

---

## Ejercicio Práctico: Identificación y manejo de Objetos Gráficos

### Asignatura: Procesamiento de Señales

Universidad del Rosario - Escuela de Ingeniería, Ciencia y Tecnología

### Actividad de Aprendizaje

Objetivo:

- Manejo del ambiente gráfico.
- Identificadores de objetos gráficos.

Procedimiento:

1. *Visualización del identificador de un objeto.*

¿Es posible visualizar un identificador?

En Matlab cada identificador debe estar asociado a una variable, luego esa variable es numérica. Con esa variable se pueden cambiar las propiedades del objeto en consideración.

Vamos a llamar al identificador el *handle* o puntero.

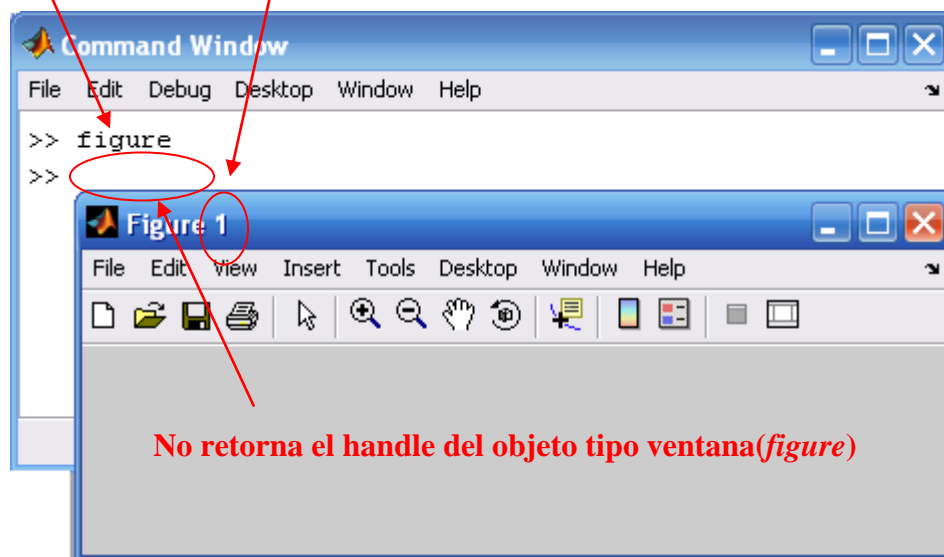
Cada vez que usted crea un objeto se genera un único *handle*(puntero).

El *handle* del objeto raíz(*root*) tiene siempre asignado el valor de 0.

El comando "*figure.m*" genera una nueva ventana y puede retornar el *handle* de este objeto tipo ventana(*figure*) cuando se llama con una variable de salida.

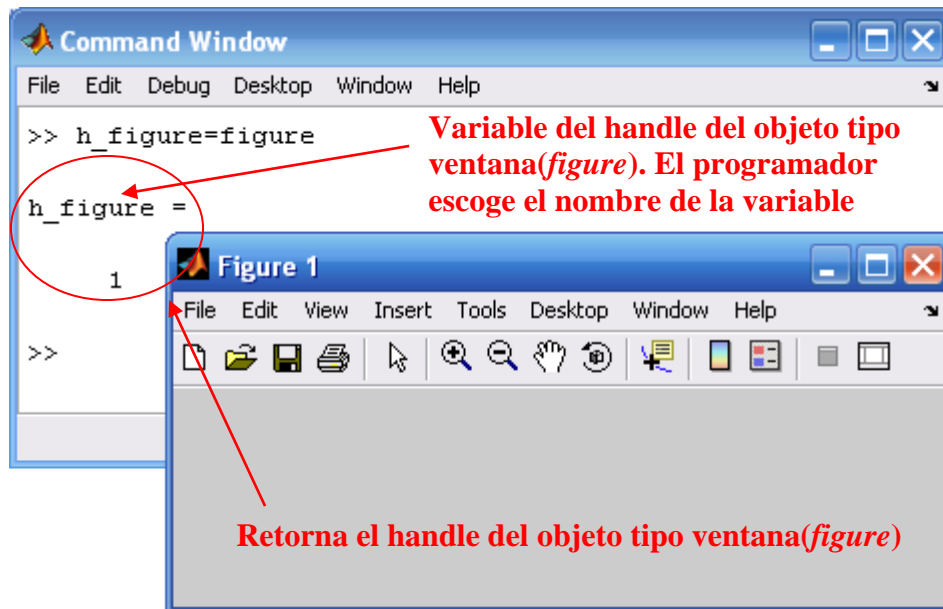
**Comando**

**Número de la ventana**



**No retorna el handle del objeto tipo ventana(*figure*)**

Ahora, es posible utilizar el comando “*figure.m*” y conocer el valor numérico del puntero.



Generalmente, se asignan enteros a los *handle* de los objeto tipo ventana(*figure*); para otros objetos, el valor numérico es un número real.

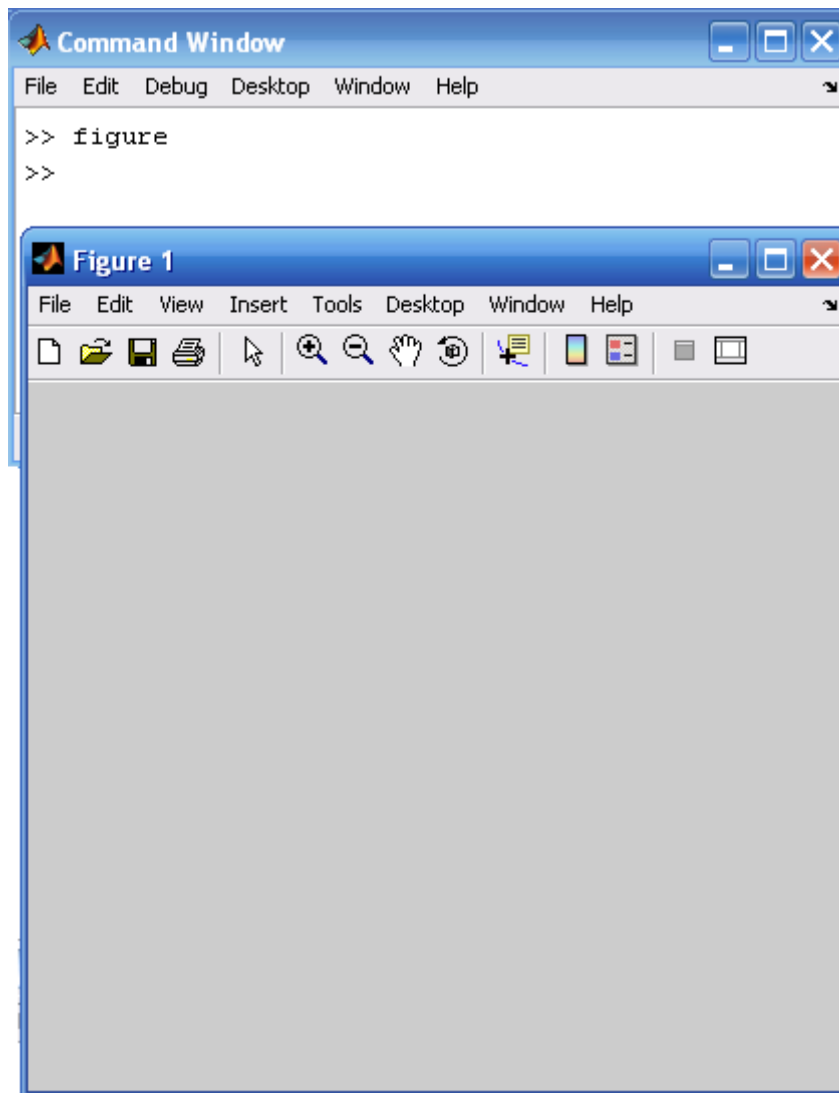
Existen ciertas funciones de Matlab que permiten crear objetos y además retornan los handle de los mismos. Estas funciones se muestran a continuación,

Objeto	Descripción
figure	Una ventana en la cual se dibujan los gráficos.
axes	Sistema rectangular coordenado para los dibujos en una figura.
line	Línea que conecta un conjunto de datos sobre unos ejes coordenados.
text	Cadena de carácter situado respecto a los ejes coordenados.
patch	Área poligonal definida por líneas conectadas.
surface	Superficie definida por caras conectadas similares a los <i>patch</i> .
light	Fuente de luz direccional en los ejes coordenados que afecta a las superficies.
image	Foto en dos dimensiones definida por colores de píxel.
uimenu	Menú programable para una ventana( <i>figure</i> ).
uicontrol	Dispositivo de interfaz de usuario programable tal como un <i>checkbox</i> , <i>slider</i> o <i>pushbutton</i> .

Es posible buscar el handle de una ventana(*figure*) que se encuentra presente en el objeto raiz (pantalla del computador), el comando para buscar el handle es “*gcf.m* (*get current figure* – obtener ventana actual)”.

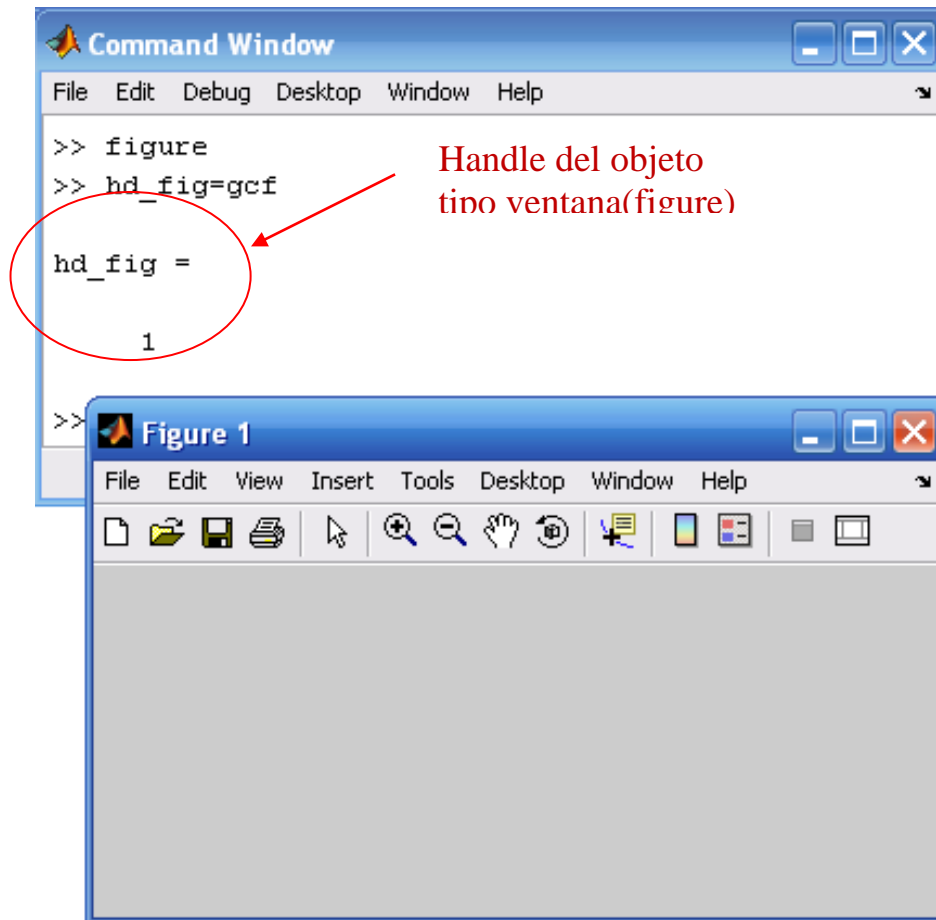
---

Supongamos que usted genera una ventana llamando simplemente el comando “figure.m”,



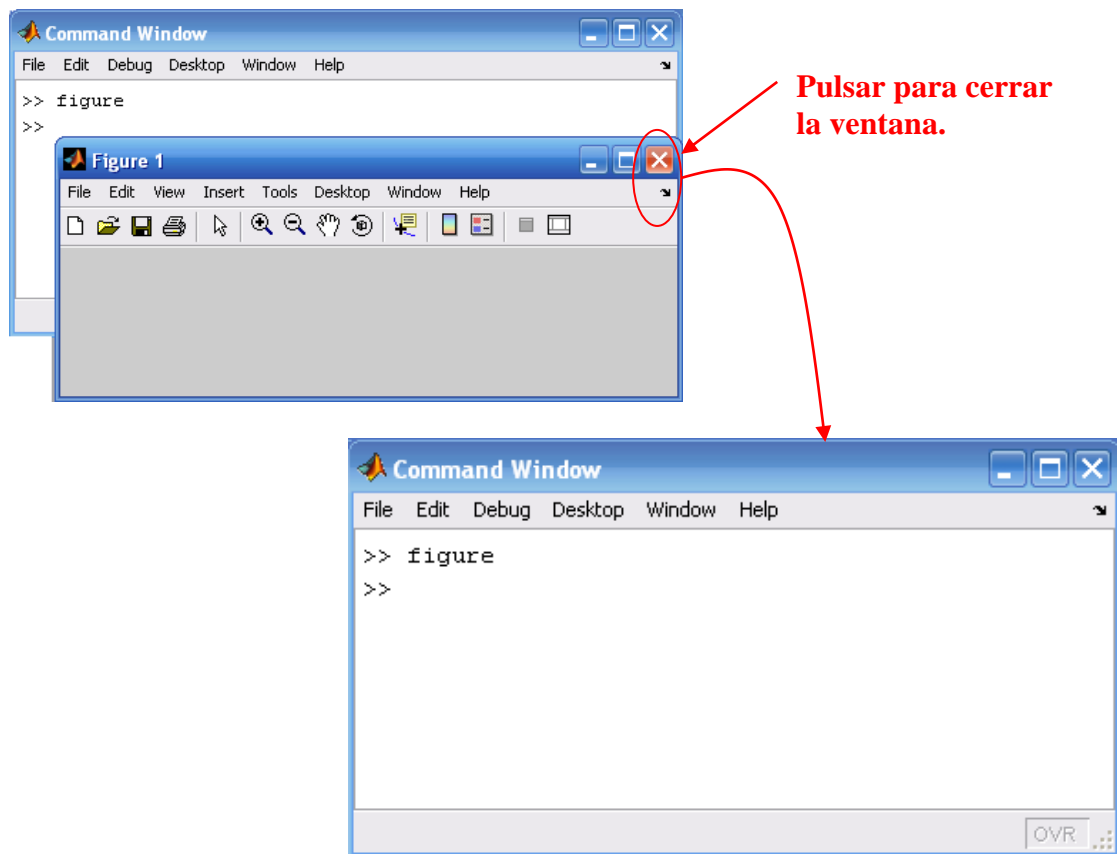
Como se observa, no se tiene el *handle* del objeto tipo ventana(*figure*) creado.

Ahora, se obtiene el *handle* de ese objeto utilizando el comando “gcf.m”,

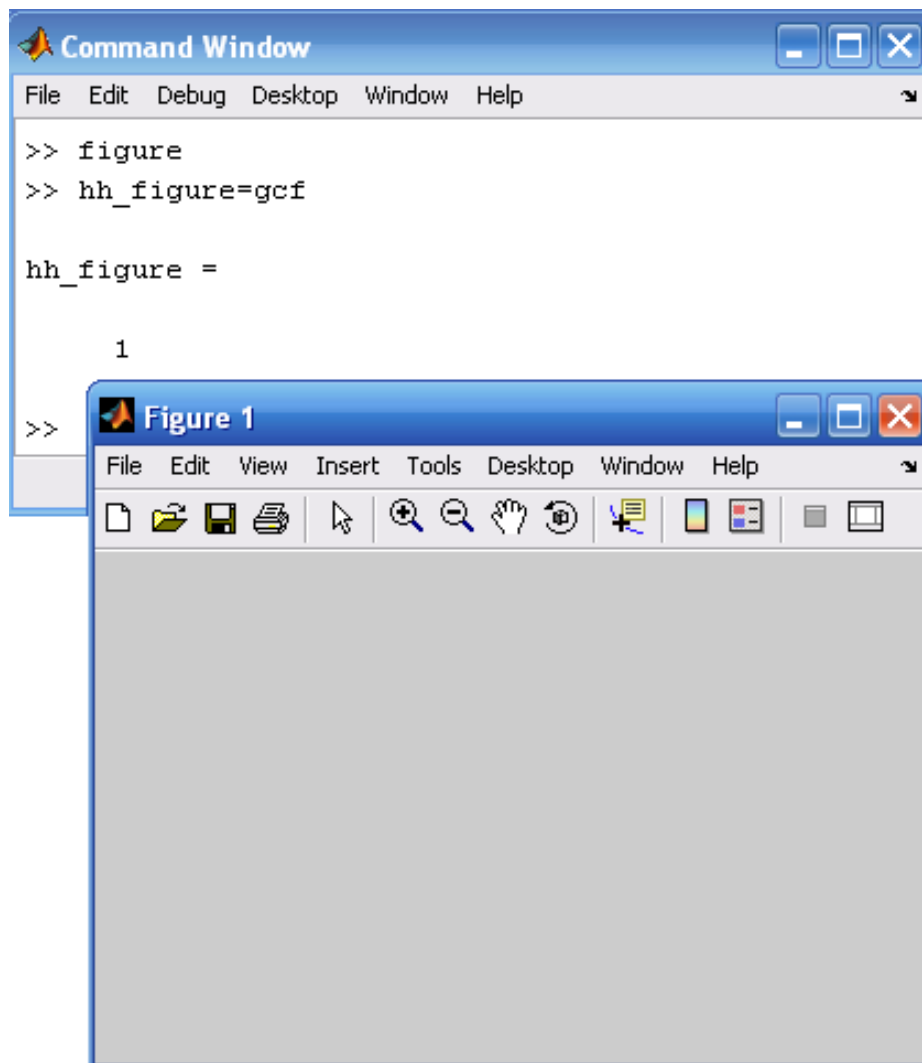


Hay que tener presente que la ventana llamada "Figure 1" debe estar presente, es decir, el objeto ventana debe estar en el objeto raíz.

Supongamos que se genera un objeto tipo ventana (*figure*) y se cierra la figura, quedando solo la ventana de comandos (*command window*),

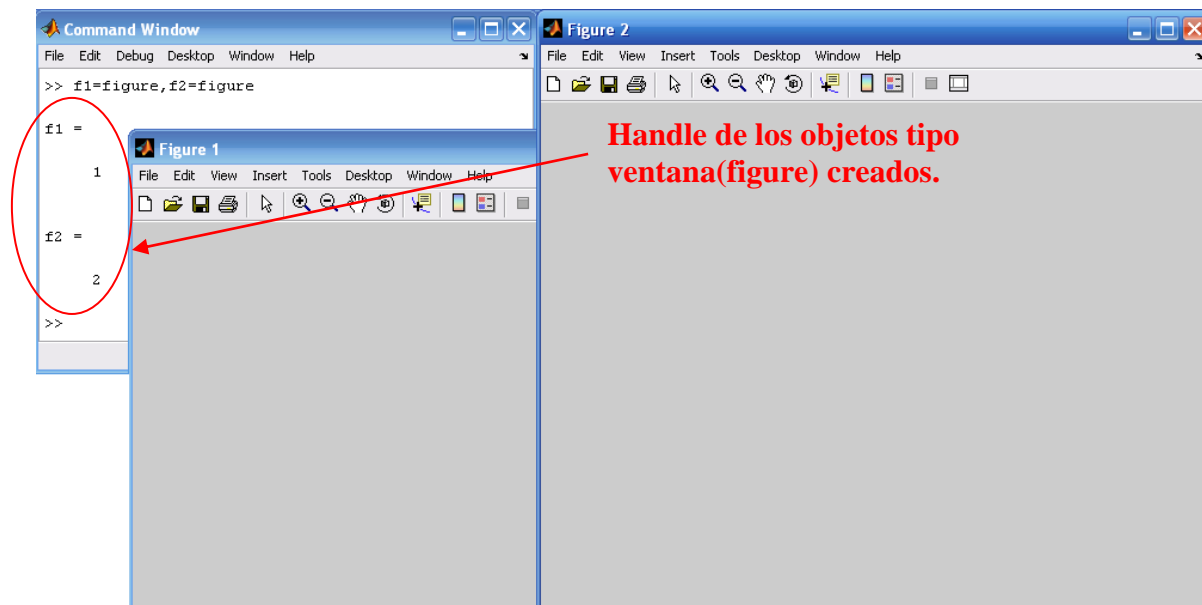


Ahora, se busca el *handle* del objeto tipo ventana(*figure*) utilizando el comando “*gcf.m*”,

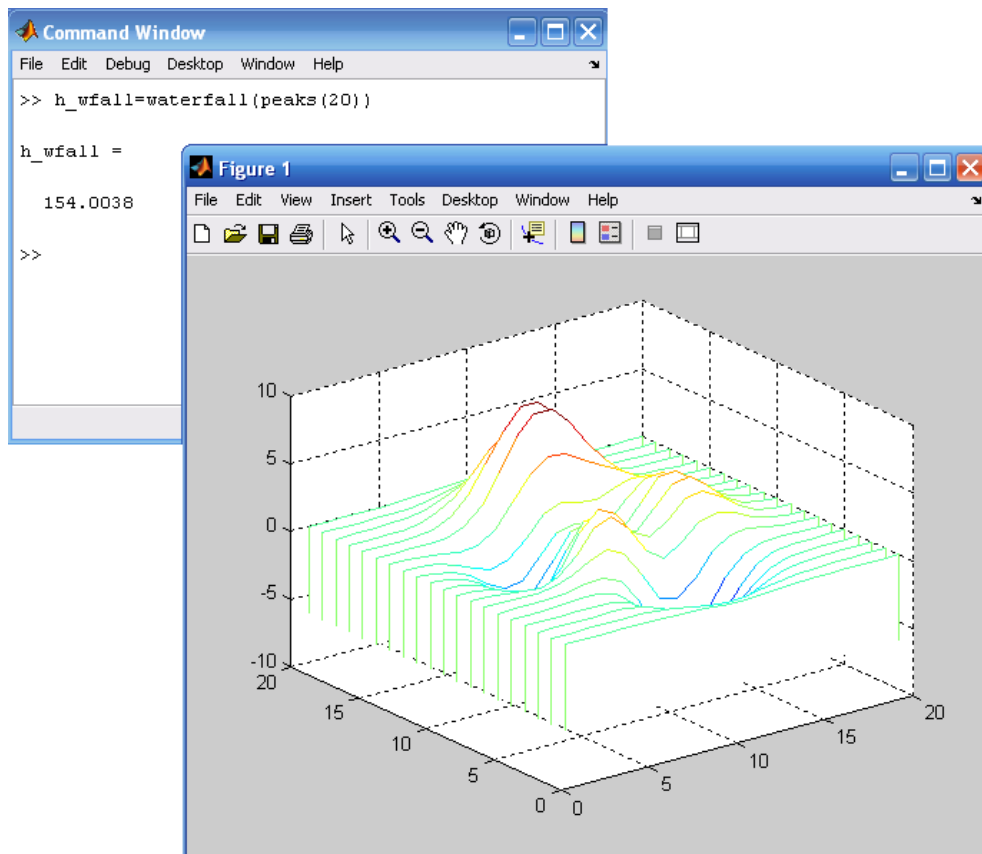


Este comando, genera una nueva ventana(*figure*) y retorna el *handle*.

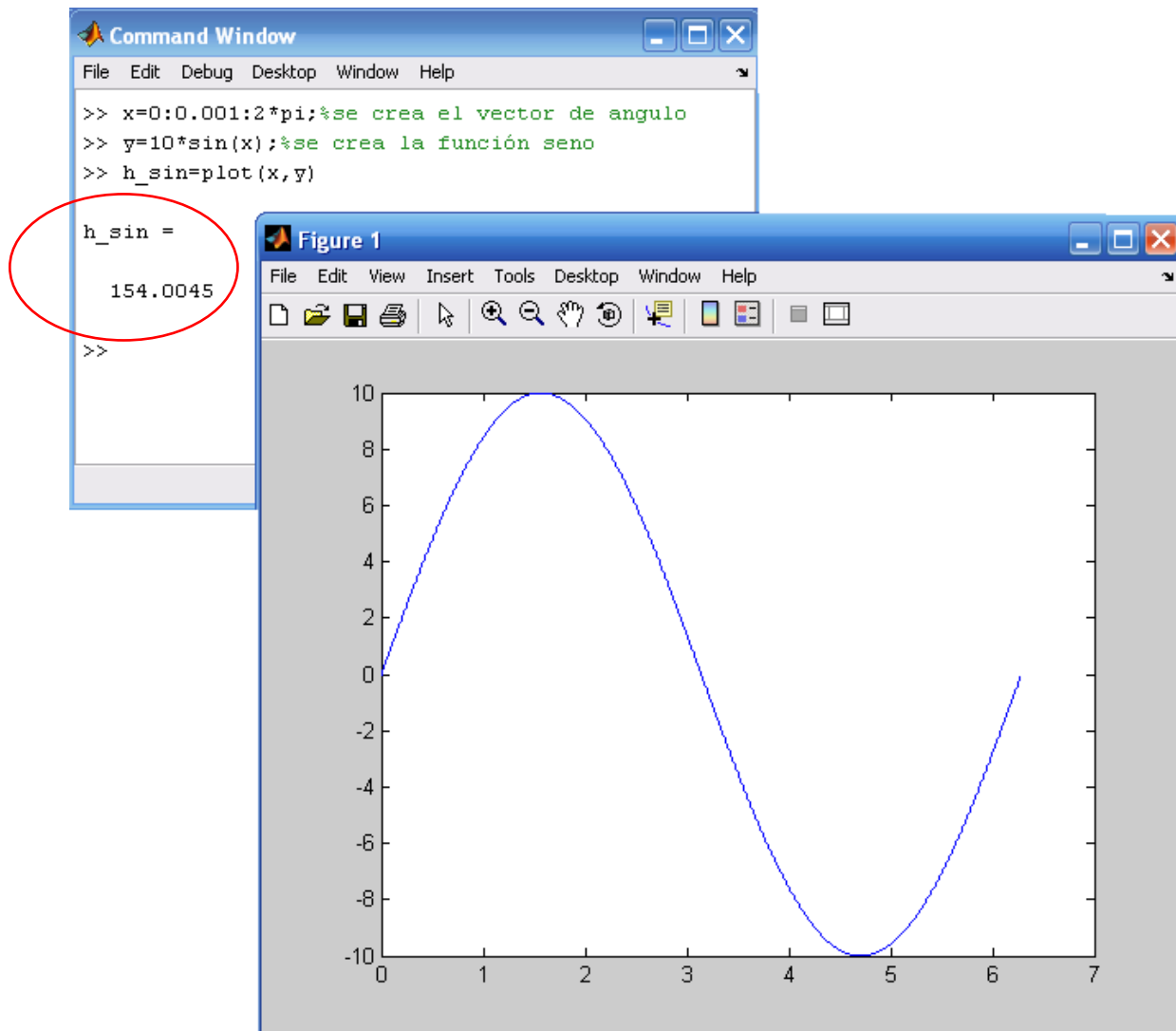
Para generar dos ventanas(*figures*), se utiliza el comando "*figure.m*" para cada ventana creada. Para ello, realice lo siguiente,



Todos los comandos de Matlab que crean objetos, retornan el *handle* o un vector columna de *handles* que corresponde a cada uno de los objetos creados,

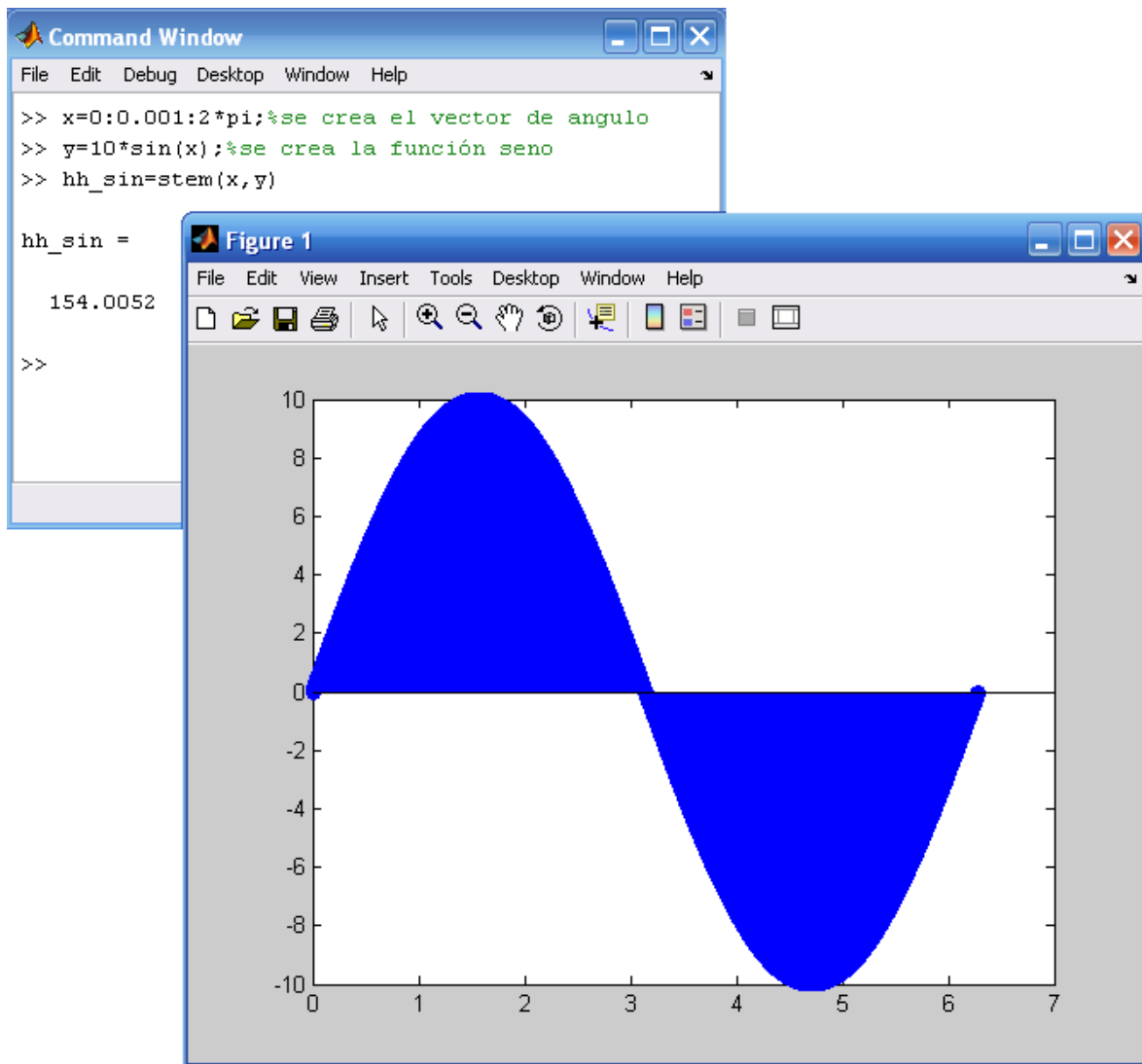


Otro ejemplo de objeto gráfico que genera un *handle* es,



Otro ejemplo se muestra a continuación,





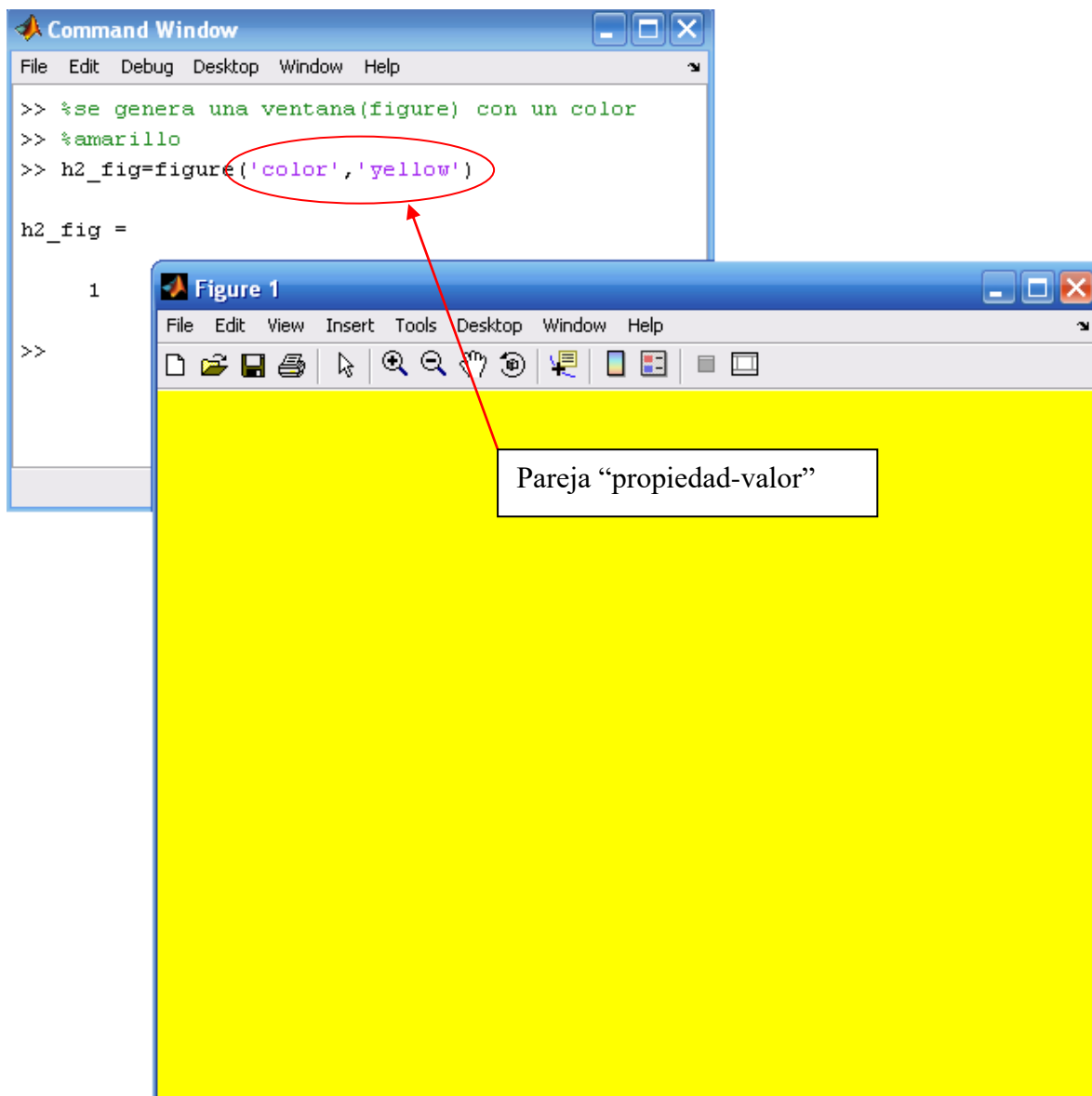
---

## 2. *Propiedades de los objetos.*

Las propiedades de un objeto definen sus características, es decir, modificándolas se puede cambiar la forma como se despliegan las gráficas construidas.

Cada uno de los objetos posee propiedades muy particulares, aunque algunas de ellas son comunes.

Las funciones que generan objetos pueden ser utilizadas con pares de valores llamados *propiedad-valor*, por ejemplo,



Este comando crea una nueva ventana(*figure*) con las propiedades por defecto excepto el color de fondo, el cual es ajustado a amarillo.

---

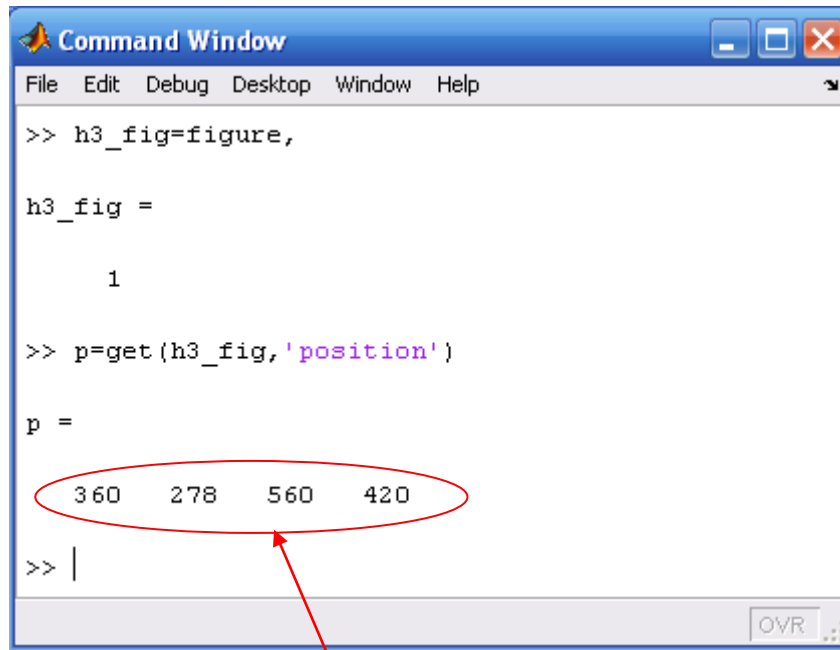
*¿Cómo se puede cambiar u obtener las propiedades del handle graphics de un objeto?*

Para ello existen dos funciones: “*get.m*” y “*set.m*”

**get.m** : retorna el valor actual de una o mas propiedades de un objeto. La sintaxis es

“ `get(handle,'propertyname')` ”

por ejemplo,



```
Command Window
File Edit Debug Desktop Window Help
>> h3_fig=figure,

h3_fig =

     1

>> p=get(h3_fig,'position')

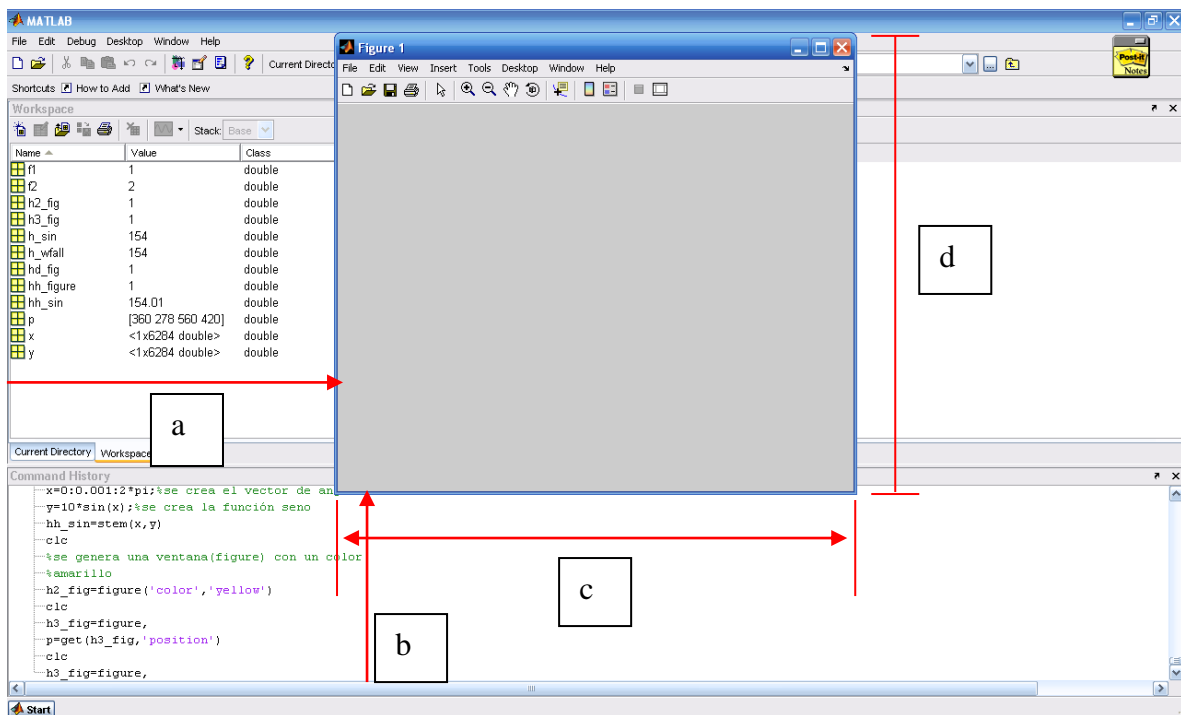
p =

    360    278    560    420

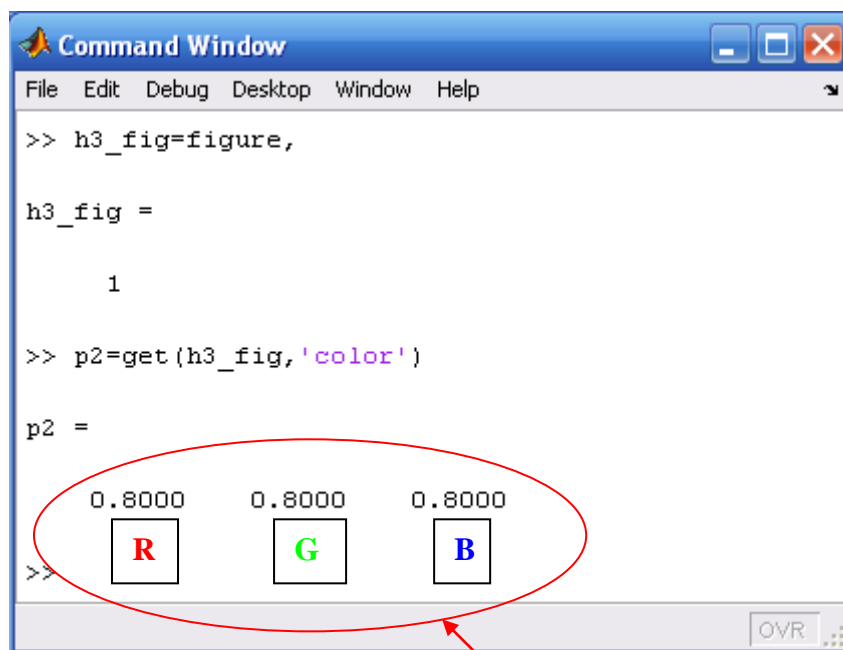
>> |
```

**Retorna la posición en pixeles**

Un vector de posición siempre tiene cuatro posiciones, [a b c d]



Se puede traer la propiedad color del objeto tipo ventana(*figure*),

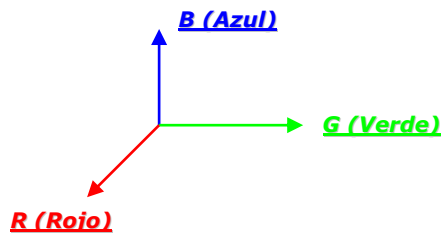


**Componentes [R,G,B]**  
que representan el color  
de fondo la ventana.

El formato de color que utiliza Matlab es con base en el modelo de color RGB que corresponde a las tres coordenadas *Red*, *Green* y *Blue*(Rojo, Verde y Azul) para formar un cubo llamado *CUBO RGB* y con el cual se pueden obtener los demás colores(ver siguiente tabla). Este concepto también es utilizado por la herramienta Microsoft Paint de Windows.

Color	Código RGB Vector tres posiciones	Símbolo
Azul(Blue)	[0 0 1]	b
Verde(Green)	[0 1 0]	g
Rojo(Red)	[1 0 0]	r
Amarillo(Yellow)	[1 1 0]	y
Magenta(Magenta)	[1 0 1]	m
Cyan(Cyan)	[0 1 1]	c
Blanco(White)	[1 1 1]	w
Negro(Black)	[0 0 0]	k

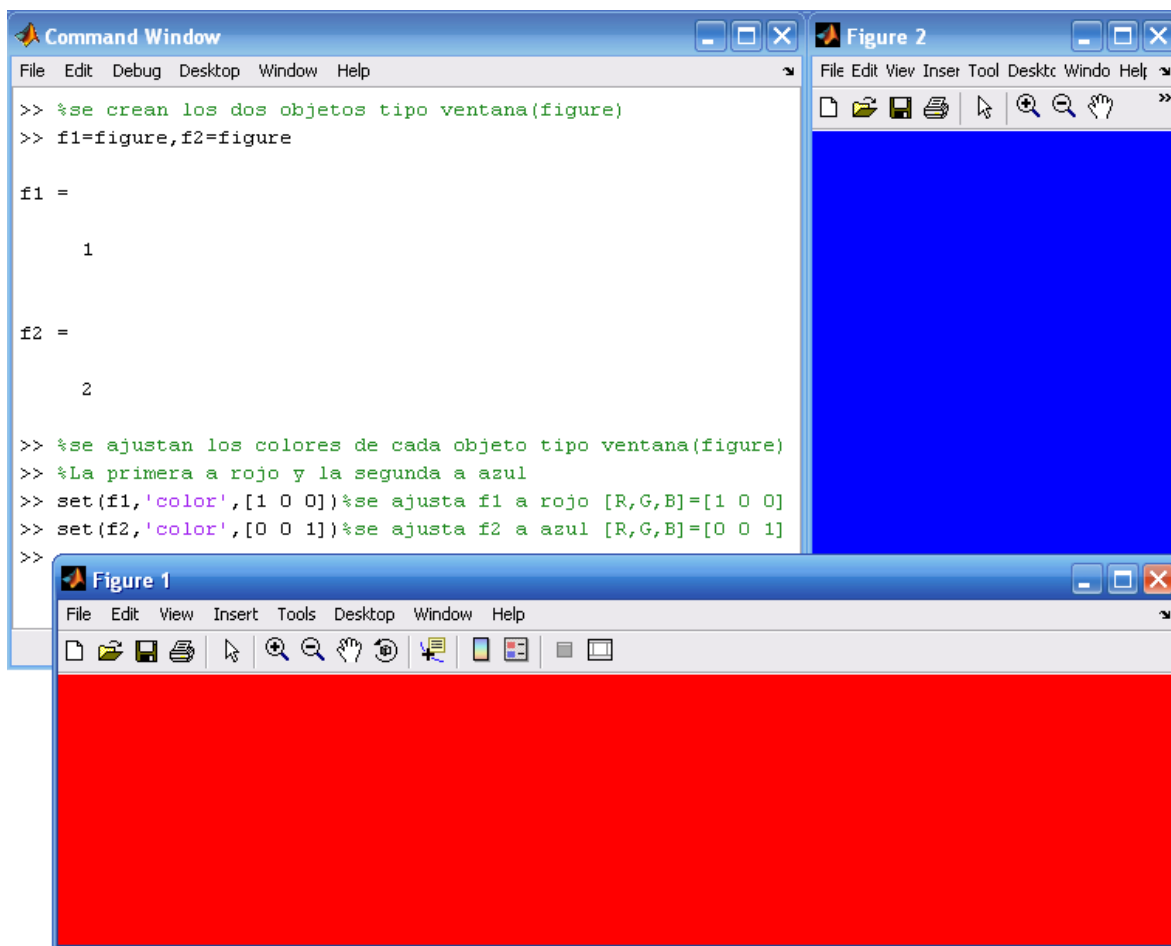
Todo color siempre se especifica como un vector de tres posiciones correspondientes a las tres coordenadas R, G y B(Espacio de color RGB).



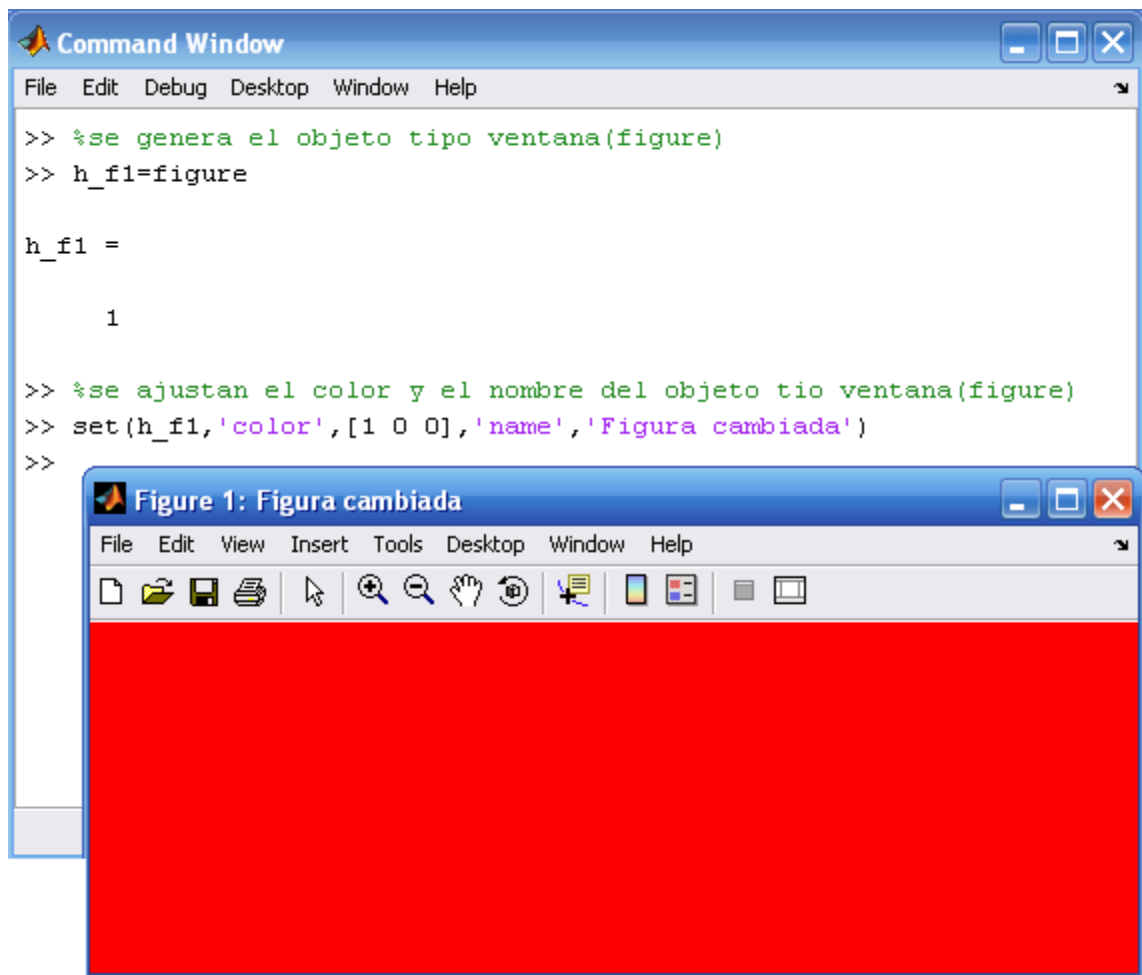
**set.m:** cambia propiedades de los valores de propiedades de objetos gráficos identificados con su *handle*. La sintaxis es

“ `set(handle,'propertyname','value')` ”

Supongamos que se generan dos objetos tipo ventana(*figure*) y se desea que la primera ventana(*figure*) posea un color rojo y la segunda ventana(*figure*) posea un color azul, para ello se hace,

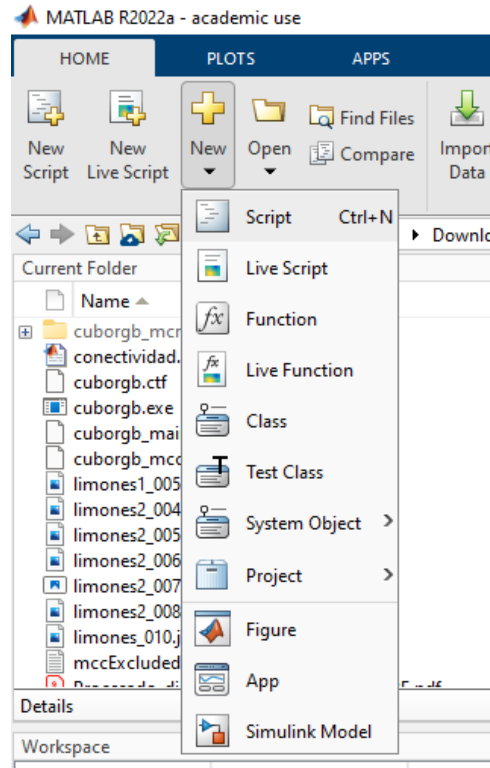


En general, el comando “*set.m*” puede poseer cualquier número de pares “propiedad-valor(property-value)”, por ejemplo, supongamos que se tiene un objeto tipo ventana(*figure*) y se desea cambiar su propiedad “color” a rojo y su propiedad “name” a “*Figura cambiada*”.



Ahora se utiliza el editor de Matlab para observar algunos efectos cuando se realiza el cambio, para esto, realice los siguientes pasos:

- Abrir el editor de Matlab,



- Ubicar el siguiente código en el editor,

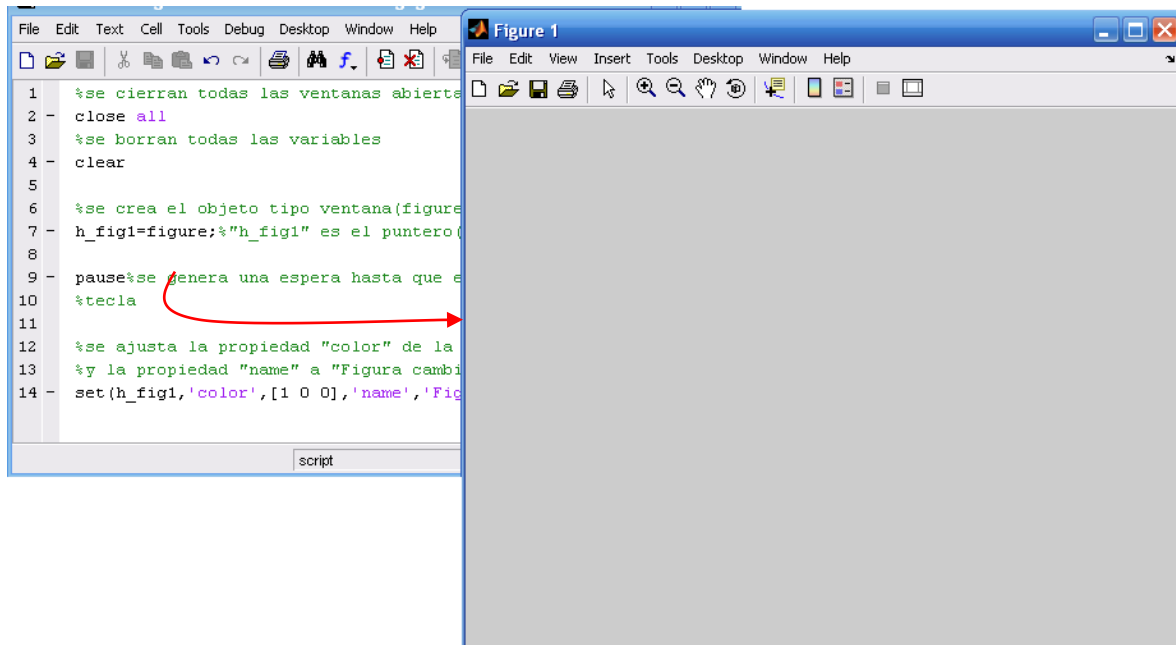
```

1  %se cierran todas las ventanas abiertas
2  close all
3  %se borran todas las variables
4  clear
5
6  %se crea el objeto tipo ventana(figure)
7  h_fig1=figure;% "h_fig1" es el puntero(handle) del objeto creado
8
9  pause%se genera una espera hasta que el usuario pulse cualquier
10 %tecla
11
12 %se ajusta la propiedad "color" de la ventana(figure) a rojo
13 %y la propiedad "name" a "Figura cambiada"
14 set(h_fig1,'color',[1 0 0],'name','Figura Cambiada')

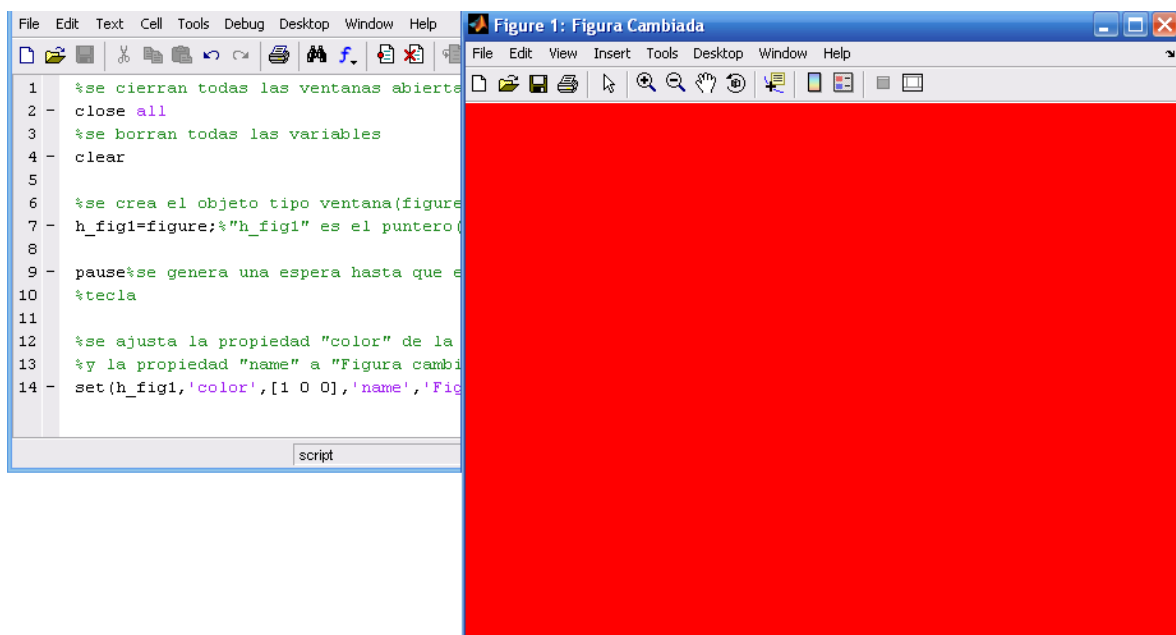
```

- Se guarda y ejecuta el programa creado como “script”.





- Pulse cualquier tecla del teclado,



El efecto es el cambio de las propiedades sin ser necesario generar un nuevo objeto tipo ventana(*figure*).

¿Cómo se pueden conocer todas las propiedades que posee un objeto?  
Utilizando el comando *set.m* se pueden conocer todas las propiedades que se pueden ajustar en el objeto.

```
>> %se genera el objeto tipo ventana.figure)
>> h2_fig=figure

h2_fig =

    1

>> %propiedades que se pueden ajustar en el objeto
>> set(h2_fig)
Alphamap
BackingStore: [ (on) | off ]
CloseRequestFcn: string -or- function handle -or- cell array
Color
Colormap
CurrentAxes
CurrentCharacter
CurrentObject
CurrentPoint
DockControls: [ (on) | off ]
DoubleBuffer: [ (on) | off ]
FileName
IntegerHandle: [ (on) | off ]
InvertHardcopy: [ (on) | off ]
KeyPressFcn: string -or- function handle -or- cell array
MenuBar: [ none | (figure) ]
MinColormap
Name
```

Ahora, si se desea conocer una propiedad en específico,

```
>> %se genera el objeto tipo ventana.figure)
>> h2_fig=figure

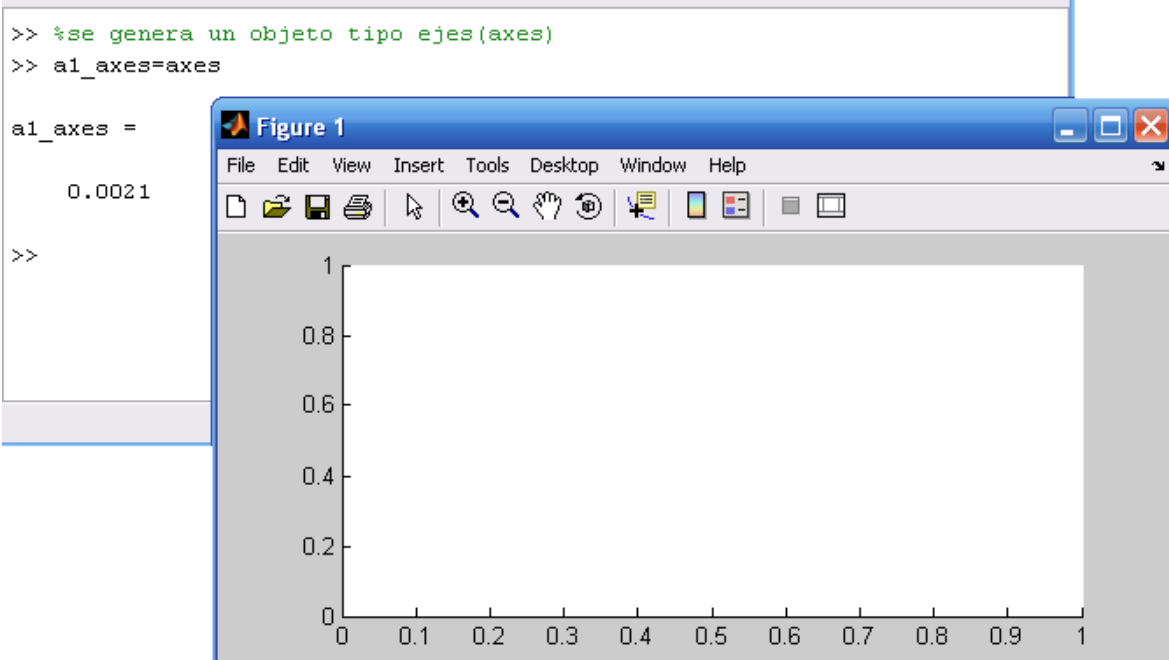
h2_fig =

    1

>> set(h2_fig,'units')
[ inches | centimeters | normalized | points | {pixels} | characters ]
>>
```

Se observa que retorna un conjunto de valores que pueden ser asignados a la propiedad “units” del objeto.

Ahora, se genera un objeto tipo ejes(axes),



Se observa que Matlab genera un objeto tipo ventana(*figure*) y luego ubica sobre éste el objeto tipo ejes(*axes*) deseado, esto hace que la jerarquía se conserve.

Además, se pueden visualizar las propiedades que se pueden ajustar en el objeto tipo ejes(axes),

```
>> %se genera un objeto tipo ejes(axes)
>> al_axes=axes;
>> %se observan las propiedades que pueden ser ajustadas
>> set(al_axes)
    ActivePositionProperty: [ position | {outerposition} ]
    ALim
    ALimMode: [ {auto} | manual ]
    AmbientLightColor
    Box: [ on | {off} ]
    CameraPosition
    CameraPositionMode: [ {auto} | manual ]
    CameraTarget
    CameraTargetMode: [ {auto} | manual ]
    CameraUpVector
    CameraUpVectorMode: [ {auto} | manual ]
    CameraViewAngle
    CameraViewAngleMode: [ {auto} | manual ]
    CLim
    CLimMode: [ {auto} | manual ]
    Color
    ColorOrder
    DataAspectRatio
    DataAspectRatioMode: [ {auto} | manual ]
    DrawMode: [ {normal} | fast ]
    FontAngle: [ {normal} | italic | oblique ]
    FontName
    FontSize
    FontUnits: [ inches | centimeters | normalized | {points} | pixels ]
    FontWeight: [ light | {normal} | demi | bold ]
    GridLineStyle: [ - | -- | {:} | -. | none ]
    Layer: [ top | {bottom} ]
    LineStyleOrder
```

### 3. Ejemplos de empleo del comando "set.m".

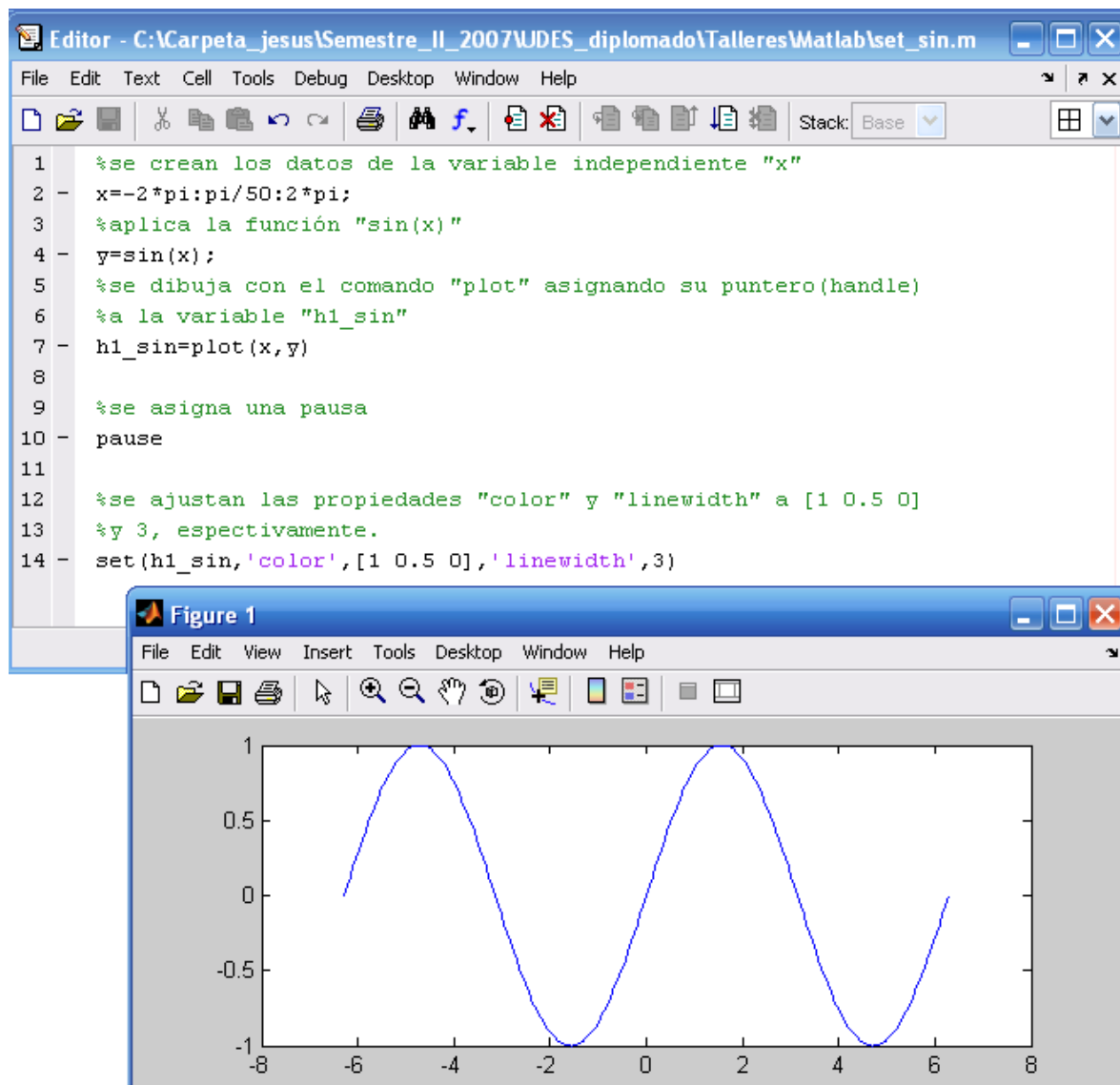
Se desea dibujar una línea en un color no estándar [R,G,B]=[1 0.5 0] (cercano al naranja), es decir, un color diferente a los ocho colores comentados anteriormente.

Escribir las siguientes líneas de código en el Editor de Matlab,

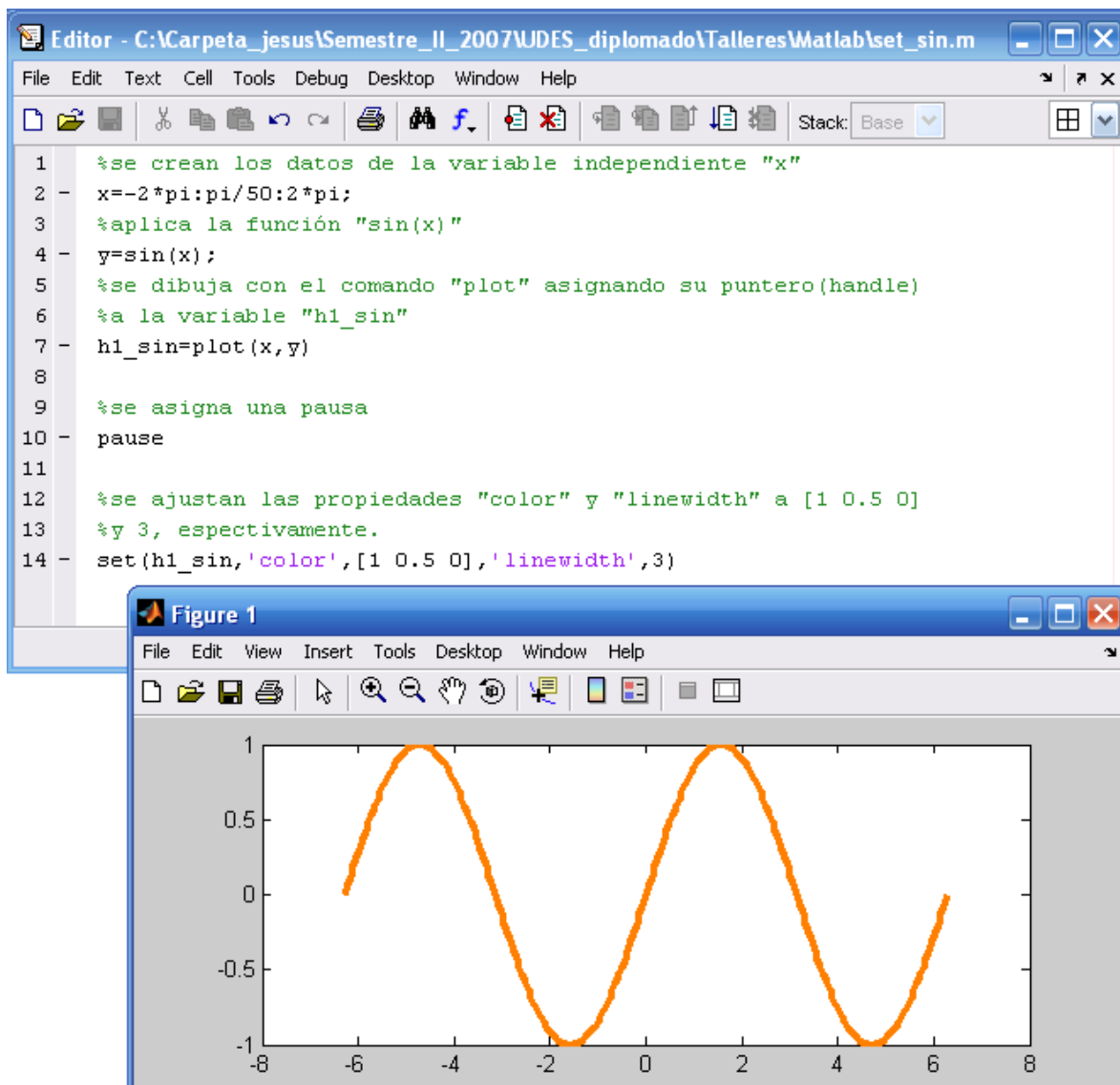
```
1 %se crean los datos de la variable independiente "x"
2 x=-2*pi:pi/50:2*pi;
3 %aplica la función "sin(x)"
4 y=sin(x);
5 %se dibuja con el comando "plot" asignando su puntero(handle)
6 %a la variable "h1_sin"
7 h1_sin=plot(x,y)
8
9 %se asigna una pausa
10 pause
11
12 %se ajustan las propiedades "color" y "linewidth" a [1 0.5 0]
13 %y 3, respectivamente.
14 set(h1_sin,'color',[1 0.5 0],'linewidth',3)|
```

script Ln 14 Col 44 OVR

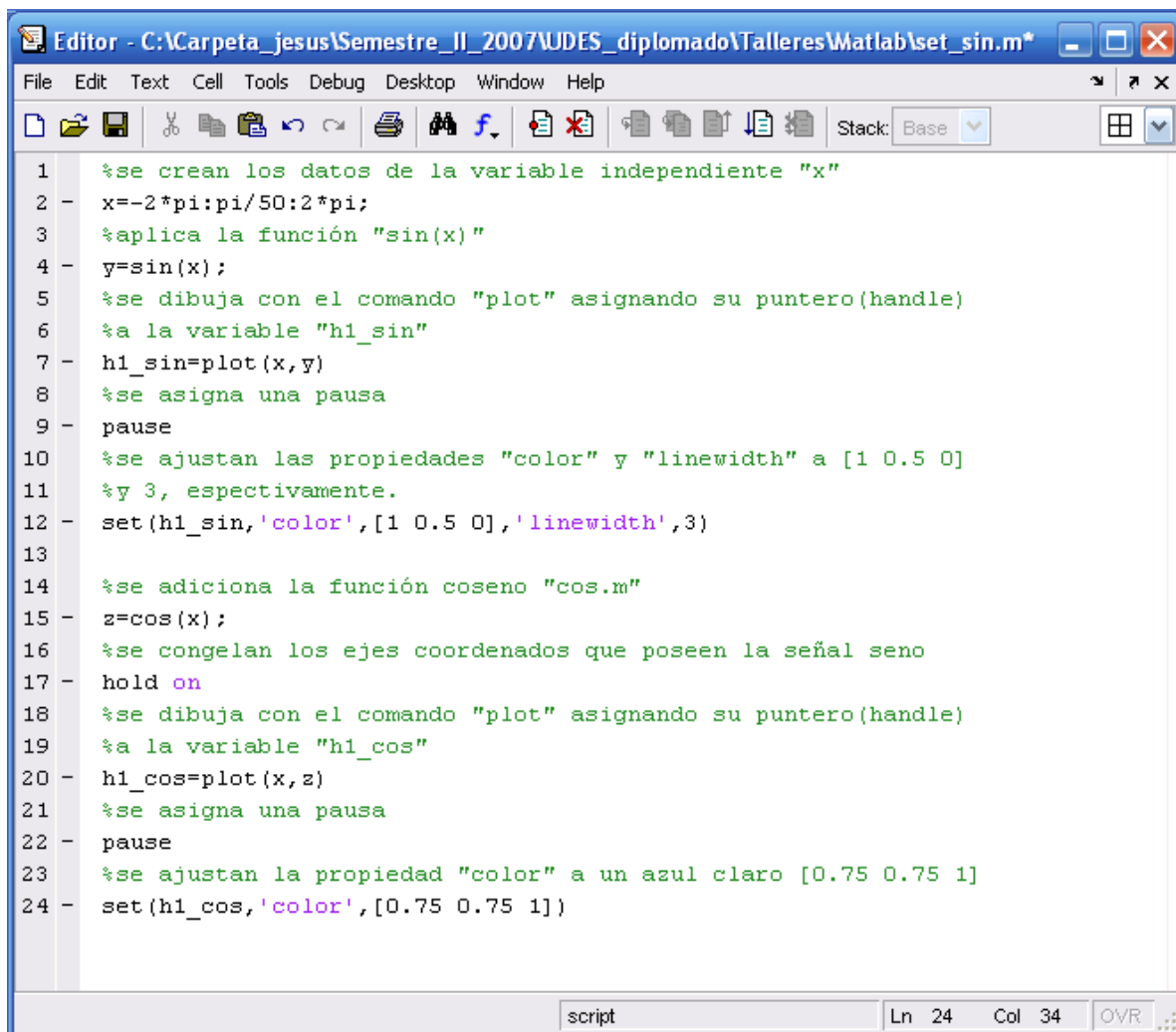
Guarde y ejecute el script,



Cuando se pulsa una tecla cualquiera,



Ahora se adiciona la gráfica de la función coseno "*cos.m*",



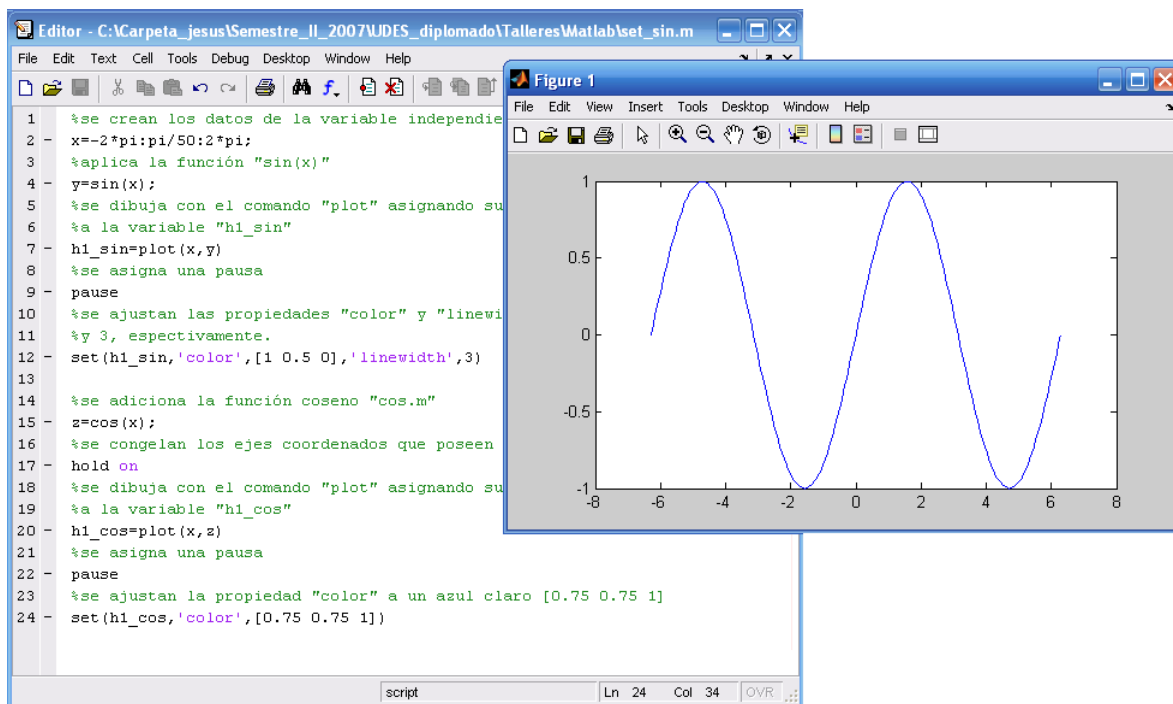
The screenshot shows a MATLAB script editor window titled "Editor - C:\Carpeta\_jesus\Semestre\_II\_2007\UDES\_diplomado\Talleres\Matlab\set\_sin.m\*". The window has a menu bar (File, Edit, Text, Cell, Tools, Debug, Desktop, Window, Help) and a toolbar with various icons. The script content is as follows:

```
1 %se crean los datos de la variable independiente "x"
2 - x=-2*pi:pi/50:2*pi;
3 %aplica la función "sin(x)"
4 - y=sin(x);
5 %se dibuja con el comando "plot" asignando su puntero(handle)
6 %a la variable "h1_sin"
7 - h1_sin=plot(x,y)
8 %se asigna una pausa
9 - pause
10 %se ajustan las propiedades "color" y "linewidth" a [1 0.5 0]
11 %y 3, respectivamente.
12 - set(h1_sin,'color',[1 0.5 0],'linewidth',3)
13
14 %se adiciona la función coseno "cos.m"
15 - z=cos(x);
16 %se congelan los ejes coordenados que poseen la señal seno
17 - hold on
18 %se dibuja con el comando "plot" asignando su puntero(handle)
19 %a la variable "h1_cos"
20 - h1_cos=plot(x,z)
21 %se asigna una pausa
22 - pause
23 %se ajustan la propiedad "color" a un azul claro [0.75 0.75 1]
24 - set(h1_cos,'color',[0.75 0.75 1])
```

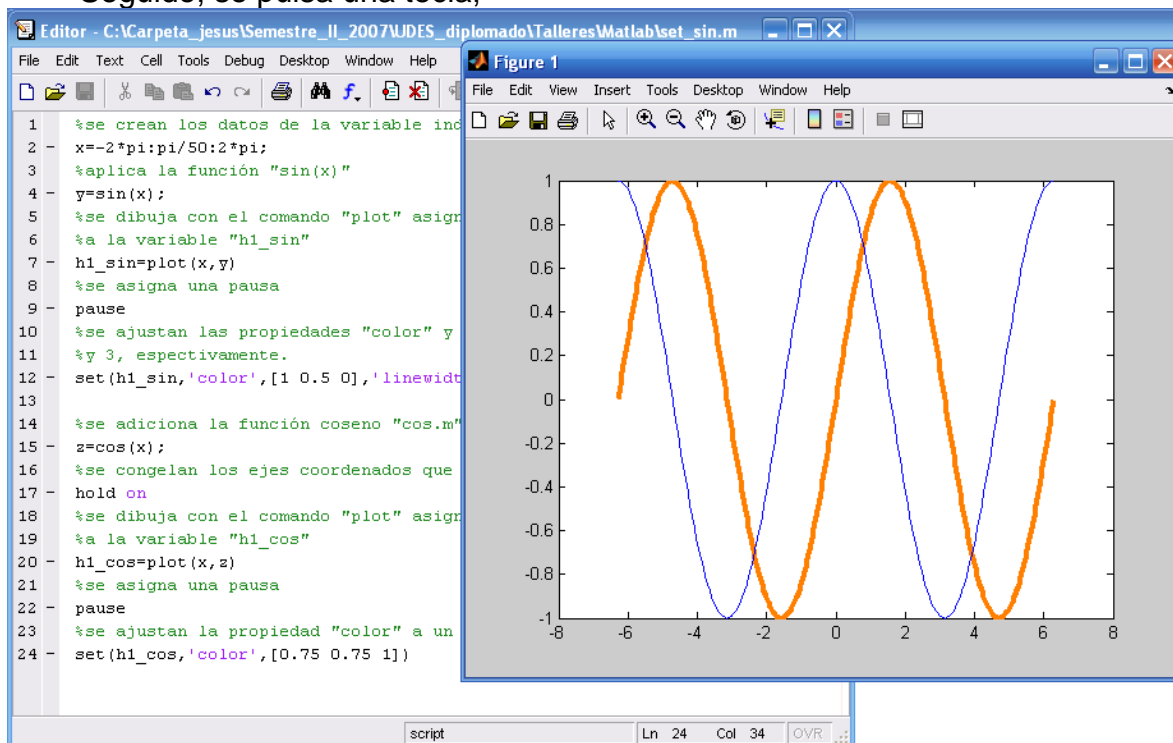
The status bar at the bottom indicates "script", "Ln 24", "Col 34", and "OVR".

Se guarda y ejecuta el programa creado,

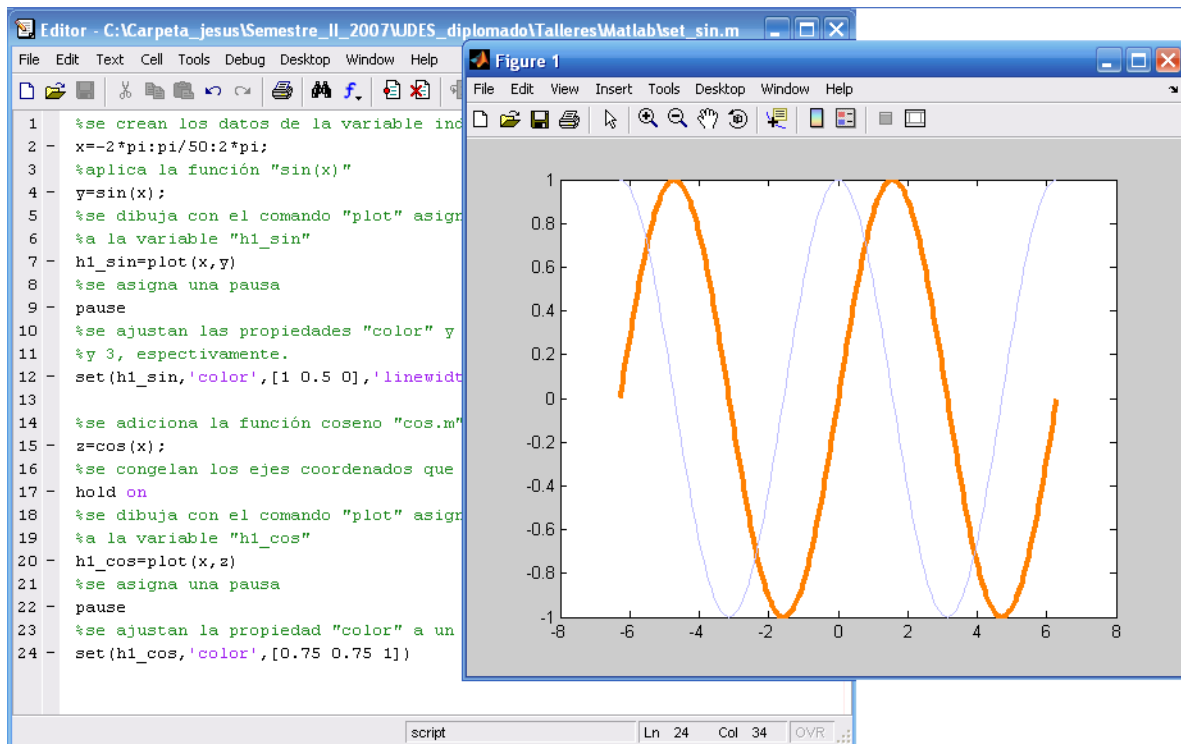




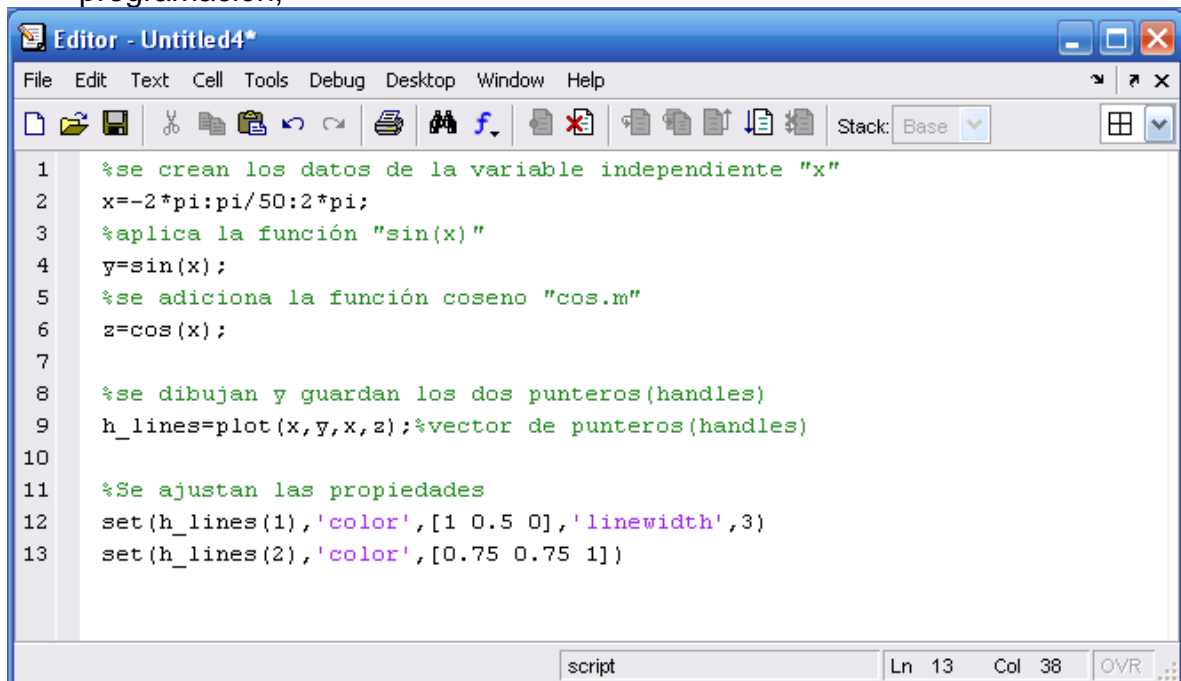
Seguido, se pulsa una tecla,



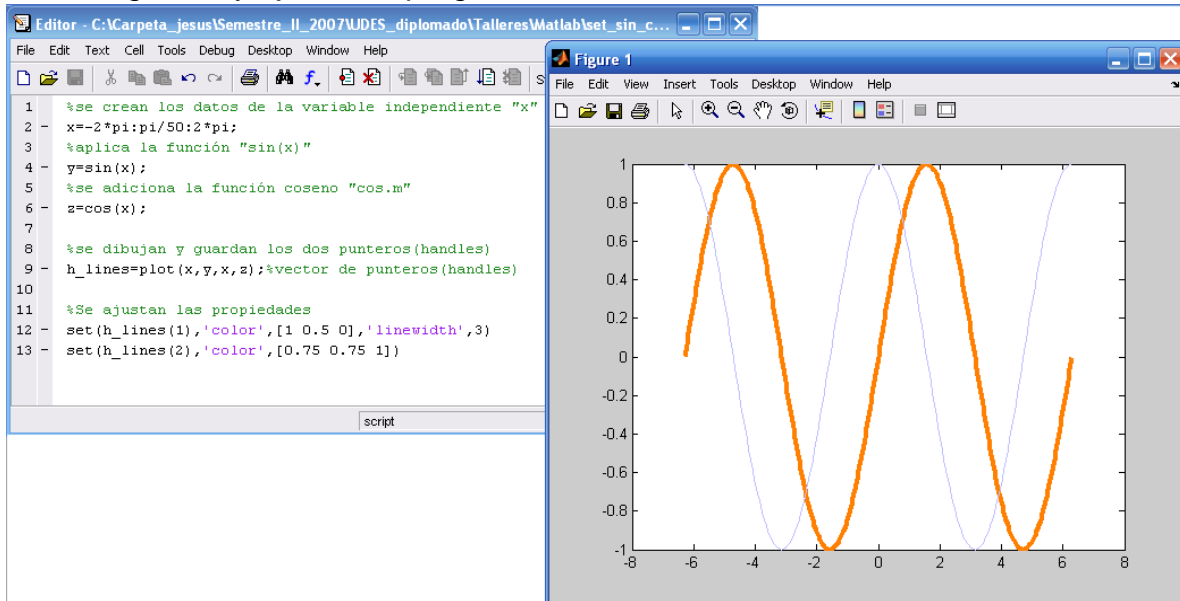
Luego, se pulsa otra tecla,



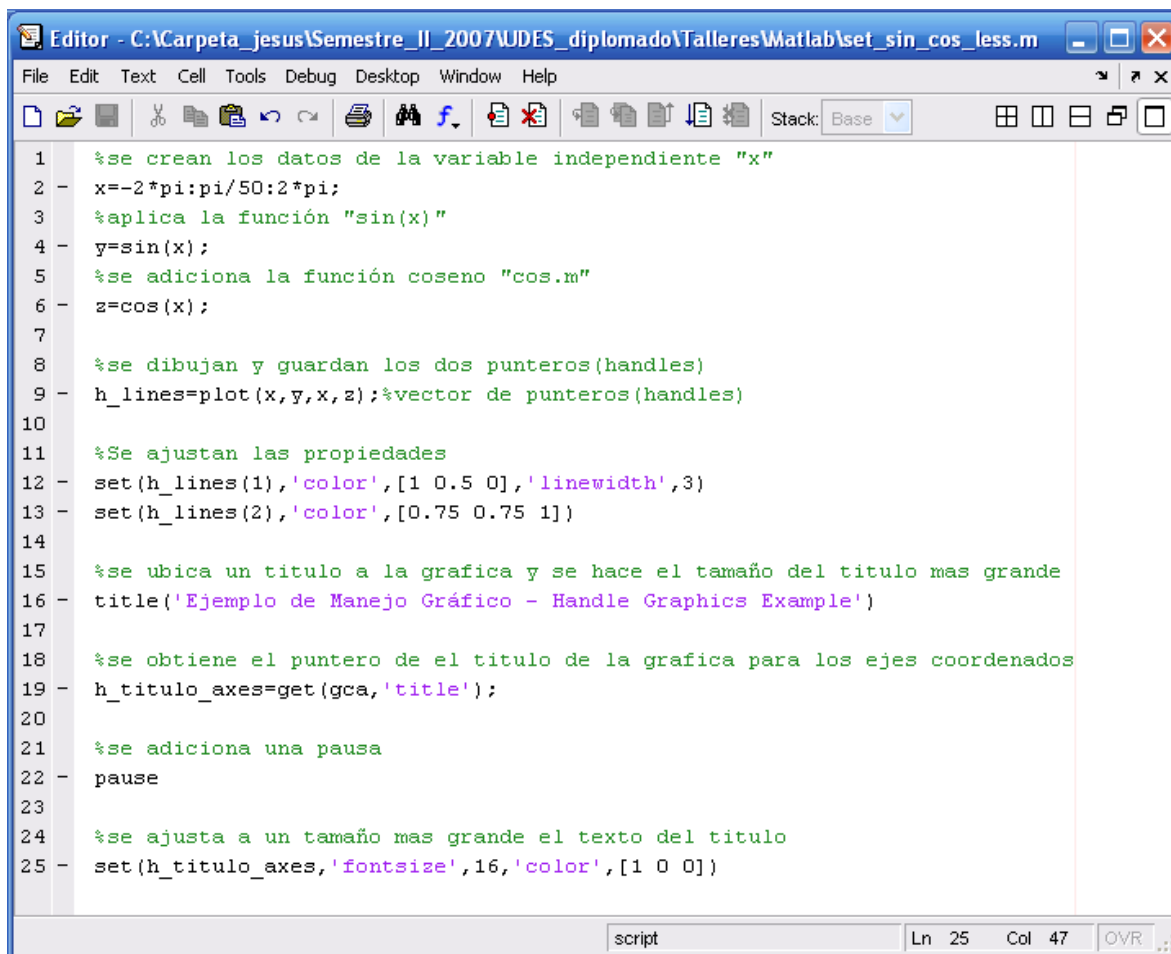
Este proceso se puede llevar a cabo utilizando menos código de programación,



Se guarda y ejecuta el programa,

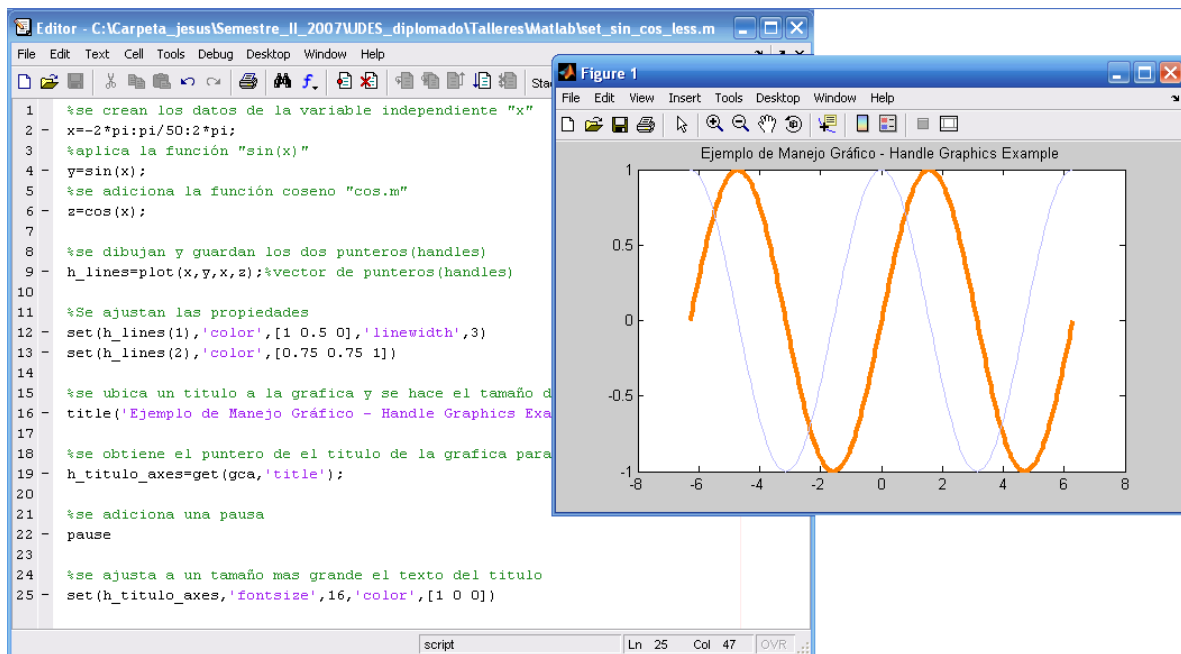


Se puede cambiar el tamaño del titulo ubicados en ejes coordenados,  
Se ubica el siguiente código,

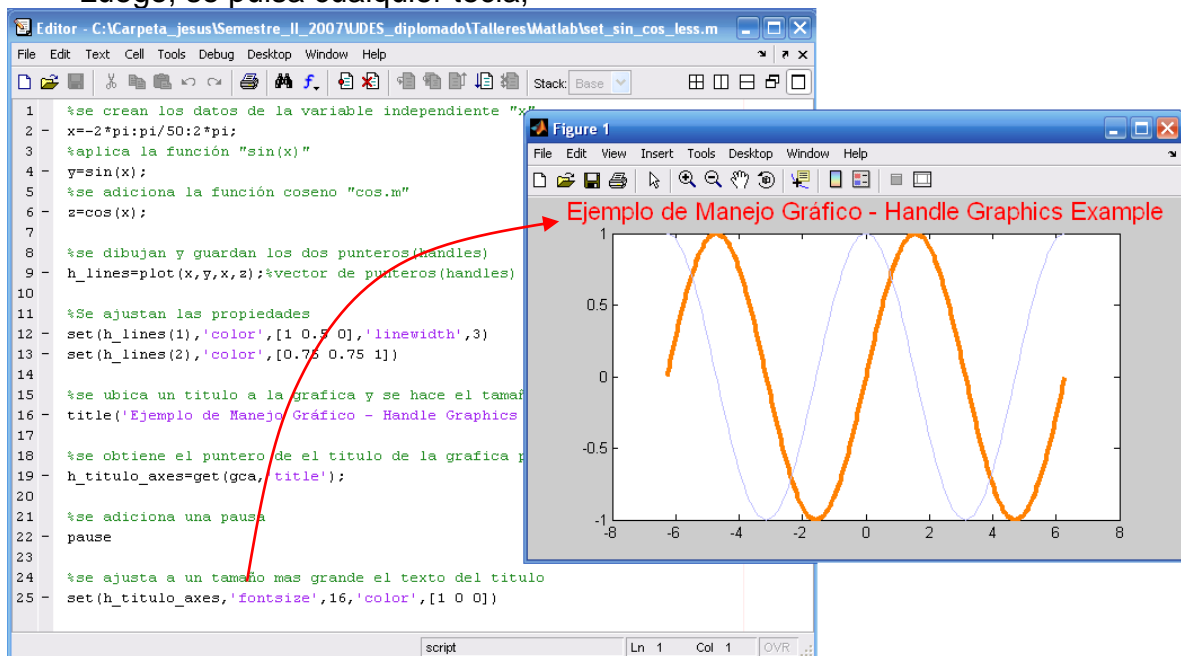


```
1 %se crean los datos de la variable independiente "x"
2 - x=-2*pi:pi/50:2*pi;
3 %aplica la función "sin(x)"
4 - y=sin(x);
5 %se adiciona la función coseno "cos.m"
6 - z=cos(x);
7
8 %se dibujan y guardan los dos punteros(handles)
9 - h_lines=plot(x,y,x,z);%vector de punteros(handles)
10
11 %Se ajustan las propiedades
12 - set(h_lines(1),'color',[1 0.5 0],'linewidth',3)
13 - set(h_lines(2),'color',[0.75 0.75 1])
14
15 %se ubica un titulo a la grafica y se hace el tamaño del titulo mas grande
16 - title('Ejemplo de Manejo Gráfico - Handle Graphics Example')
17
18 %se obtiene el puntero de el titulo de la grafica para los ejes coordenados
19 - h_titulo_axes=get(gca,'title');
20
21 %se adiciona una pausa
22 - pause
23
24 %se ajusta a un tamaño mas grande el texto del titulo
25 - set(h_titulo_axes,'fontsize',16,'color',[1 0 0])
```

Se ejecuta la función,



Luego, se pulsa cualquier tecla,



- 
4. Empleo del comando “subplot.m” para construir varias graficas en la misma ventana. En algunas ocasiones se deben mostrar dos dibujos en una misma ventana debido a que se encuentran relacionados.

Se desea dibujar en la misma ventana las gráficas de magnitud y ángulo de la transformada de fourier de una señal con tres armónicos y frecuencia fundamental de 60 [Hz]. Para ello, realice lo siguiente:

- La señal se puede representar por la siguiente expresión,

$$x(t) = \sin(w_0 t) + 0.6 \sin(3w_0 t) + 0.2 \sin(5w_0 t)$$

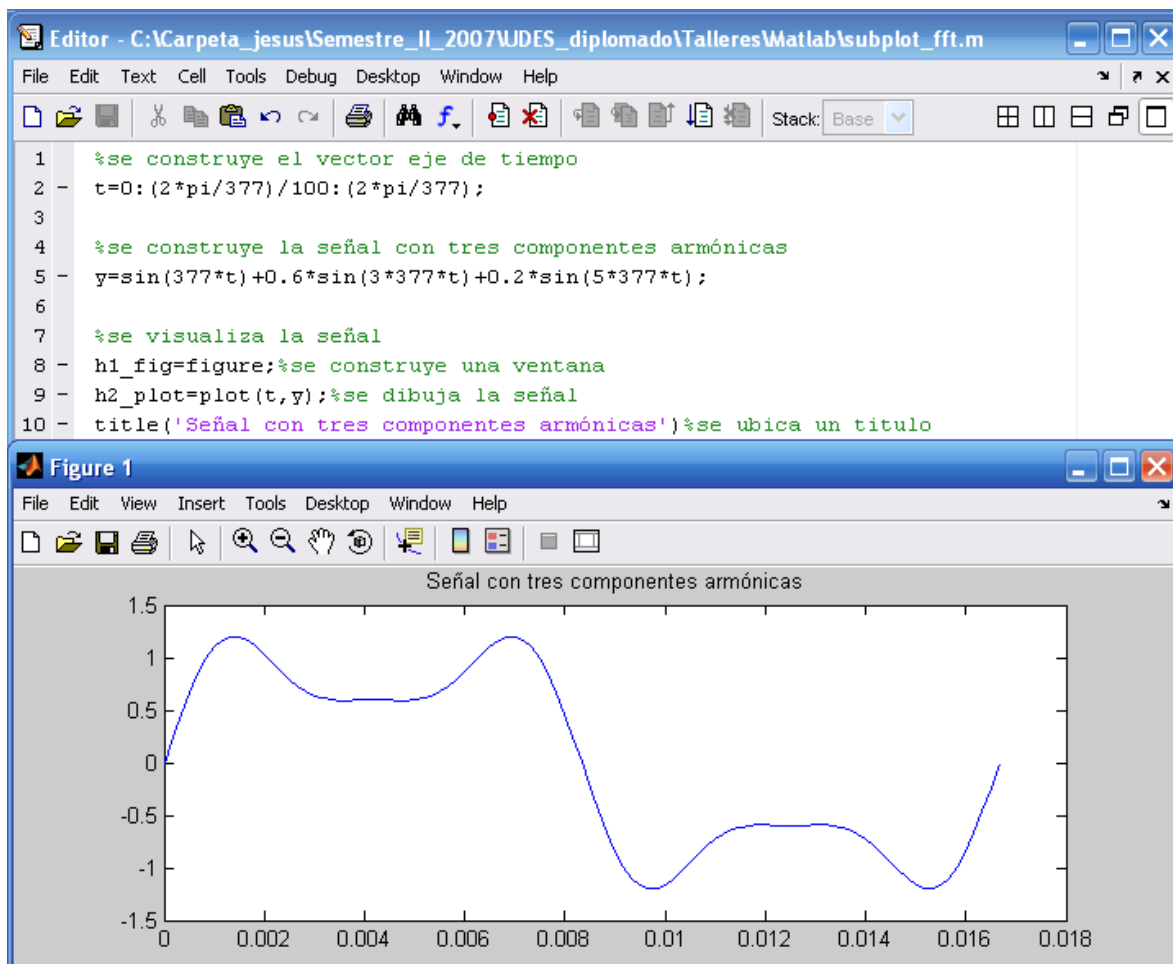
$$w_0 = 2\pi f = 2(60)\pi = 377$$

$$x(t) = \sin(377t) + 0.6 \sin(3(377)t) + 0.2 \sin(5(377)t)$$

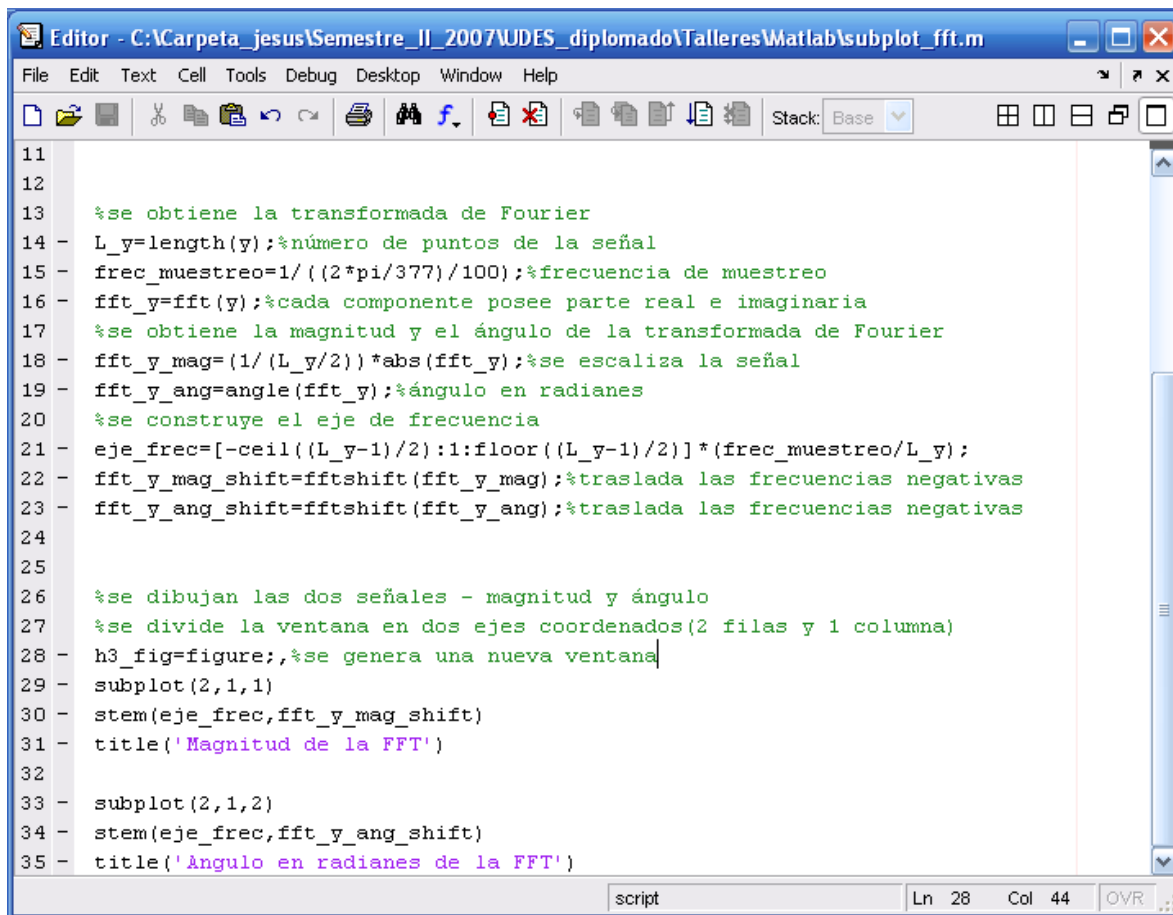
El eje de tiempo se construye para un período,

$$w_0 = \frac{2\pi}{T} \Rightarrow T = \frac{2\pi}{w_0} = \frac{2\pi}{377} = 0.016666$$

- Se introduce el siguiente código de Matlab,



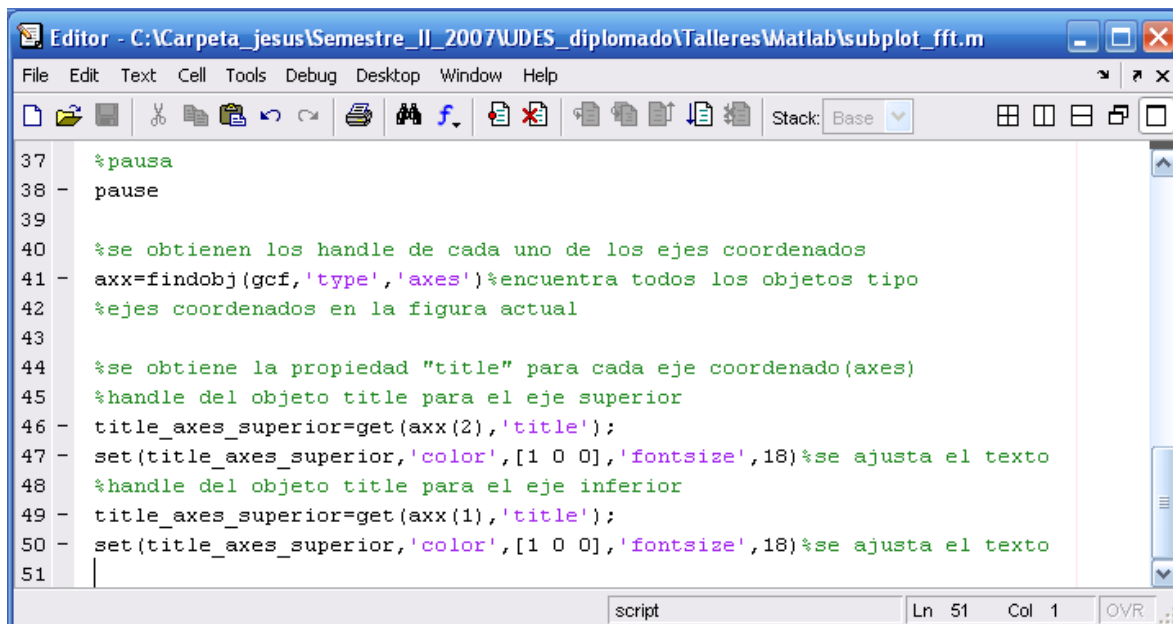
- Se obtiene la transformada de fourier,



```
11
12
13 %se obtiene la transformada de Fourier
14 - L_y=length(y);%número de puntos de la señal
15 - frec_muestreo=1/((2*pi/377)/100);%frecuencia de muestreo
16 - fft_y=fft(y);%cada componente posee parte real e imaginaria
17 %se obtiene la magnitud y el ángulo de la transformada de Fourier
18 - fft_y_mag=(1/(L_y/2))*abs(fft_y);%se escala la señal
19 - fft_y_ang=angle(fft_y);%ángulo en radianes
20 %se construye el eje de frecuencia
21 - eje_frec=[-ceil((L_y-1)/2):1:floor((L_y-1)/2)]*(frec_muestreo/L_y);
22 - fft_y_mag_shift=fftshift(fft_y_mag);%traslada las frecuencias negativas
23 - fft_y_ang_shift=fftshift(fft_y_ang);%traslada las frecuencias negativas
24
25
26 %se dibujan las dos señales - magnitud y ángulo
27 %se divide la ventana en dos ejes coordenados(2 filas y 1 columna)
28 - h3_fig=figure;%se genera una nueva ventana
29 - subplot(2,1,1)
30 - stem(eje_frec,fft_y_mag_shift)
31 - title('Magnitud de la FFT')
32
33 - subplot(2,1,2)
34 - stem(eje_frec,fft_y_ang_shift)
35 - title('Ángulo en radianes de la FFT')
```

- Finalmente se cambia el tamaño y color del texto de cada uno de los títulos de los ejes coordenados.



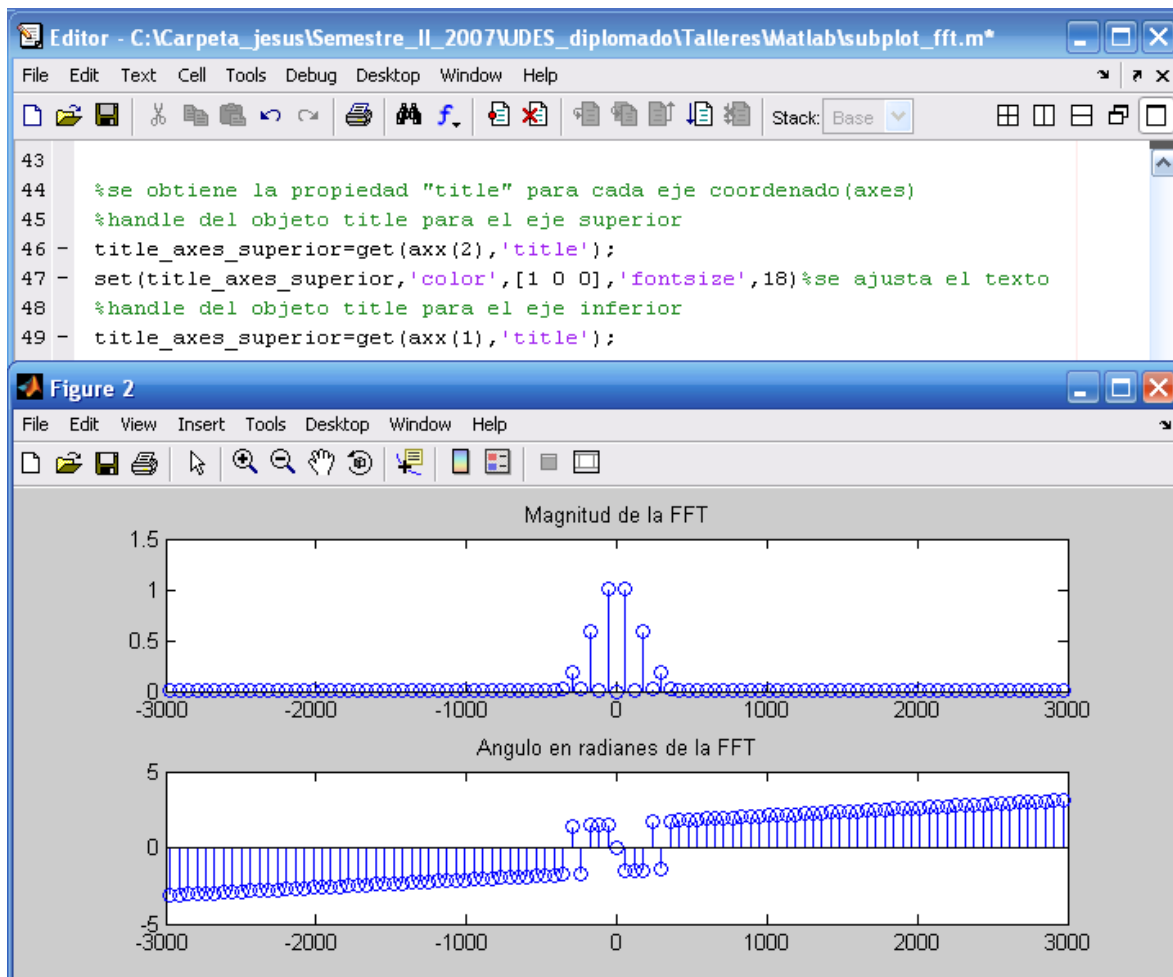


The image shows a MATLAB script editor window titled "Editor - C:\Carpeta\_jesus\Semestre\_II\_2007\UDES\_diplomado\Talleres\Matlab\subplot\_fft.m". The window has a menu bar (File, Edit, Text, Cell, Tools, Debug, Desktop, Window, Help) and a toolbar with various icons. The script content is as follows:

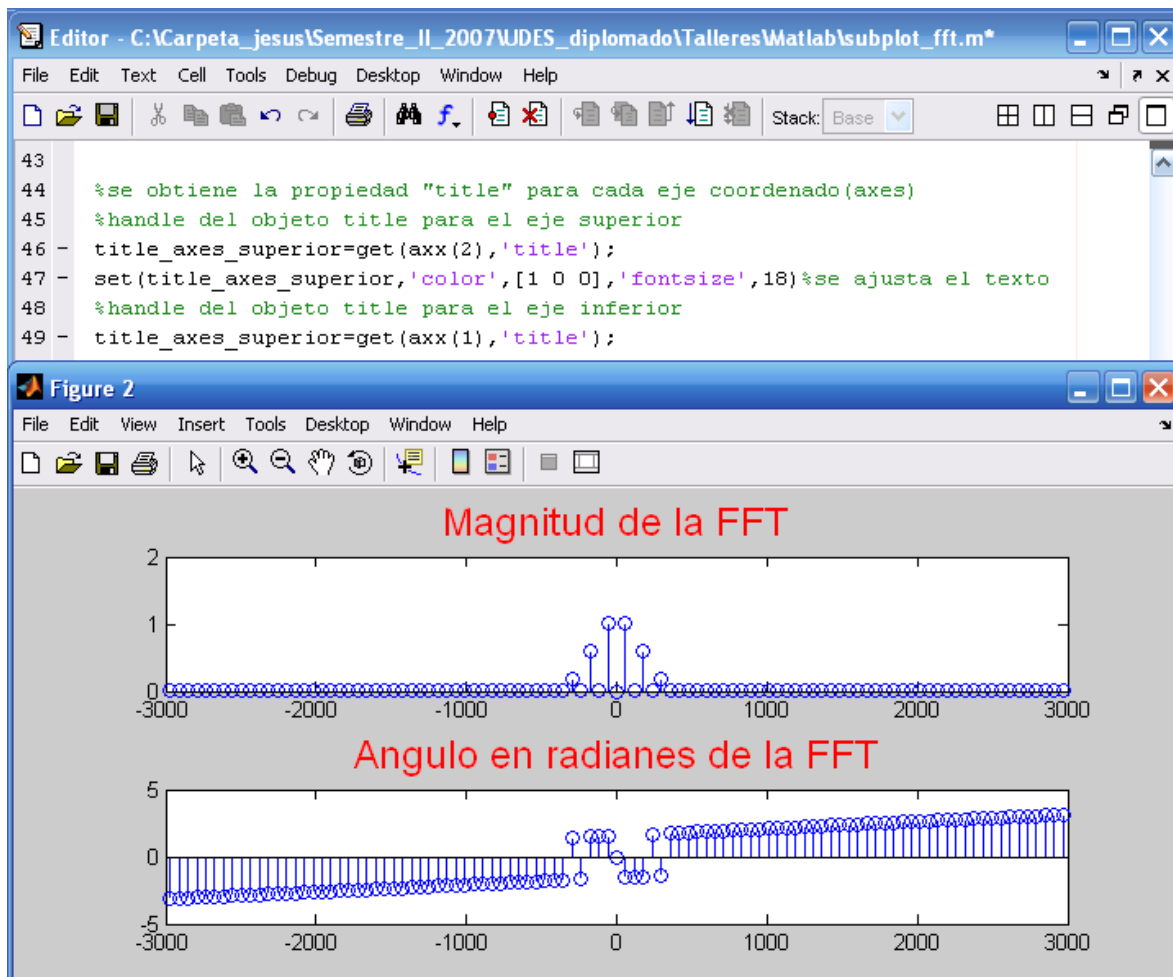
```
37 %pausa
38 - pause
39
40 %se obtienen los handle de cada uno de los ejes coordenados
41 - axx=findobj(gcf,'type','axes')%encuentra todos los objetos tipo
42 %ejes coordenados en la figura actual
43
44 %se obtiene la propiedad "title" para cada eje coordenado(axes)
45 %handle del objeto title para el eje superior
46 - title_axes_superior=get(axx(2),'title');
47 - set(title_axes_superior,'color',[1 0 0],'fontsize',18)%se ajusta el texto
48 %handle del objeto title para el eje inferior
49 - title_axes_superior=get(axx(1),'title');
50 - set(title_axes_superior,'color',[1 0 0],'fontsize',18)%se ajusta el texto
51
```

The status bar at the bottom indicates "script", "Ln 51", "Col 1", and "OVR".

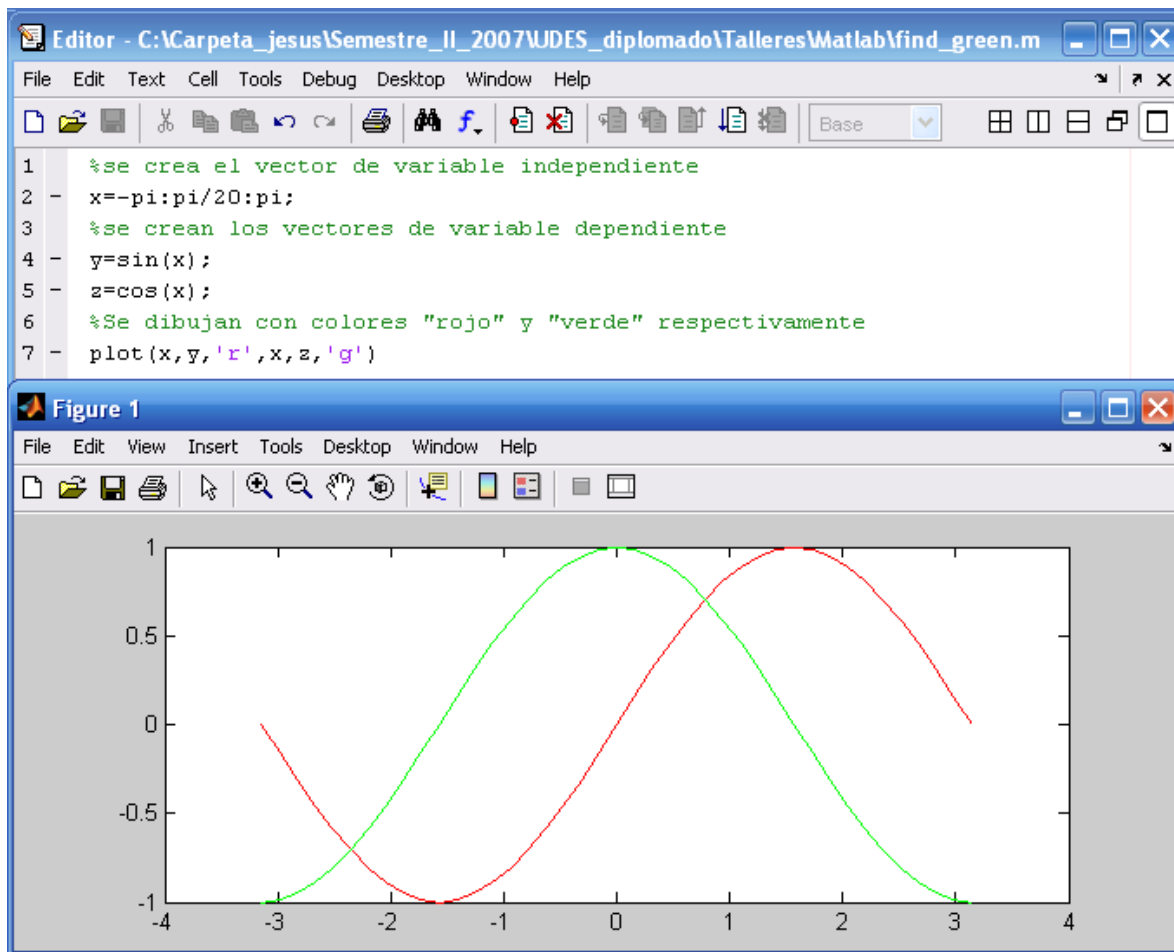
Se ejecuta la aplicación,



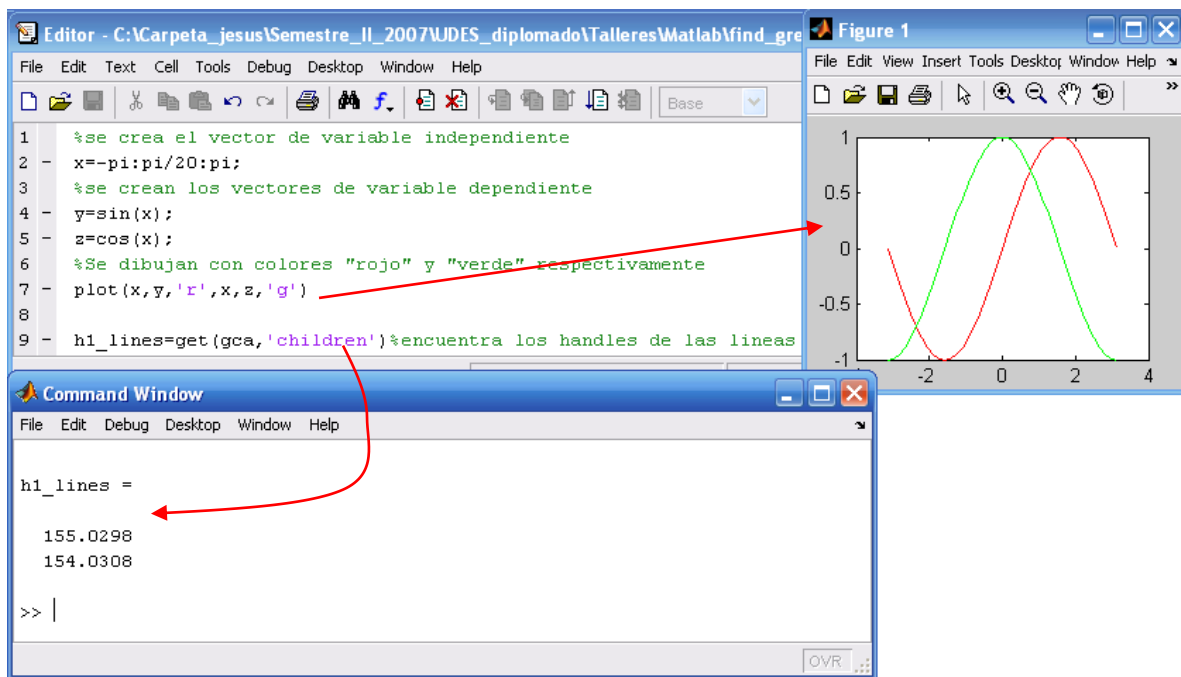
Se pulsa cualquier tecla,



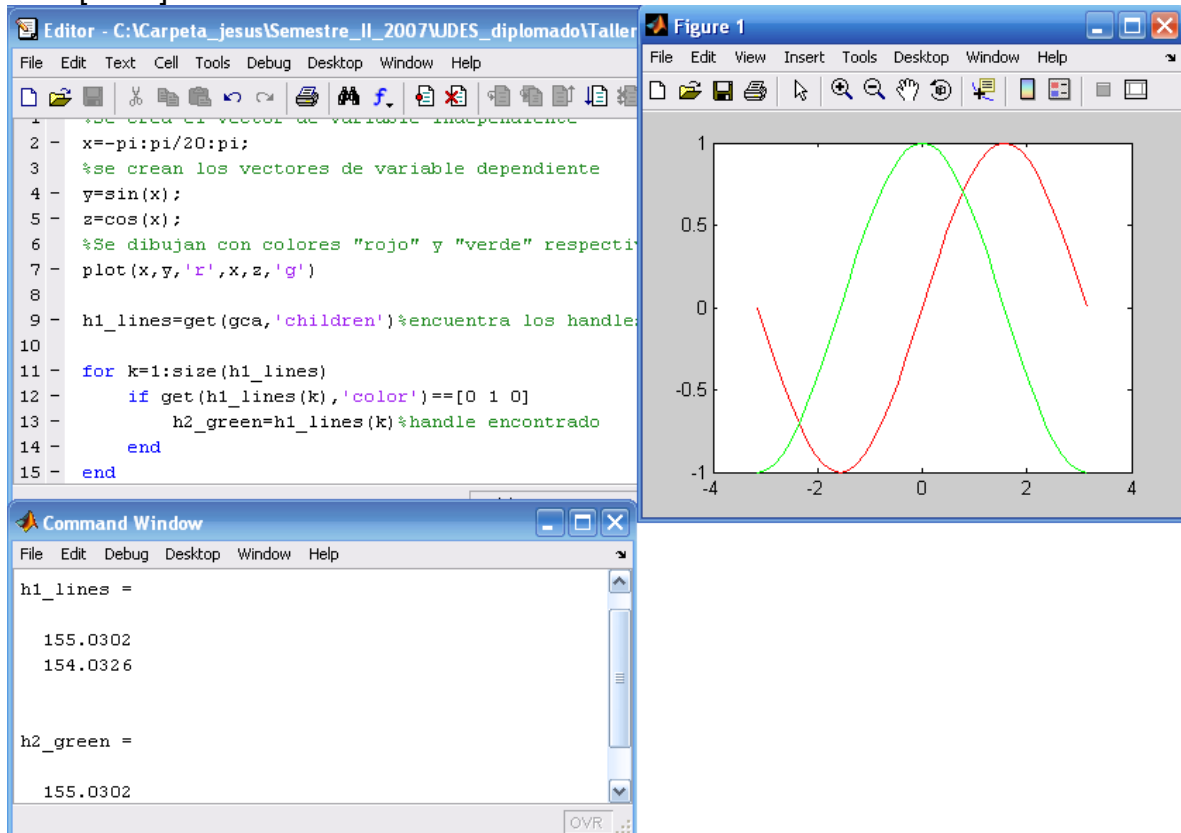
5. Considere el ejercicio de encontrar el handle de un objeto línea verde después de dibujar dos conjuntos de datos en unos ejes coordenados y luego cambiar el color a azul.  
Se dibujarán dos señales, una sinusoidal y otra coseno cada una de ellas a una frecuencia de 1 [rad/s].



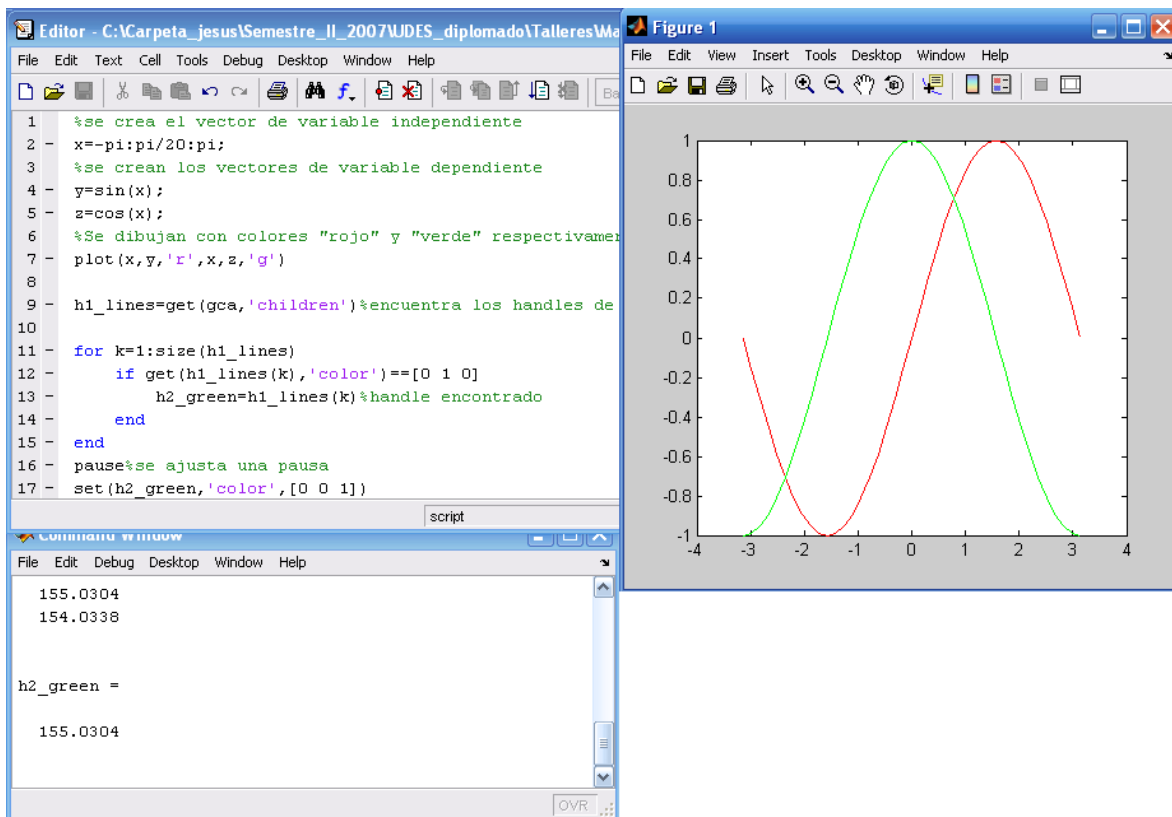
Sobre los ejes dibujados(actuales), se buscan los “*children*”(jerarquía) es decir las líneas que corresponden a las dos señales dibujadas.  
Se adiciona la línea 9,



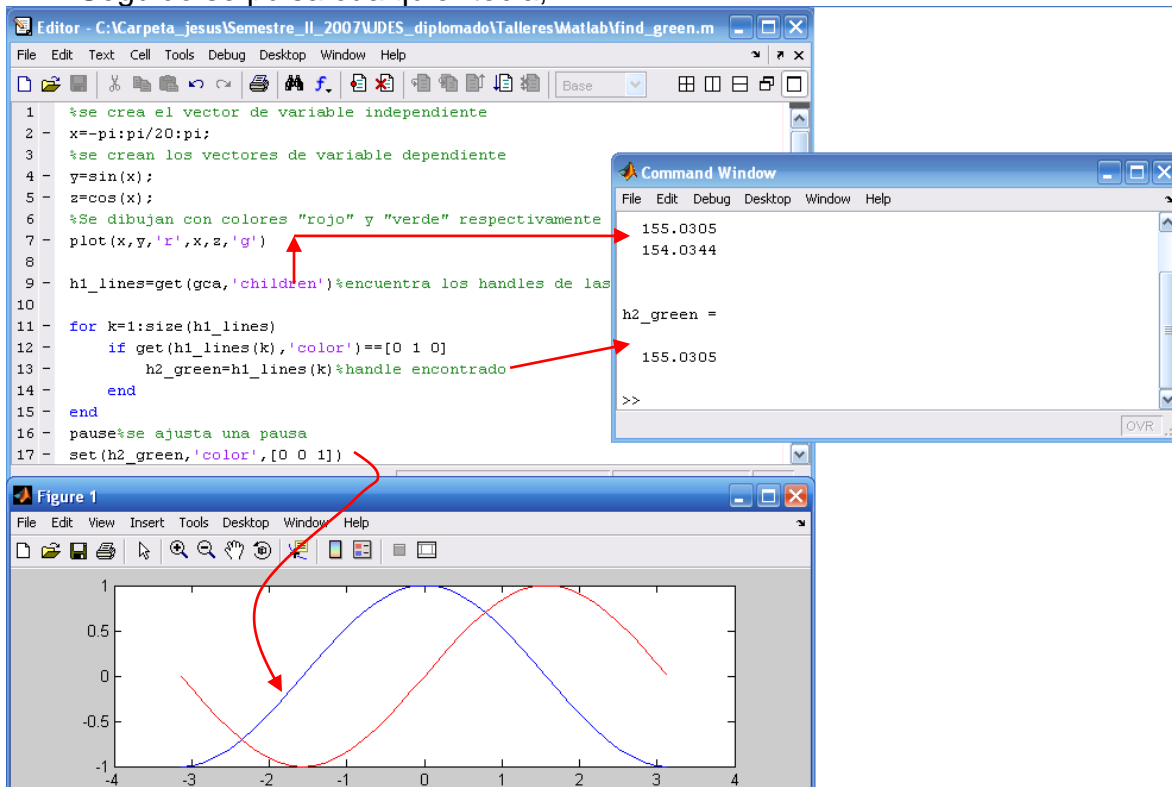
Ahora, se encuentra la línea de color verde, luego la búsqueda es por color [0 1 0].



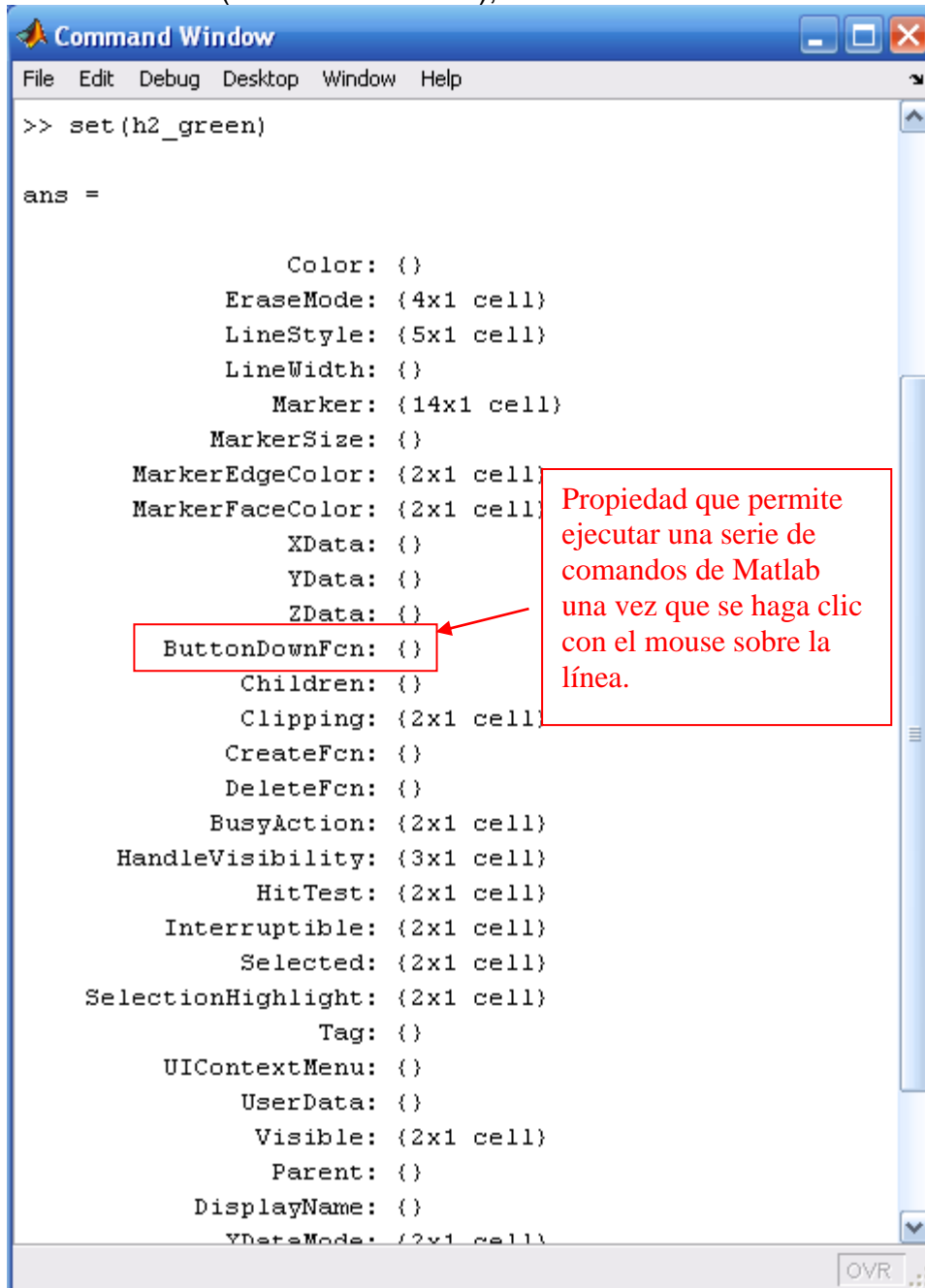
Finalmente la línea dibujada de color "verde" se ajusta a color "azul",



Seguido se pulsa cualquier tecla,

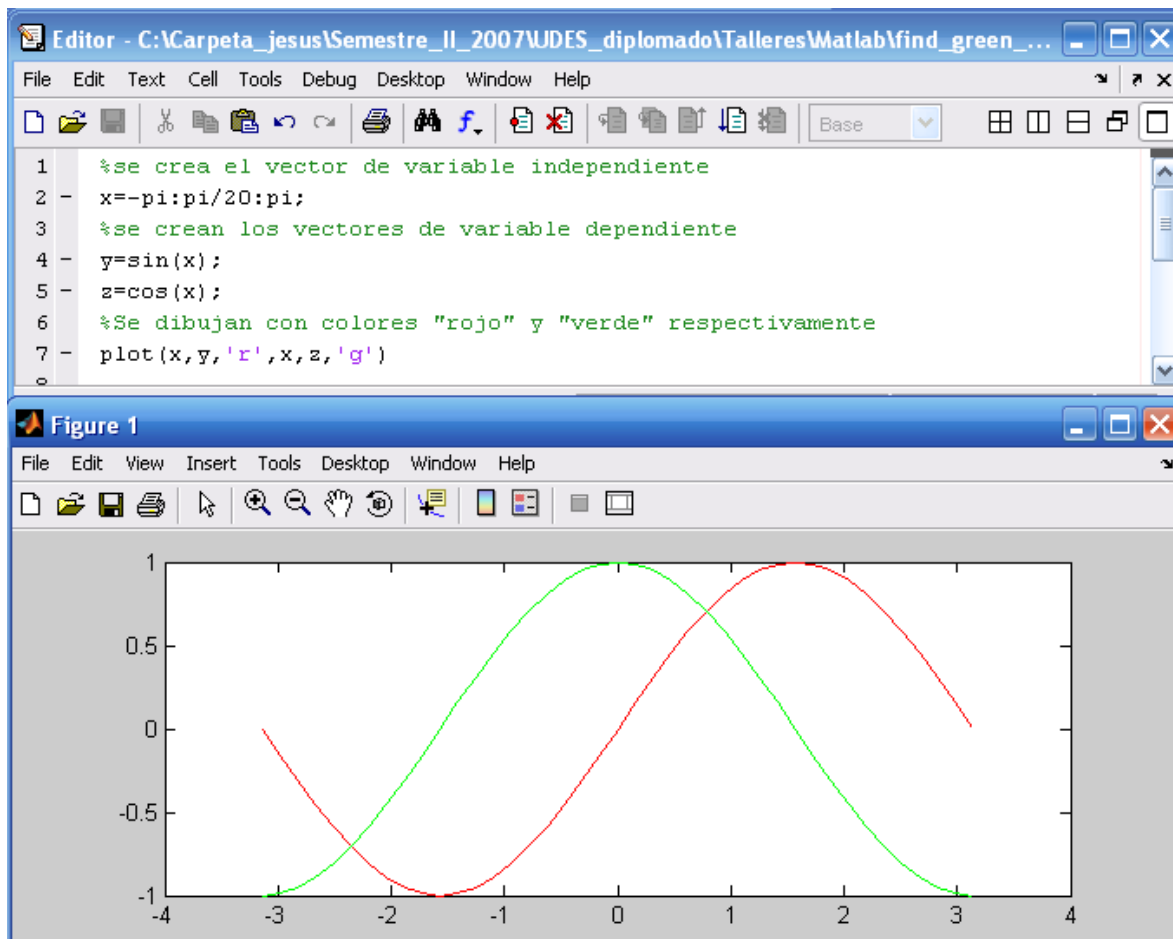


Si se observa las propiedades del objeto línea en la ventana de comandos(command window),



Utilizaremos la función "buttondownfcn" para cambiar el color de la línea a "azul".

Primero que todo, se dibujan las señales,



Ahora, se busca la línea dibujada de color "verde",

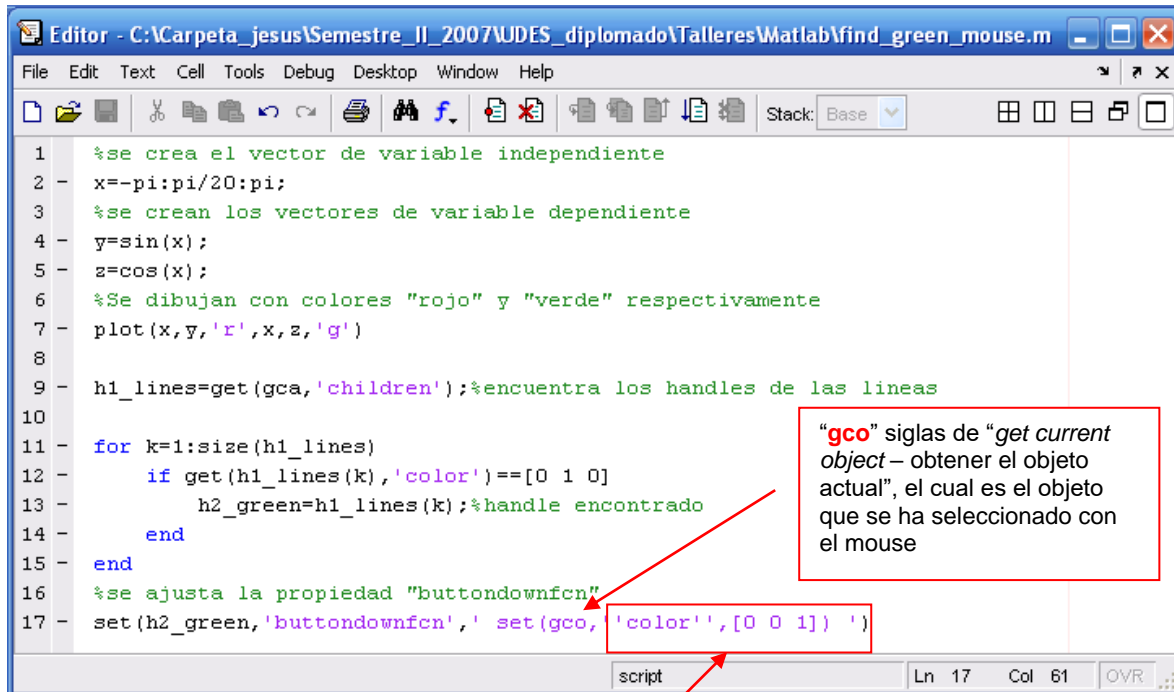
Editor - C:\Carpeta\_jesus\Semestre\_II\_2007\UDES\_diplomado\Talleres\Matlab\find\_green\_...

```
1 %se crea el vector de variable independiente
2 - x=-pi:pi/20:pi;
3 %se crean los vectores de variable dependiente
4 - y=sin(x);
5 - z=cos(x);
6 %Se dibujan con colores "rojo" y "verde" respectivamente
7 - plot(x,y,'r',x,z,'g')
8
9 - h1_lines=get(gca,'children');%encuentra los handles de las lineas
10
11 - for k=1:size(h1_lines)
12 -     if get(h1_lines(k),'color')==[0 1 0]
13 -         h2_green=h1_lines(k);%handle encontrado
14 -     end
15 - end
```

script Ln 17 Col 61 OVR



Luego, se ajusta la propiedad “*buttondownfcn*” del objeto línea de color “verde” encontrado.



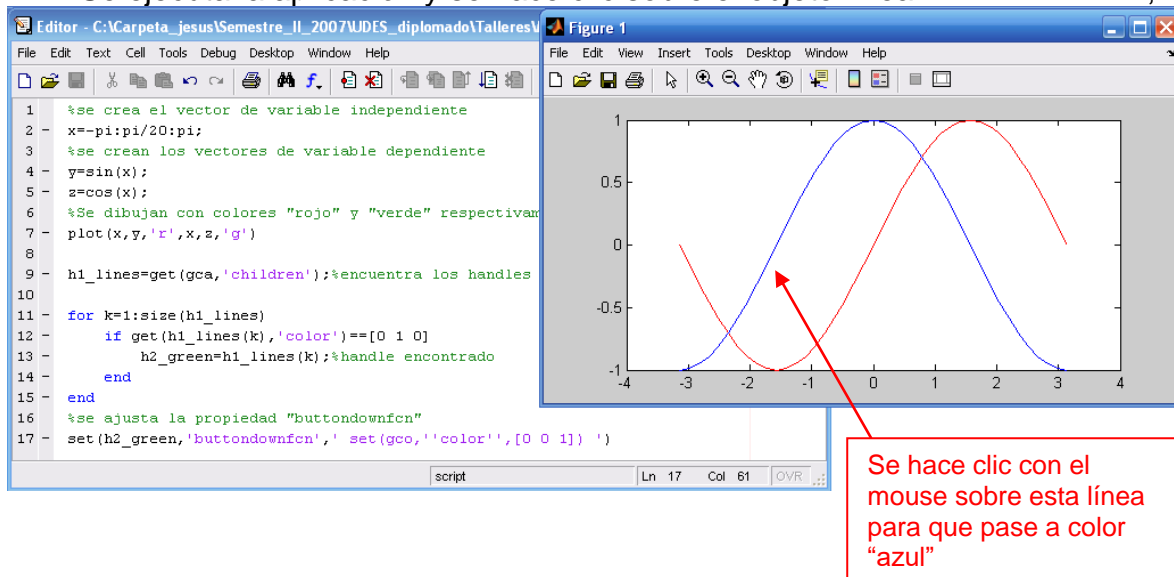
```
1 %se crea el vector de variable independiente
2 x=-pi:pi/20:pi;
3 %se crean los vectores de variable dependiente
4 y=sin(x);
5 z=cos(x);
6 %Se dibujan con colores "rojo" y "verde" respectivamente
7 plot(x,y,'r',x,z,'g')
8
9 h1_lines=get(gca,'children');%encuentra los handles de las lineas
10
11 for k=1:size(h1_lines)
12     if get(h1_lines(k),'color')==[0 1 0]
13         h2_green=h1_lines(k);%handle encontrado
14     end
15 end
16 %se ajusta la propiedad "buttondownfcn"
17 set(h2_green,'buttondownfcn',' set(gcf,'color',[0 0 1]) ')
```

“gco” siglas de “get current object” – obtener el objeto actual”, el cual es el objeto que se ha seleccionado con el mouse

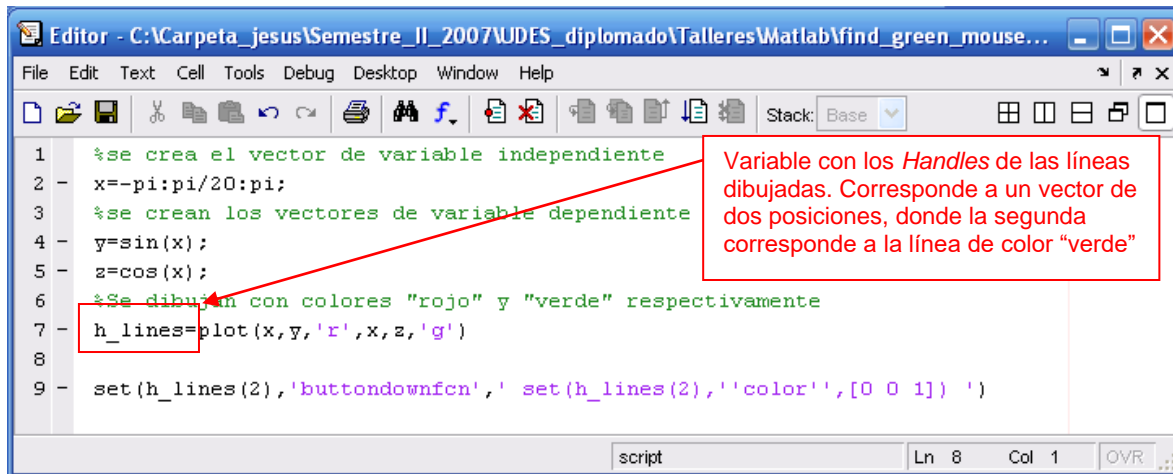
script Ln 17 Col 61 OVR

Se ajusta la propiedad “color” de ese objeto línea de color “verde” seleccionado a color “azul”.

Se ejecuta la aplicación y se hace clic sobre el objeto línea de color “verde”.



Existe una forma alterna de realizar el anterior proceso, es decir, cambiar el color de la línea y es el que se muestra a continuación,



```
1 %se crea el vector de variable independiente
2 x=-pi:pi/20:pi;
3 %se crean los vectores de variable dependiente
4 y=sin(x);
5 z=cos(x);
6 %Se dibujan con colores "rojo" y "verde" respectivamente
7 h_lines=plot(x,y,'r',x,z,'g')
8
9 set(h_lines(2),'buttondownfcn',' set(h_lines(2),'color',[0 0 1]) ')
```

Variable con los *Handles* de las líneas dibujadas. Corresponde a un vector de dos posiciones, donde la segunda corresponde a la línea de color "verde"

Ejecute la aplicación.