**UNIVERSIDAD TECNOLÓGICA NACIONAL**
**FACULTAD REGIONAL BUENOS AIRES**
**ANÁLISIS DE SEÑALES Y SISTEMAS**

# *Gráficos en MatLab*

Ing. Franco Martin Pessana

# *ÍNDICE*

# *Gráficos en MatLab*

## *Figuras*

El manejo de Gráficos bajo MatLab está organizado por medio de las **Ventanas de Gráfico,** también llamadas **Figuras**. Estas ventanas son realmente ventanas para Windows, con lo que tienen asociadas propiedades, menúes y procedimientos.

Cada ventana de gráfico es reconocida por un número, llamado **Handle**. Cualquier operación que quiera realizarse sobre esa ventana debe realizarse por medio de su handle. En realidad el concepto de Handle es más amplio, pues todos los objetos que se utilizan en el sistema gráfico de MatLab se referencian por medios de Handles, no solamente las ventanas.

La función usada para crear una ventana de gráficos es la función **Figure**. Si se la llama sin parámetros crea una nueva figura, devolviendo un **handle** a la figura creada. Si se llama con un handle, selecciona la ventana si ya existe, o crea una ventana con ese handle si no existe.

Ejemplos:

- **h = figure** Abre una nueva ventana y devuelve el handle en *h*.
- **h = figure(2)** Selecciona la ventana con handle 2, y pone un 2 en *h*.
- **figure(3)** Selecciona la ventana con handle 3.

Para ver las propiedades de una figura se utiliza el comando **Get ,** y para modificarlas se utiliza el comando **Set**.

## Menú Predefinido

Una de las propiedades de las figuras es la propiedad **MenuBar** , que determina si utiliza o no el menú predefinido por MatLab para las figuras. Sus valores posibles son :

- **none** No usa el menú definido por MatLab
- **figure** Usa el menú definido por MatLab

Las formas de consultar o modificar su valor son las siguientes :

- **get(h,'MenuBar' )** Muestra el Estado de la Propiedad
- **str= get(h,'MenuBar' )** Asigna el valor a la variable *str*
- **set(h,'MenuBar','none' )** Deshabilita el menú predefinido

en todos los casos la variable *h* es el handle a la ventana cuya propiedad se desea modificar o consultar.

En cualquiera de las dos situaciones (habilitado o deshabilitado) se pueden agregar menúes definidos por el usuario.

## Menú definido por el Usuario

Con MatLab se puede crear para cada Figura una barra de menúes personalizada, cuyo comportamiento es idéntico a las barras de menús de todos los programas Windows. Para ello la

Ventana de Gráfico (o figura) posee en su parte superior una **Barra de Menúes**, que inicialmente puede estar vacía o puede contener el menú predefinido de MatLab.

Para agregar una nueva opción en esta barra se utiliza la función **uimenu**. Debe recibir siempre como parámetro el handle a la ventana donde se va a agregar y devuelve un handle. Las diferentes opciones de la Barra de menús se llaman **Elementos de Menú** y están referenciadas por el handle devuelto por **uimenu**.

Ejemplos:

- **uimenu(h)** Agrega una nueva opción a la barra de menús de la figura *h*.
- **m=uimenu(h)** Devuelve un handle al menú en *m.*

Estos ejemplos agregan un nuevo elemento de menú a la barra de menúes, pero este elemento de menú no tiene ningún texto. Para ello debemos usar la instrucción Set, con la cual le podemos asignar valores a todas las propiedades de los elementos de menú. Las propiedades más importantes son:

- **CallBack** Nombre de la función a llamar cuando se seleccione el elemento de menú
- **Checked** Indica si tiene o no marca de verificación. Sus valores posibles son : **on** y **off**
- **Enable** Indica si está disponible. Sus valores posibles son : **on** y **off**
- **Label** String con el texto del elemento de menú

Por lo tanto, para que un elemento de menú tenga una funcionalidad básica, debemos asignarle una etiqueta y una función con la función Set:

Ejemplos:

- **set(m,'Label','Abrir Archivo')**
- **set(m,'CallBack','openarch')**
- **set(m,'Label','Cerrar','CallBack','close')**
- **set(m,'checked','off'')**

Estas asignaciones de valores a las propiedades también se pueden realizar en el momento de ser definido el menú con la instrucción **uimenu**, poniendo a continuación del handle a la figura la lista de propiedades y valores.

Ejemplos:

- **m=uimenu(h,'Label','Abrir Archivo')**
- **m=uimenu(h,'CallBack','openarch')**
- **m=uimenu(h,'Label','Cerrar','CallBack','close')**
- **m=uimenu(h,'checked','off'')**

Si se quiere saber el valor de una de estas propiedades se debe usar la función **Get**, indicando el objeto y la propiedad que quiere verse.

Ejemplos:

- **get(m,'Label' )** Muestra el Estado de la Propiedad
- **Chk= get(m,'Checked' )** Asigna a la variable *Chk* el estado del menú
- **get(m)** Muestra todas las propiedades del menú

## Resumen

Hasta este punto se ha visto como agregar opciones (elementos de menú) personalizadas a la barra de menús, dejando intacto o eliminando los elementos de menú predefinidos de MatLab. Un ejemplo básico es la siguiente secuencia de comandos:

```
h = figure
```

```
set(h, 'menubar', 'none')
m1=uimenu(h,'label','Cerrar','Callback','close')
m2=uimenu(h,'label','Ayuda','Callback','help')
```

# *Submenúes*

Hasta ahora se vio cómo crear uno o varios elementos de menú en la Barra de Menús de la Ventana de Gráfico. Al seleccionar una opción se ejecuta una rutina o un comando, definidos por medio de la propiedad **Callback**. Nuestro interés es ahora que al seleccionar un elemento de menú se despliegue un submenú. La forma de hacerlo es similar a la forma de definir un elemento de menú para la Barra de Menús. La única diferencia es que en vez de usar como primer parámetro del comando **uimenu** un handle a la figura, se utiliza el handle al elemento de menú.

Ejemplo:

- **m=uimenu(h,'Label','Archivo')**
- **m1=uimenu(m,'Label','Abrir','CallBack','openarch')**
- **m2=uimenu(m,'Label','Cerrar','CallBack','close')**

En este ejemplo, al seleccionar la opción **Archivo** de la Barra de Menúes se despliega un submenú con las opciones **Abrir** y **Cerrar**. Al seleccionar una de estas opciones se ejecuta la rutina o el comando asociado. Vemos también que no es necesario definir una rutina asociada al elemento **Archivo**.

Cada elemento del submenú es por si mismo un **Elemento de menú**, por lo que también puede tener submenúes. La forma de definirlos es idéntica a la recién explicada.

# Menúes con marca de verificación (checked)

Una propiedad útil de los menúes es la posibilidad de poner una marca de verificación (o Check) junto al nombre de un elemento de menú. Esta marca puede servir para indicar si una característica del programa está activada o no. Por ejemplo puede indicar si se muestra o no una barra de estado en la parte inferior de la ventana.

La forma de activar la marca en un elemento de menú es simplemente asignandole el valor **on** a su propiedad **Checked**. Para desactivar la marca se asigna el valor **off** a la misma propiedad.

Ejemplo:

- **m = uimenu(h,'Label','Ver')**
- **m1 = uimenu(m,'Label','Barra de Herramientas','callback','ver_herr')**
- **m2 = uimenu(m,'Label','Barra de Estado','callback','ver_stat','checked','off')**
- **set(m1,'checked','on')**

Luego, en las rutinas **ver_stat.m** y **ver_herr.m** debemos detectar si esta marcado o no, activar o desactivar la barra correspondiente y marcar o deshacer la opción. Esto se puede hacer con un programa como el siguiente:

- 
```
function ver_stat()
old_stat = get(m2,'checked')              Lee el estado (off / on )
if strcmp(old_stat,'on')==1               Detecta si es 'on' (tiene marca)
   set(m2,'checked','off')                Quita la marca
   [ rutina para ocultar la barra de estado ]   Oculta la barra de estado
else                                      En este caso es 'off' (sin marca)
   set(m2,'checked','on')                 Pone la marca
   [ rutina para mostrar la barra de estado ]   Muestra la barra de estado
```

```
end
```

Con este ejemplo, cada vez que se selecciona la opción **Ver/Barra de estado** pueden darse dos casos:

1. **Estaba Marcada**: Quita la marca y oculta la barra de estado
2. **No estaba marcada**: La marca y muestra la barra de estado

# La propiedad Children

Un problema que se presenta en el manejo de los menúes es que debemos guardar en variables los handles a cada elemento de menú, para poder modificar posteriormente su propiedad **checked**. El mismo problema se presenta se quiere utilizar la propiedad **enable** desde cualquier rutina.

En realidad, cada ventana de gráficos tiene una propiedad, llamada **children**, que es un vector con los handles a todos los objetos que contiene, entre ellos los elementos de menú de la Barra de Menúes. El orden de aparición de los handles en el vector es inverso al orden en que se definen los objetos. Por ejemplo, si primero definimos el elemento de menú **Archivo** y luego el elemento de menú **Opciones** , entonces el primer elemento del vector es el handle a **Opciones** y el segundo elemento es el handle a **Archivo**. Si luego agregamos otro objeto, su handle pasa a ocupar el primer lugar, desplazando una posición hacia abajo los dos anteriores.

Cada elemento de menú de la barra de menúes también tiene su propiedad **children**, que guarda los handles a los submenúes que contiene (si tiene alguno). Por lo tanto, si queremos obtener el handle a la opción **Cerrar** del menú **Archivo**, en el ejemplo anterior, debemos realizar los siguientes pasos (suponiendo que **h** es el handle a la figura):

- ch0= get(h,'children') Vector con los handles de los menús de la Figura
  m0 = ch0(2) Handle al menú **Archivo**
  ch1 = get(m0,'children') Vector con los handles de las opciones de **Archivo**
  m1 = ch1(1) Handle a la opción **Cerrar** del menú **Archivo**
  set(m1,'enable','off') Deshabilita la opción **Cerrar**

Para obtener el handle **h** a la figura, podemos utilizar la función **Gcf**, siempre que sea la figura activa.

De esta manera se evita el uso de variables globales para guardar los handles a la figura y los elementos de menú. Con esto podemos tener varias instancias (copias) de una misma ventana, cada una con el mismo menú, sin que la acción sobre una de ellas afecta al resto. Con variables globales, cada instancia usa exactamente los mismos nombres de variables que las demás, por lo que al iniciar una segunda instancia esta sobreescribe las variables globales definidas por la primera.

# Controles

De la misma manera que usamos la instrucción **uimenu** para definir un elemento de menú en una figura, usaremos la instrucción **uicontrol** para definir **controles**. Los controles son todos los objetos dentro de la figura que nos permiten interactuar con ellos. A diferencia de los menúes, no puede definirse un control dentro de otro. Los controles que permite definir el MatLab son los siguientes:

**PushButton** **:** Botón donde uno puede hacer click para ejecutar una opción.
**RadioButton** **:** Botón que permite elegir una opción entre varias.
**CheckBox** **:** Botón que permite habilitar o deshabilitar alguna opción.
**Edit** **:** Cuadro en el que el usuario puede ingresar texto.
**Text** **:** Cuadro donde se puede mostrar texto que el usuario no puede modificar.
**Slider** **:** Barra deslizante.
**PopupMenu** **:** Lista desplegable.

La propiedad que determina el tipo de control es la propiedad **Style**, que debe tomar necesariamente alguno de los valores listados anteriormente. Los siguientes son ejemplos de creación de un control:

- **h=figure(1)**
- **u=uicontrol(h,'style','slider')** Devuelve un handle a un slider.

- **v=uicontrol(h,'style','text')** Devuelve un handle a un cuadro de texto

## La propiedad 'String'

Esta propiedad tiene diferentes significados, dependiendo del tipo de control. Si el control es del tipo **PushButton**, define el texto que aparece en el botón. Si el control es del tipo **RadioButton** o **CheckBox**, define el texto que aparece junto a la casilla de verificación. En un control **Edit** o **Text**, define el texto que aparece inicialmente en el cuadro de texto. Los controles de tipo **Slider** no usan esta propiedad. Por ultimo, para los controles del tipo **PopupMenu**, esta propiedad debe tener una matriz donde cada fila tiene el texto de una opción de la lista desplegable.

Ejemplos:

```
h=figure(1)
u1=uicontrol(h,'style','pushbutton','string','Pulse aqui','position',[20 200 100 20])
u2=uicontrol(h,'style','radiobutton','string','Color','position', [20 180 100 20])
u3=uicontrol(h,'style','radiobutton','string','Grises','position', [20 160 100 20])
u4=uicontrol(h,'style','radiobutton','string','Blanco y Negro','position', [20 140 100 20])
u5=uicontrol(h,'style','checkbox','string','Usa Decimales','position', [20 120 100 20])
u6=uicontrol(h,'style','text','string','Seleccione una opción','position', [20 100 100 20])
u7=uicontrol(h,'style','edit','string','c:\matlab\matlabrc.m','position', [20 80 100 20])
u8=uicontrol(h,'style','popupmenu','string',['Color ';'Grises';' B/N '],'position', [20 60 100 20])
```

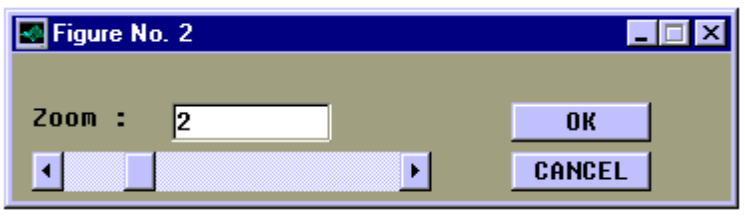
## Las propiedades Value, Max y Min

Esta propiedad (con valores numéricos) tiene diferentes funciones según el tipo de control. Si el control es del tipo **CheckBox** o **Radiobutton**, esta propiedad determina si está seleccionado o no el control. Para ello también interviene otra propiedad llamada **Max**, si el control está seleccionado, el valor de **value** es igual al de **Max**, caso contrario es igual al de **Min**.

# *Ejemplos*

## Barra deslizante

En este ejemplo tenemos un valor numérico que se debe ingresar en un cuadro de edición (**style** = 'edit'). Se desea que este valor también se pueda modificar por una barra deslizante que interactúe con el cuadro de edición; cuando se modifica el valor, la barra debe desplazarse hasta el valor ingresado y cuando se desplaza la barra, el valor del cuadro de edición debe cambiar.




Para ello se deben definir dos **controles**, uno para el cuadro de edición y otro para la barra deslizable. Se usarán los **handles u2, u3** que determinan el cuadro de edición y la barra deslizante respectivamente.

```
u2 = uicontrol('style','edit','string','0','backgroundcolor',[1 1 1],...
     'foregroundcolor',[0 0 0],'position',[82 32 76 16]);
```

```
u3 = uicontrol('style', 'slider','value',0,'backgroundcolor',[0 0 0],...
     'foregroundcolor',[1 1 1],'position',[10 5 200 20]);
```

Con estas instrucciones se tienen definidos los controles, pero aún no se comentó nada sobre su comportamiento. Como primer paso, se debe determinar donde se guardará el valor que halla que modificar. En este caso puede ser la propiedad **value** del cuadro de edición, que no es utilizado internamente por el MatLab (También podría usarse la propiedad **userdata**). Por lo tanto, se debe inicializar su valor en cero.

```
Set(u2, 'value',0);
```

Ahora se debe determinar las funciones asociadas a cada control.

```
set(u2,'callback','pruebaE');
set(u3,'callback','pruebaB');
```

La rutina PruebaE.m será llamada cada vez que se asigne un valor al cuadro de edición y se pulse ENTER o se seleccione otro control. La rutina PruebaB.m será llamada cada vez que se modifique la posición de la barra deslizante.

PruebaE.m se debe encargar de convertir el texto ingresado a número, asignarlo a la propiedad **value** del cuadro de edición y luego mover la barra a la posición correcta.

PruebaB.m se debe encargar de leer la posición de la barra, asignar ese valor a la propiedad **value** del cuadro de edición y luego cambiar el texto por el número correspondiente.

Las rutinas son las siguientes:

## PruebaE.m

```
u2  = gco;                              Handle al cuadro de Edición (Objeto actual)
f   = get(u2,'parent');                 Handle a la ventana (Contenedor del Cuadro)
u3  = findobj(f,'style','slider');      Handle a la barra deslizante (Slider)
mvar = str2num(get(u2,'string'));       Convierte el String a número
if mvar>=0 & mvar<=10                   Verificación de Rango (A elección)
    set(u3,'value',mvar/10);            Asigna a la barra el valor (entre 0 y 1)
    set(u2,'value',mvar);               Guarda el valor en el Cuadro de Edición
else                                    Valor fuera de Rango
    mvar = get(u2,'value');             Lee el valor anterior
    set(u2, 'string',num2str(mvar));    Cambia el texto ingresado
    set(u3,'value',mvar/10)             Corrige la barra de menú (No es necesario)
end
drawnow                                 Muestra todo actualizado
```

## PruebaB.m

```
u3  = gco;                              Handle a la barra deslizante (Objeto Actual)
f   = get(u3,'parent');                 Handle a la ventana (Contenedor del slider)
u2  = findobj(f,'style','edit');        Handle al cuadro de Edición (Edit)
mvar = get(u3,'value')*10;              Lee la posición del Slider
set(u2,'string',num2str(mvar));         Cambia el texto del cuadro de edición
set(u2,'value',mvar);                   Guarda el valor en al Cuadro de edición
drawnow                                 Muestra todo actualizado
```

## Ejercicios

1) Optimizar las rutinas para que el rango de valores permitidos esté guardado en las propiedades **max** y **min** de la Barra Deslizante, evitando el uso de un factor de escala (el 10)

2) Hacer una rutina que reciba como parámetro un handle a otra ventana. Esta rutina debe mostrar una ventana como la de la figura, sin reemplazar la existente. En el cuadro de edición se debe mostrar la altura en pixeles de la otra ventana. Al seleccionar **OK** debe modificar la altura de la otra ventana según el valor ingresado y cerrar esta ventana. Al pulsar **CANCEL** debe cerrar esta ventana sin realizar modificaciones en la otra.

# *Manejo de imágenes*

Aquí se propone un método para manejar imágenes dentro de una aplicación desarrollada completamente en MatLab. La idea básica es trabajar con dos imágenes, una sobre la cual se aplican las transformaciones y otra que mantiene la imagen anterior a la transformación. De aquí en adelante se llamarán **Imagen 1** y **Imagen 2** respectivamente. El núcleo de la aplicación es una ventana de gráficos de MatLab, que inicialmente está vacía. Esta ventana sólo tiene una barra de menúes con todas las opciones disponibles.

Si se decide no usar variables globales, se deben definir dónde estarán guardadas las imágenes. Se podría pensar en usar la propiedad **UserData** de la ventana, pero solo puede contener una matriz, y en este caso se utilizará para guardar la lista de **handles** a los menús. Una posibilidad es definir un submenú llamado **Ver** con dos opciones: "**Ver imagen 1"** y "**Ver imagen 2"**, que también son elementos de menú, y luego guardar las matrices con las imágenes en la propiedad **UserData** de estas dos opciones.

Por lo tanto, las variables "globales" de la aplicación estarán guardadas de la siguiente manera:

- **Lista de Handles a los menús**: Propiedad **UserData** de la Ventana
- **Imagen 1**: Propiedad **UserData** del elemento de menú **"Ver Imagen 1"**
- **Imagen 2**: Propiedad **UserData** del elemento de menú **"Ver Imagen 2"**

Una primera aproximación de una aplicación debe posibilitar leer una imagen, colocándola en la **Imagen 1**, intercambiar ambas imágenes y visualizar cualquiera de las dos.

## El programa

Todas las rutinas principales del programa estarán dentro de un mismo archivo, llamado en este caso **cur_main.m**. Cada vez que se llama a este archivo, se le debe pasar un parámetro (línea 1), que le indicará qué rutina debe ejecutar. Si el string es **'inicio'** se crea la ventana y se inicializan las variables, aunque también se lo puede llamar sin parámetros, con lo que supone que debe ejecutar la rutina de inicio (Líneas 2 a 4). El núcleo del programa es una sucesión de sentencias **IF**, **ELSEIF**, **END**. En cada sentencia se compara el parámetro recibido con un string y, si son iguales, se ejecuta el bloque de instrucciones correspondientes (Líneas 5, 6, 7, 8, 9, 10, 11)

## Listado 1 (Núcleo del programa)

```
1    function cur_main(opcion)
2    if nargin<1
3        opcion = 'inicio'
4    end
5    if strcmp(opcion,'inicio')
( ... Creación de la ventana e inicialización de las variables )
6    elseif strcmp(opcion,'arch_abre')
( ... Apertura de una imagen )
7    elseif strcmp(opcion,'arch_sale')
( ... Finalización del programa. Cierre de la ventana )
```

```
8       elseif strcmp(opcion,'visu_img1')
( ... Visualización de la imagen 1)
9       elseif strcmp(opcion,'visu_img2')
( ... Visualización de la imagen 2)
10      elseif strcmp(opcion,'visu_camb')
( ... Intercambio entre la imagen 1 y la imagen 2)
11      end
```

Por lo tanto, la primera vez que se llama el programa se lo debe hacer con el parámetro **'inicio'** o sin parámetros. La sintaxis sería la siguientes:

```
cur_main
cur_main('inicio')
```

Al llamar de esta forma al programa, se ejecuta la primera opción (línea 5), por lo que en este bloque debemos crear la ventana de trabajo, asignarle una barra de menús e inicializar las variables (Lista de handles a los menúes). La secuencia de instrucciones puede ser la siguiente:

## Listado 2 (Apertura de una imagen)

```
1       h       = gcf;
2       l = get(h,'userdata');
3       fl      = l(5);
4         [x,map] = pdi_load;
5         set(fl,'userdata',x);
```

En la primera instrucción (línea 1) se recupera el handle a la ventana abierta (que es en la que seleccionamos una opción de menú). Para ello se usa la función **gcf** (Get Current Figure) y se asigna el handle a la variable **h**. La imagen se va a guardar en la propiedad **UserData** del elemento de menú **Ver Imagen 1**, por lo que se debe conseguir un handle a él. La forma de hacerlo es recordando que en la propiedad **UserData** de la ventana se había guardado la lista de handles a los menúes. Por lo tanto se puede guardar en la variable **l** esta lista utilizando la instrucción **get** (línea 2).

Hay que recordar en que posición de la lista está el handle a la opción **Ver Imagen 1**, que en este caso es la posición numero 5 (ver definición de la variable **a** en el listado 1). Por lo tanto se puede asignar este handle a la variable **fl**, que apuntará al elemento de menú **Ver Imagen 1** (línea 3).

Para cargar la imagen se puede utilizar la función **pdi_load**, que no necesita parámetros y devuelve la imagen cargada y el mapa de colores (línea 4).

Por último, se debe guardar la imagen en la propiedad **UserData** del elemento de menú **Ver Imagen 1** , que está apuntado por el handle **fl** (línea 5). En este caso no se guarda en ningún lugar el mapa de colores **map**, pues se supone que se está trabajando con 256 niveles de gris, por lo que cuando haga falta un mapa, podrá crearse con la función **gray** del MatLab.