Concepto de Sistema

Ahora vamos a realizar en matlab un par de sistemas sencillos. Los textos de teoría definen un sistema como cualquier ente capaz de transformar señales. Para afianzar la idea de que un sistema transforma señales vamos a usar el concepto de "función de matlab" (que ya conocemos de la práctica introductoria) para crear nuestros primeros sistemas.

Un sistema lo vamos a definir como una función que podemos crear con el editor de matlab (o con cualquier otro editor) utilizando una cabecera del tipo:

```
function [y,ty] = Nombre(x,tx)
%
% Instrucciones que generen "y" y "ty"
% (a partir de x y tx).
%
% Estas instrucciones se graban en "Nombre.m"
%
```

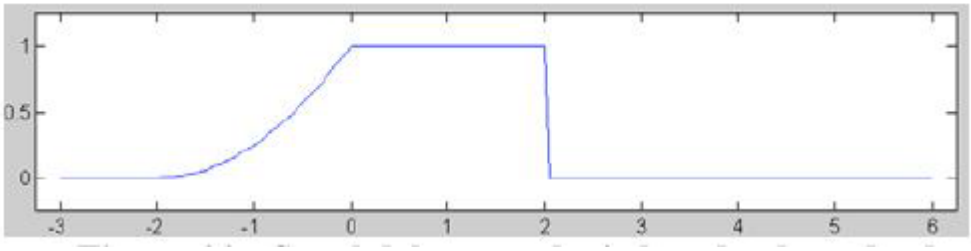
Nótese que hemos querido reforzar la idea de que una señal en matlab es una base de tiempos y un vector de valores. La función recibe como argumentos los dos vectores de la señal **x(t)** (entrada) y devuelve como resultados los dos vectores de la señal **y(t)** (salida).

Por ejemplo, para implementar el sistema que eleva al cuadrado ($\mathbf{y(t)=[x(t)]^2}$) podemos usar la función:

```
function [y,ty] = Cuadrado(x,tx)
ty = tx; % La base de tiempos es la misma
y = x.^2; % Los valores se elevan al cuadrado
```

Pruebe esta función (mantenga los valores de t y x

```
[y, ty]=cuadrado(x, t);
```



Señal elevada al cuadrado

DISEÑO DE UNA INTERFAZ GRAFICA USANDO *GUIDE* PARA GENERAR UNA *GUI* (GRAPHIC USER INTERFACE)

Preparación

Antes de asistir al laboratorio Ud. Debe:
-Ver el demo que tiene Matlab sobre Diseño de GUIs
-Leer los help que posee Matlab sobre el mismo tema
-Tener a disposición (en un disco) el logo de la CETI y la rutina fftplot. Esta rutina permite determinar el contenido espectral de una señal y graficar o su espectro o su Densidad Espectral de Potencia. Asegúrese de comprenderla.

Conceptos sobre Diseño de Interfaces gráficas (GUI)

Introducción:

GUIDE es un ambiente de desarrollo que permite crear interfaces gráficas con el usuario, tales como botones y ventanas de selección, ventanas gráficas, menús, ejes para graficar, etc. Cuando en el ´command window´ se escribe guide, se ofrece la posibilidad de abrir hojas de trabajo ya creadas (p.ej. >>guide archivo.fig) o una nueva sobre la cual se irán agregando componentes. Lo que se cree aquí se guardará con la extensión **.fig.**

La primera vez que uno salva la interfaz que está diseñando se crea también un archivo **.m** sobre el cual habrá que programar lo que se quiere ver o controlar desde el GUI

Una vez que se diseña la interfaz gráfica (GUI) que uno desea fijando las características de botones, ventanas, etc. que la conforman, se puede entonces programar dicha interfaz con el editor de archivos **.m**

Herramientas:

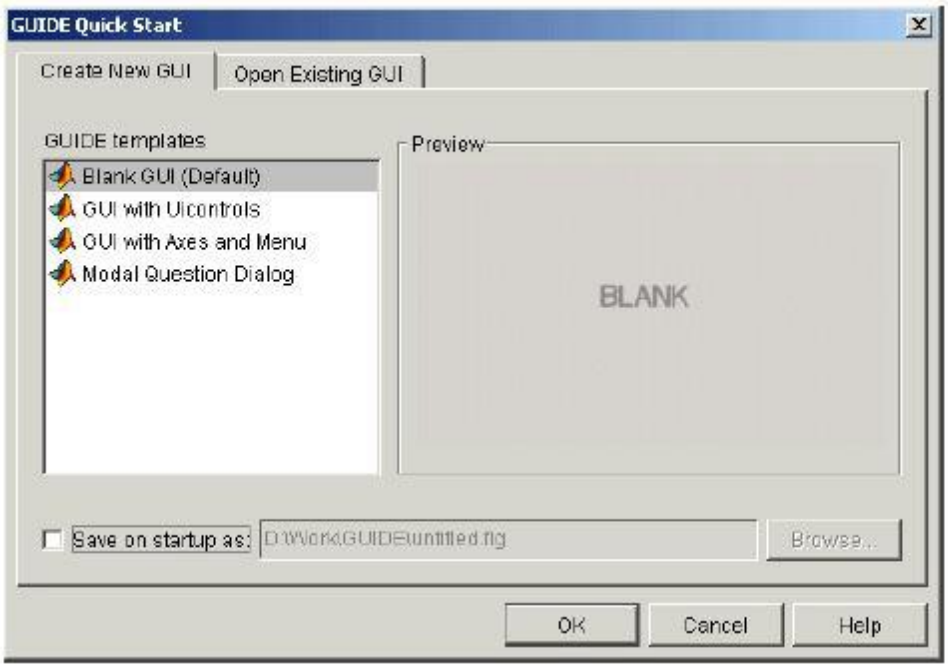
1.- Puede seleccionar botones, menús desplegables, graficas, etc. y transferirlos a la hoja de trabajo en una posición determinada. Haciendo doble clic sobre algún elemento y luego, con el cursor cambiado a una X, uno se posiciona en la hoja de trabajo y selecciona el tamaño que quiere que ocupe la herramienta seleccionada dentro de la hoja de trabajo.

Cuando uno selecciona una herramienta, con doble clic se pueden seleccionar todos los detalles:

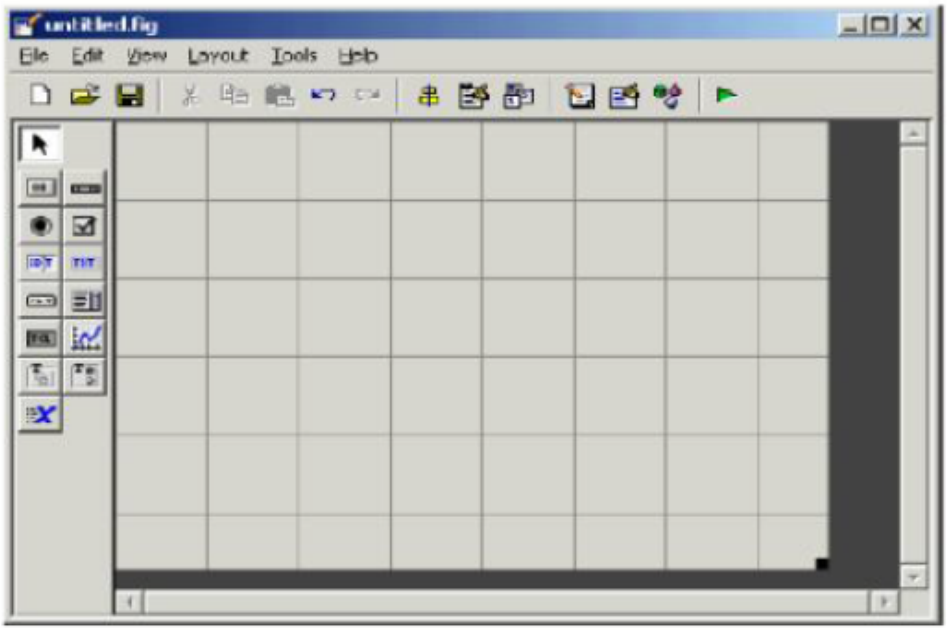
Por ejemplo el color de fondo, los nombres con los cuales se les relacionará en el archivo .m, etc.

2.- A diferencia de la ejecución de funciones o scripts de MATLAB tradicionales, la ejecución de GUIs viene dada por la secuencia de eventos y las acciones asociadas a cada uno de ellos. El script no finaliza a menos que el usuario genere un botón que ejecute tal acción. El desarrollo de GUIs se realiza en dos etapas:

a) Selección de las herramientas (controles, menús y ejes) que conformarán el GUI.
b) Codificación de la respuesta de cada uno de los componentes ante la interacción del usuario.
Lo primero que se debe hacer es escribir *guide* en el *command window*. Aparecerá la siguiente ventana. Se selecciona una hoja en blanco (Blank GUI)



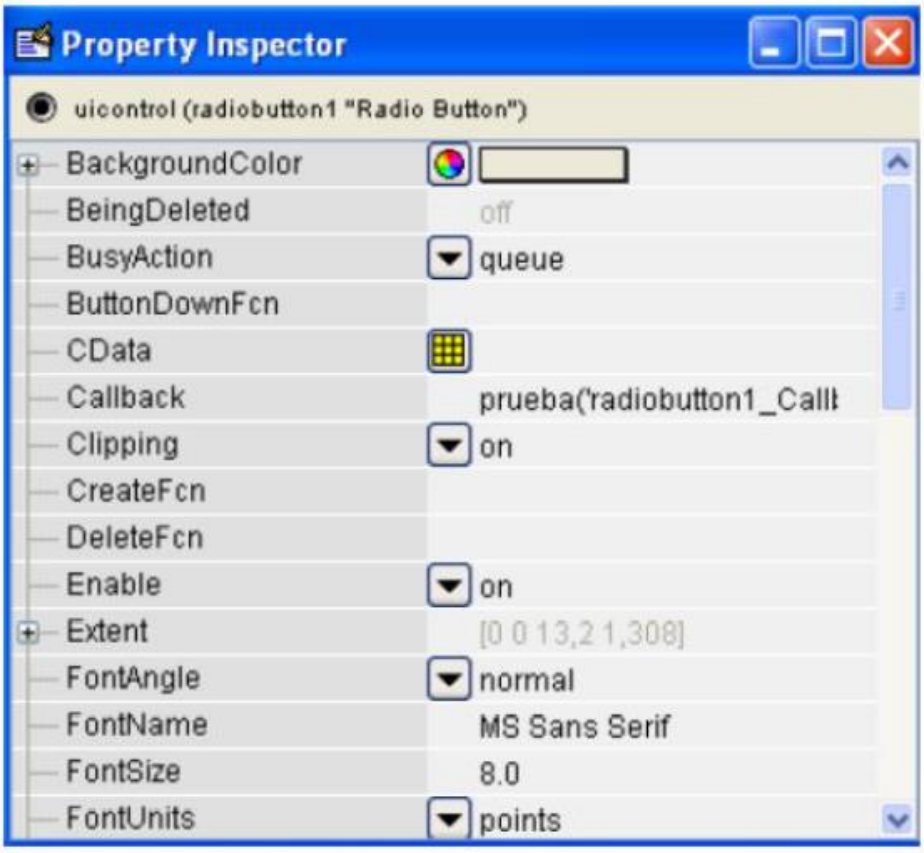
Aparecerá de inmediato una hoja de trabajo como la que sigue:



A la izquierda están las herramientas disponibles. Si en este momento se salva el GUI debemos seleccionar un nombre. Se almacenará en el directorio donde estemos ubicados **nombre.fig** y **nombre.m** .

Para utilizar una herramienta se selecciona y arrastra a la posición deseada en la hoja de trabajo; se puede modificar su tamaño tal y como se hace con cualquier figura. Otras características se pueden modificar haciendo doble click. Aparecerá un editor de propiedades como el que sigue:

Al hacer una modificación en el editor de propiedades también cambiará el código relacionado a cada botón en el archivo **.m**



3.- El archivo **nombre.m** tiene toda una estructura de handles (manejadores) que alimentarán a la GUI. La estructura de handles es pasada como una entrada a cada callback (llamada a una parte de un programa).

Puedes usar la estructura de handles para:
Compartir datos entre callbacks
Acceder la data en el GUI
Para almacenar los datos contenidos en una variable X, se fija un campo de la estructura de handles
igual a X y se salva la estructura de handles con guidata como se muestra a continuación:
handles.current_data=X
guidata(hObject, handles)
En cualquier momento puedes recuperar la data haciendo
X=handles.current_data

Código asociado a cada elemento del GUI

Automáticamente al crear el GUI y salvarlo aparece en el archivo .m una cantidad de líneas de código fijas. En la primera parte del script aparece una cantidad de líneas de código fijo. La primera instrucción :

function varargout = untitled(varargin) indica que se está creando un GUI de nombre untitled con argumentos de salida varargout y argumentos de entrada varagin. Solo se muestran las dos primeras y la última línea.

```
function varargout = untitled(varargin)
% UNTITLED M-file for untitled.fig
……..
% Last Modified by GUIDE v2.5 19-Oct-2005 16:25:55
```

Aquí comienza un código de inicialización que se pide no se edite. También se muestra solo la primera y última línea

```
% Begin initialization code - DO NOT EDIT
```

4.-

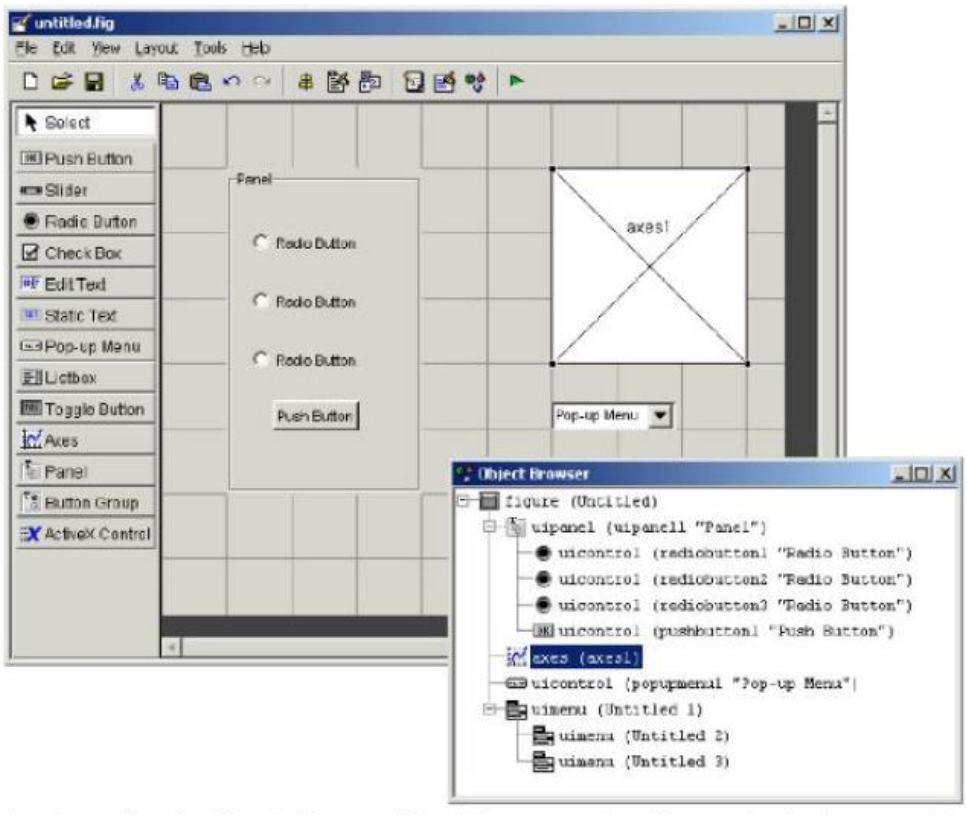
```
……
% End initialization code - DO NOT EDIT
```

Hasta aquí llega el código de inicialización. Comienza entonces lo que queremos que ocurra antes de que el GUI se haga visible. Estas instrucciones se colocan después de la línea donde dice
function untitled_.....

```
% --- Executes just before untitled is made visible.
function untitled_OpeningFcn(hObject, eventdata, handles, varargin)
…..
……
varargout{1} = handles.output;
```

Luego de esto aparecen los callback's dependiendo de las herramientas que se han incorporado al GUI.

Por ejemplo:



Aquí se han colocado 3 Radio Button, 1 Push Button, un Pop-Up menú y 1 eje para graficar.
A continuación describiremos brevemente las herramientas disponibles, el código automático asociado a cada una de ellas y como interactuar con las mismas.

boton pushbutton: Se ejecuta una determinada acción cuando son presionados. En el archivo **.m** aparecen automáticamente un grupo de instrucciones asociadas a él.

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)


5.- AQUI SE COLOCA EL CODIGO QUE SE ESPERA EJECUTAR CUANDO SE
PRESIONES EL PUSHBUTTON1
% hObject handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

listas de selección: Aquí se puede colocar una lista de elementos para que el usuario pueda seleccionar alguno. En el archivo **.m** aparecen automáticamente un grupo de instrucciones asociadas a él.

```
% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
……
% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
……..
% Hints: contents = get(hObject,'String') returns listbox1 contents as cell array
% contents{get(hObject,'Value')} returns selected item from listbox1
```

Como se observa en la ayuda (Hint) que aparece en las dos últimas líneas, si se coloca la Instrucción Contents= get(hObject,'String')
Se obtendrá un número que indica que selección se hizo. Por ejemplo si se seleccionó el tercer elemento de la lista, Contents dará 3

Boton edit

Permite a los usuarios ingresar o modificar parámetros que se quieren introducir.

```
function edit1_Callback(hObject, eventdata, handles)
…..
% Hints: get(hObject,'String') returns contents of edit1 as text
% str2double(get(hObject,'String')) returns contents of edit1 as a double
```

Observe la ayuda (HINT)
SI SE COLOCA LA INSTRUCCION
A= str2double(get(hObject,'String'))
Entonces, se podrá tener en A el valor del número que uno escribió en la casilla
SI SE COLOCA LA INSTRUCCION
A= get(hObject,'String')
Entonces, se podrá tener en A los caracteres que uno escribió

6.- Botón RadioButton

Son botones de selección. Si hay varios generalmente son mutuamente excluyentes. Para Seleccionarlo basta ubicarse y presionar el ratón.

```
% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% Hint: get(hObject,'Value') returns toggle state of radiobutton1
```

SI SE COLOCA LA INSTRUCCION
a=get(hObject,'Value')
Entonces en a se tendrá 1 si el botón fue seleccionado

Ejes para graficar:

Aquí no se genera nada especial pero uno debe fijar las condiciones de la gráfica y activarla o desactivarla según convenga.
Por ejemplo si la gráfica tiene asociado el nombre de axes1 y se quiere deshabilitar

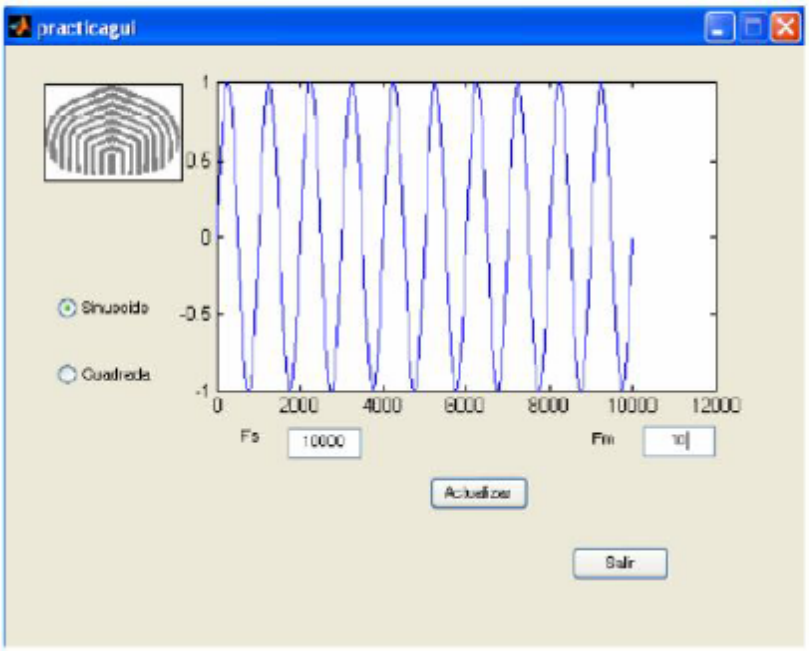
```
set(handles.axes1,'Visible','off');
```

PopUp menú. Cuando se hace click despliega opciones. Para agregar elementos a la lista, en el editor de propiedades se busca el elemento string y allí se coloca la lista de las opciones.

```
% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
…….
% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
……..
% Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell array
% contents{get(hObject,'Value')} returns selected item from popupmenu1
```

SI SE COLOCA LA INSTRUCCION
contents=get(hObject,'Value')
ENTONCES SE OBTENDRA UN NUMERO QUE INDICA QUE SELECCIÓN SE HIZO.
POR EJEMPLO SI SE SELECCIONÓ EL TERCER ELEMENTO DE LA LISTA, contents DARA 3

7.-
Experimento



Ud. Debe diseñar una GUI con las siguientes características:
• Debe presentar del lado izquierdo el sello y nombre del CETI
• Permitirá dibujar 1 de dos señales (una sinusoide y una cuadrada)
• La selección del tipo de señal se hará con radiobutton.
• El usuario debe fijar el valor de Fs y Fseñal y utilizar un botón para actualizarlos
• En cualquier momento se puede cambiar el tipo de señal.

• Debe tener un botón para salir del programa que cierra todas las ventanas
Para esto desde el comand window escriba *guide*. Debe escoger Blank Gui o sea una hoja de trabajo en blanco con las herramientas del lado izquierdo. Seleccione el icono de gráficas y arrástrelo hacia la hoja de trabajo. Este se llamará axes1. Arrastre otro; este se llamará axes2. En estos lugares Ud. colocará el sello de la UABC (axes2) y la gráfica (sinusoide o cuadrada) (axes1); dándole doble click sobre esta puede ver sus propiedades; busque la variable Tag y cambie el nombre de axes2 a Imagen.
Guarde el archivo con el nombre practicaGui. Si Ud. quiere deshabilitar inicialmente axes 1 debe escribir (luego del grupo de instrucciones que se ejecutan antes de que el GUI sea visible)

```
set(handles.axes1,'Visible','off');
```

Esto "apaga" la rejilla donde se graficará la señal seleccionada
La instrucción SET tiene la siguiente estructura:
SET(H,'PropertyName',PropertyValue)
Luego se habilitará Imagen que es donde aparecerá el sello del CETI.

```
axes(handles.Imagen);
image(imread('usb','jpg'));
```

8.-

Además eliminará las escalas en X y en Y

```
set(handles.Imagen,'Xtick',[]);
set(handles.Imagen,'Ytick',[]);
```

Ahora arrastre un radiobutton que se llamará radiobutton1. Actívelo agregando, en la zona de código que se ejecuta antes de que la interfaz sea visible, la siguiente instrucción:

```
set(handles.radiobutton1,'Value',1);
```

Ahora arrastre otro radiobutton que se llamará radiobutton2. Inicialmente desactívelo:

```
set(handles.radiobutton2,'Value',0);
```

En este momento aparecerán dos callback cada uno asociado a cada radiobutton. En el callback de radiobutton1 agregue las siguientes instrucciones:

```
set(handles.radiobutton1,'Value',1);
set(handles.radiobutton2,'Value',0);
```

En el callback de radiobutton2 coloque, en cambio:

```
set(handles.radiobutton1,'Value',0);
set(handles.radiobutton2,'Value',1);
```

Esto los hará excluyentes.

Si en este momento ejecuta el archivo **.m** le aparecerán: el sello de la UABC y los dos Radiobutton estando activado el 1 (Sinusoide). Ahora le pondrá nombres al lado de los radio button. Para eso hará doble clic sobre el radio button y aparecerá el Inspector de Propiedades (Property Inspector);
Busque String y allí pondrá el nombre deseado (**Sinusoide**). Haga lo mismo con el otro y póngale **Cuadrada.**
Agregue ahora un pushbutton: Su tag será pushbutton1 y al String pongale **"Salir".** En donde está el código correpondiente al pushbutton escriba la instrucción "close all".

Cuando se presione el botón Salir se cerrarán todas las ventanas incluyendo el **.fig** que se está ejecutando

```
% --- Executes on button press in salir.
function salir_Callback(hObject, eventdata, handles)
% hObject handle to salir (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
close all
```

Ahora generará el vector de tiempo y los dos mensajes de la siguiente manera (coloque fs y fm a su gusto, por ejemplo10000 y 10). Este código puede colocarlo inmediatamente después de donde el comentario dice: % --- Executes just before NOMBRE DE SU PROGRAMA is made visible.

9

```
t=[0:1/fs:1];
mensaje1=sin(2*pi*fm*t);
mensaje2 = square(2*pi*fm*t);
handles.mensaje1 = mensaje1;
handles.mensaje2 = mensaje2;
Se habilita el sitio donde se graficará la función
set(handles.axes2,'Visible', 'on');
axes(handles.axes2)
```

AQUI SE SELECCIONA SI SE TRABAJARA CON LA SINUSOIDE O CON LA CUADRADA DE ACUERDO AL BOTON QUE ESTE SELECCIONADO Y SE GRAFICA

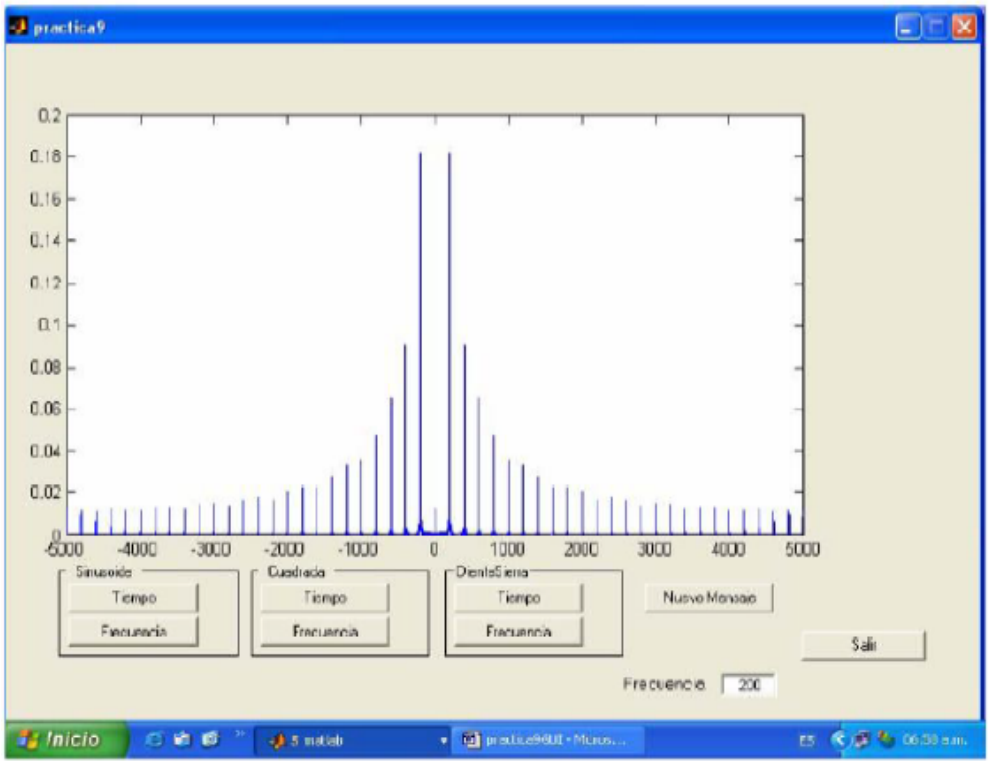
```
switch get(handles.radiobutton1,'Value')
case 1
mensaje = handles.mensaje1;
case 0
mensaje = handles.mensaje2;
end
plot(mensaje)
```

Ahora ejecute y vea lo que ocurre. Arregle el código para que la sinusoide aparezca desde el comienzo.

Añada ahora la posibilidad de agregarle a la GUI dos ventanas que permitan fijar la frecuencia de muestreo y la frecuencia de las sinusoides y la cuadrada.

10

Ejercicio



• Ud. Debe diseñar una GUI similar a la siguiente:
• Esta interfaz le permitirá presentar 3 señales distintas o en tiempo o en frecuencia. La frecuencia(fm) de la Sinusoide, la fundamental de la Cuadrada y la fundamental de la Diente de Sierra podrá ser fijada por el usuario; cuando se cambie la frecuencia se presionará el botón Nuevo Mensaje para que esta sea incorporada como dato; luego se puede visualizar con la frecuencia cambiada. Se debe disponer de un botón de Salir que cierre todas las ventanas.
• Antes de que aparezca la primera señal debe aparecer el sello de la CETI al lado izquierdo.
Cuando se grafique la primera señal este desaparecerá.
• Luego de que presione cualquiera de los botones que indican Tiempo o Frecuencia aparecerá la primera señal o el primer espectro graficado.
• En cualquier momento Ud. podrá cambiar fm, Presionar nuevo mensaje y estará listo para Poder ver otra Sinusoide, Cuadrada o Diente de Sierra y sus espectros.
• Una vez que tenga operativa esta GUI, debe agregarle nuevas ventanas que fijen la frecuencia de muestreo y el número de puntos que Ud. desea trabajar. (fs y N).
• Posteriormente piense en un diseño diferente en el cual las señales (sinusoidal, cuadrada y Diente de sierra) se puedan seleccionar por medio de un menú del tipo listbox o pop-up menú, mientras que el dominio de presentación sea elegido con un radio button.