivo con nombre algunas_variables.mat
%en la ruta C:\MiCarpetaDeEjemplo\
%debe escribir:
save('C:\MiCarpetaDeEjemplo\algunas_variables.mat','c','d');
```

Figura 30: Grabando variables en archivos .mat.

Ejemplo de uso del comando *load*

```
close all
clear all
clc

%Para cargar TODAS las variables disponibles del archivo
%todas_las_variables.mat que se encuentra en la ruta C:\MiCarpetaDeEjemplo\
%se debe escribir:
load('C:\MiCarpetaDeEjemplo\todas_las_variables.mat');

%Para cargar SOLO la variables "d" disponible en el archivo
%algunas_variables.mat que se encuentra en la ruta C:\MiCarpetaDeEjemplo\
%se debe escribir:
load('C:\MiCarpetaDeEjemplo\algunas_variables.mat','d');
```

Figura 31: Cargando variables en el workspace a partir de archivos .mat.

Octave

- Es un programa que comparte muchos de los atributos de Matlab, incluyendo la mayor parte de su sintaxis, con la ventaja que es gratuito.
- Octave es una excelente alternativa para aquellos usuarios que no poseen una licencia de Matlab en sus casas o lugares de trabajo.
- Octave es multiplataforma. Existen versiones para Windows, Mac, Linux e incluso Android.
- Se puede descargar de <http://www.gnu.org/software/octave/>
- Las nuevas versiones de Octave (≥ 3.8) para Windows incorporan además de la tradicional interfaz por consola, una interfaz gráfica experimental y un editor de texto. <https://www.gnu.org/software/octave/>
- En Windows se recomienda descargar Octave desde <http://mxeoctave.osuv.de/>

Línea de comandos en Octave usando la interfaz tradicional



```
C:\Octave\Octave-3.8.2\bin\octave-cli.exe
GNU Octave, version 3.8.2
Copyright (C) 2014 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.

Octave was configured for "i686-w64-mingw32".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-involved.html
Read http://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.
octave:1>
```

Figura 32: Pantalla inicial de Octave usando la interfaz por consola.

Interfaz de usuario experimental de Octave

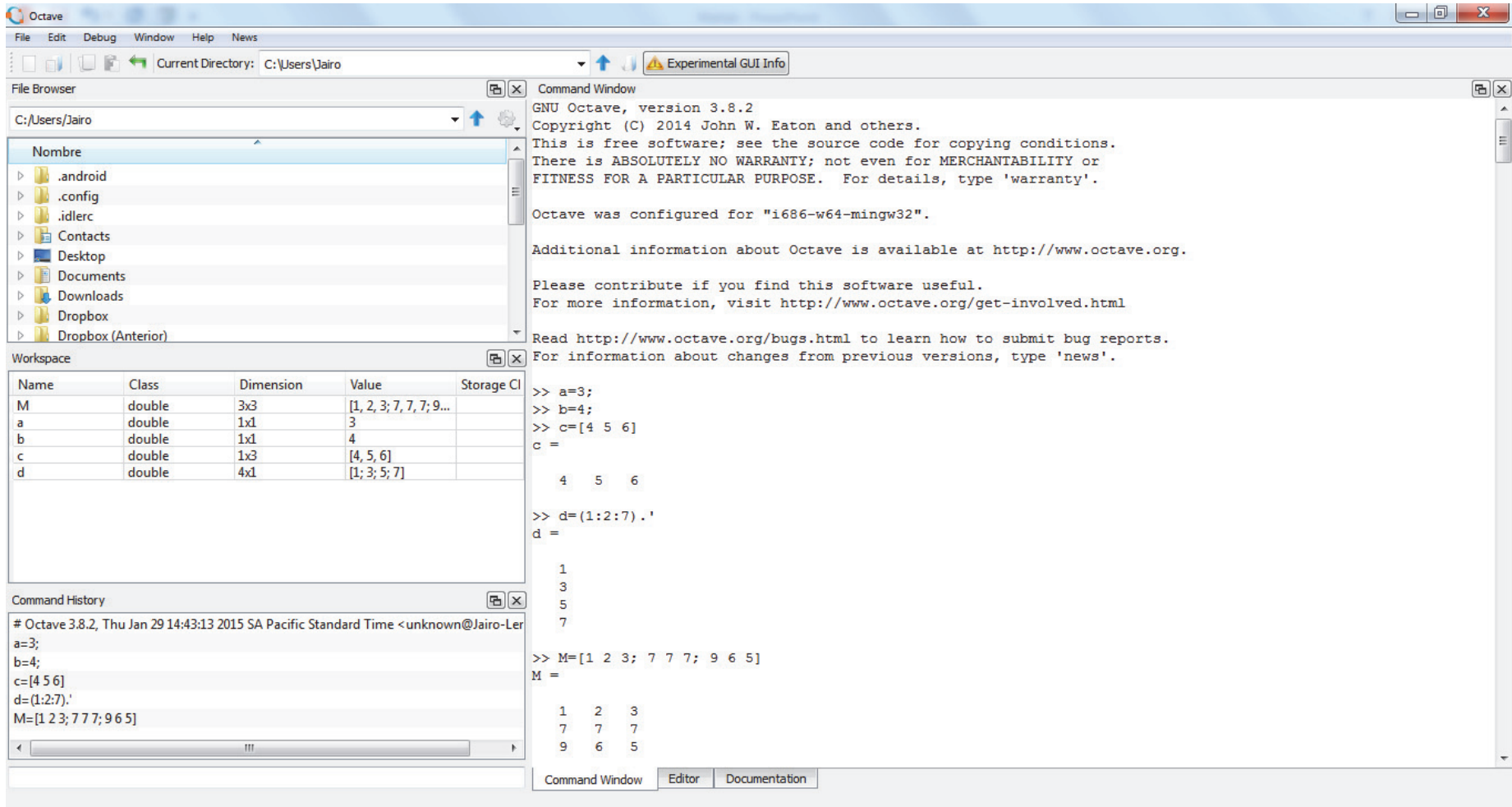


Figura 33: Pantalla inicial de Octave usando la interfaz gráfica.

Observación

- Si prefiere usar la interfaz por consola, puede emplear como editor de texto un programa como sublime (<http://www.sublimetext.com/>) y configurarlo para que resalte la sintaxis. Para ello, en sublime presione “*ctrl shift p*”, luego escriba “*ssMATLAB*” y presione enter.
- En la consola de Octave usted puede usar los comandos que utiliza en Linux para navegar por los directorios:
`cd` `cd ..` `ls` `pwd`
- Una vez el path de Octave apunte al directorio donde están sus archivos, puede ejecutar un script, escribiendo el nombre correspondiente sin la extensión y presionando enter.

Ejercicios propuestos

1. Crear un vector columna con 20 elementos. Los valores del vector deben ir creciendo de 1 en 1 hasta 20.
2. Crear una matriz de 5 filas, 4 columnas cuyos elementos sean todos 7.
3. Cómo podría implementar la siguiente matriz en Matlab usando el menor número de líneas de código posible?

```
[100 90 80 .... 0  
 0  1  2 .... 10]
```

4. Concatene horizontalmente dos vectores fila de 4 elementos cada uno. Usted puede escoger los valores que tienen los vectores.
5. Si A es una matriz de 10 x 10 elementos, cómo puede recuperar la matriz superior izquierda de 5 x 5 y reemplazarla por una matriz diagonal. Tip: consulte la función `eye` de Matlab.
6. Es posible realizar las siguientes operaciones? (indique falso o verdadero):
 - `a.*b` si a es un vector fila y b es un vector columna, ambos con el mismo número de elementos?
 - `(a.').*b` si a es un vector fila y b es un vector columna, ambos con el mismo número de elementos?
 - Concatenar horizontalmente dos matrices que tengan el mismo número de filas pero un número diferente de columnas?
7. Como podría obtener un vector columna de números pares entre 50 y 100 en Matlab?

- 8) Desarrollar una función que tome un vector t , una amplitud A , la frecuencia angular w , y retorne la función $A \cdot \sin(w \cdot t)$. La función debe adicionalmente graficar “ y ” vs “ t ”, indicando el título del gráfico y los nombres de los ejes “ x ” e “ y ”. Adicional: El gráfico debe mostrar la cuadrícula y debe tener fondo blanco.
- 9) Genere una matriz aleatoria con números ENTEROS de dimensiones 5×7 .
- 10) Genere una matriz de 10×2 que tenga todos sus elementos en ceros.
- 11) Genere una matriz de 3×4 que tenga todos sus elementos en 3.5.
- 12) Convierta una matriz de 3×2 en un vector de 6×1 .
- 13) Desarrolle una función que calcule el promedio aritmético de los datos de un vector a .
- 14) Desarrolle una función que calcule la hipotenusa h de un triángulo rectángulo con catetos a , b .
- 15) Como elevaría al cuadrado cada uno de los elementos de un vector b ?
- 16) Utilizando los comandos *imread* e *imshow*, cargue y muestre una imagen en Matlab de su preferencia.
- 17) Investigue sobre el comando *find*.

Preguntas??