Tema 5. Representación de datos. Interfaces gráficas en MATLAB

María del Mar Sanz Lluch

Borja Bordel Sánchez

Marina Pérez Jiménez

MATLAB aplicado a la instrumentación electrónica

Departamento de Electrónica Física (UPM)

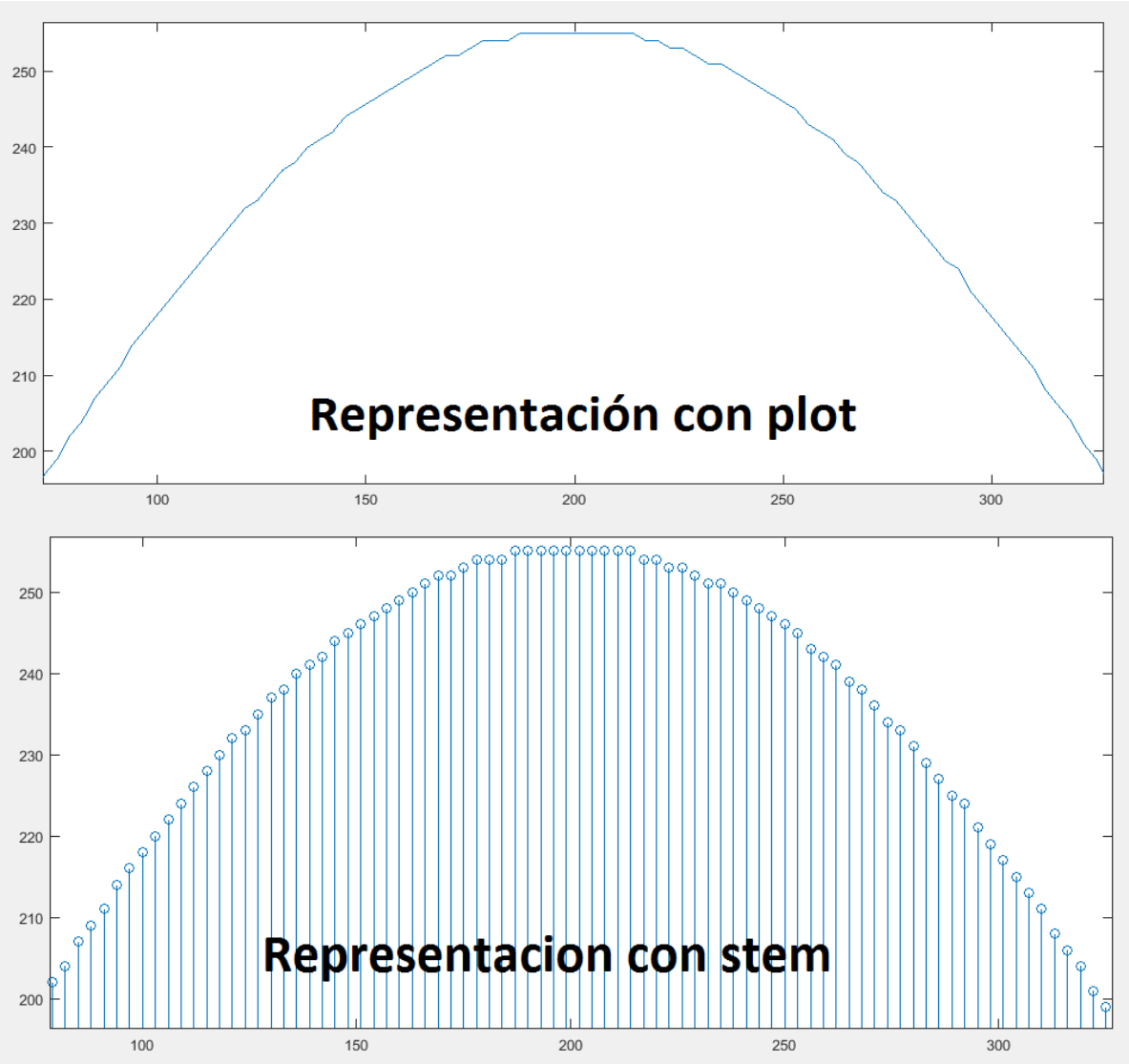
Contenido del tema:

- Funciones básicas para la representación de resultados.
- La GUI de MATLAB
- Funciones principales para el intercambio de datos.
- Ejemplo. Programa para la generación de señales que se enviarán mediante puerto serie.

Funciones básicas para la representación de resultados (1)

- La forma principal de representación de datos en MATLAB es la representación gráfica. Mediante dos funciones principales:
  - Plot: representa los resultados de forma contínua.
  - Stem: representa los datos de forma discreta.

Funciones básicas para la representación de resultados (2)



Funciones básicas para la representación de resultados (3)

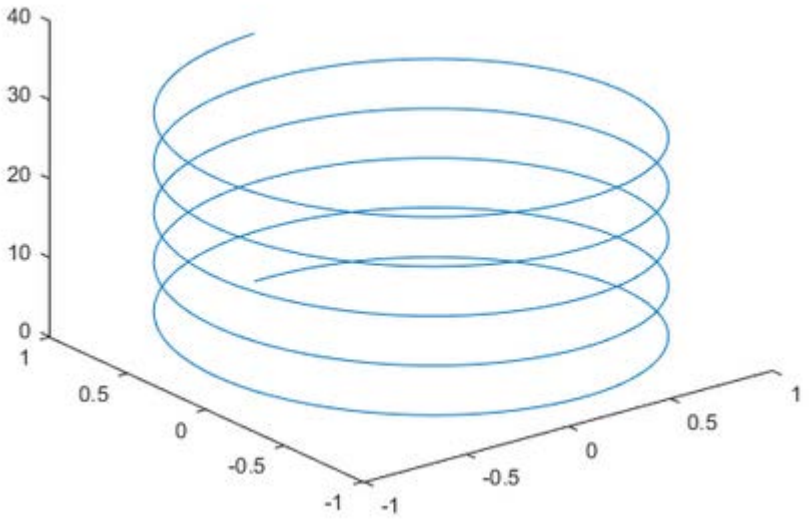
- La función ***plot*** representa una serie de datos de forma continua uniendo las muestras con líneas rectas para que el usuario tenga sensación de continuidad. Hay que tener en cuenta que es sólo una representación y que lo que se muestra no es contínuo, teniendo en cuenta que la forma de tomar los datos en MATLAB (al ser un programa infórmatico) es siempre discreta.
- La función plot tiene varios argumentos o modificadores que son muy interesantes, pero debido a las limitaciones del curso vamos a hablar sólo de los más importantes.
- La forma estándar de escribir la función es: plot (argumento 1, argumento 2, argumento 3) Cada uno de ellos son:
  - Argumento 1: Eje X de la representación
  - Argumento 2: Eje Y de la representación. Si no se especifica uno de los dos argumentos. Se representaran las muestras con un vector generado automáticamente y usado como eje X.
  - Argumento 3: color de la línea especificado entre comillas simples. Por ejemplo: 'r' es rojo, 'k' es negro…
  - Otros modificadores: otro argumento que puede ser interesante es el que sirve para especificar el grosor de la línea de representación. Para ello escribieremos *'LineWidth'* seguido de un valor numérico que corresponderá al grosor de la línea.

Funciones básicas para la representación de resultados (4)

- La función ***stem*** se utiliza para la representación de datos discretos como muestras sin continuidad entre ellas.
- Al igual que sucedía con la función plot, vista anteriormente, la función stem se "llama" con una serie de modificadores que tendrán que ver con lo que se quiere representar y con como se quiere hacer.
- Los modificadores o argumentos más importantes que vamos a estudiar son los siguientes:
  - Argumento 1: Eje X de la representación
  - Argumento 2: Eje Y de la representación. Si no se especifica uno de los dos argumentos. Se representaran las muestras con un vector generado automáticamente y usado como eje X.
  - Argumento 3: color de la muestra especificado entre comillas simples. Por ejemplo: 'r' es rojo, 'k' es negro…
  - Otros modificadores: otro argumento que puede ser interesante es el que sirve para especificar el grosor de las muestras representadas. Para ello escribiremos *'LineWidth'* seguido de un valor numérico que corresponderá al grosor.

Funciones básicas para la representación de resultados (5)

- Aunque existen muchas más funciones para la representación de resultados, las más interesantes desde el punto de vista de la instrumentación electrónica son las dos que se han visto.
- Algunas otras funciones como ***plot3*** permiten la representación de elementos definidos en 3 dimensiones. Es interesante para los casos en los que la variables que se quieren representar tienen relación entre ellas.

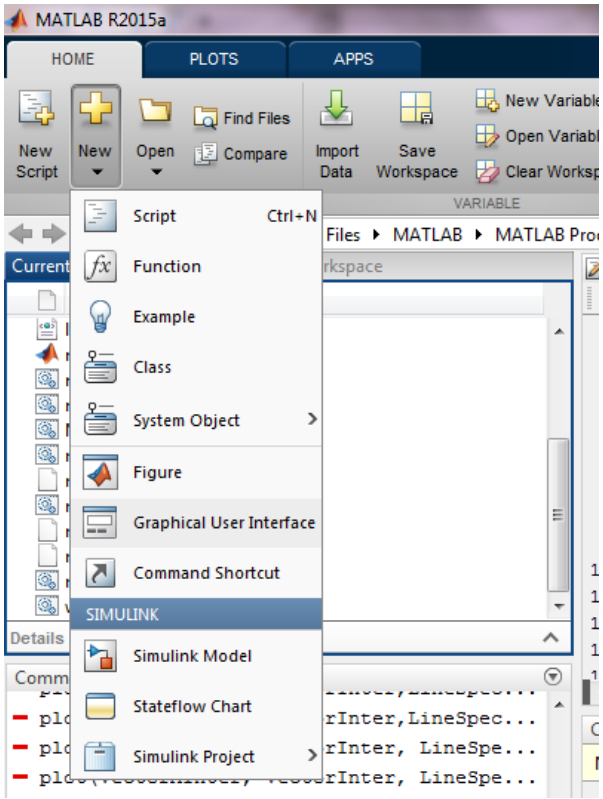


La GUI de MATLAB(1)

- El desarrollo de entornos gráficos para el manejo del hardware de instrumentación, es tan importante como la instrumentación en sí. Actualmente existen muchas plataformas informáticas que permiten el desarrollo de aplicaciones gráficas para tal efecto, pero en este curso nos vamos a centrar en el manejo de las funciones principales asociadas a la GUI de MATLAB que es el editor de interfaces propio.
- GUI son las siglas de “Graphical User Interface” y es un editor de interfaces gráficas que permite, de manera muy simple, el enlace entre el panel de la interfaz con las funciones asociadas a elementos del mismo.
- La forma de interactuar con las funciones de MATLAB que subyacen a cada uno de los elementos gráficos se hace mediante la escritura de dichas funciones en el código asociado a los elementos. Por ejemplo, podemos pulsar un botón que invoca a una función concreta porque dicha función se encuentra escrita en el código del botón.
- A continuación vamos a estudiar los pasos principales para desarrollar una interfaz gráfica.

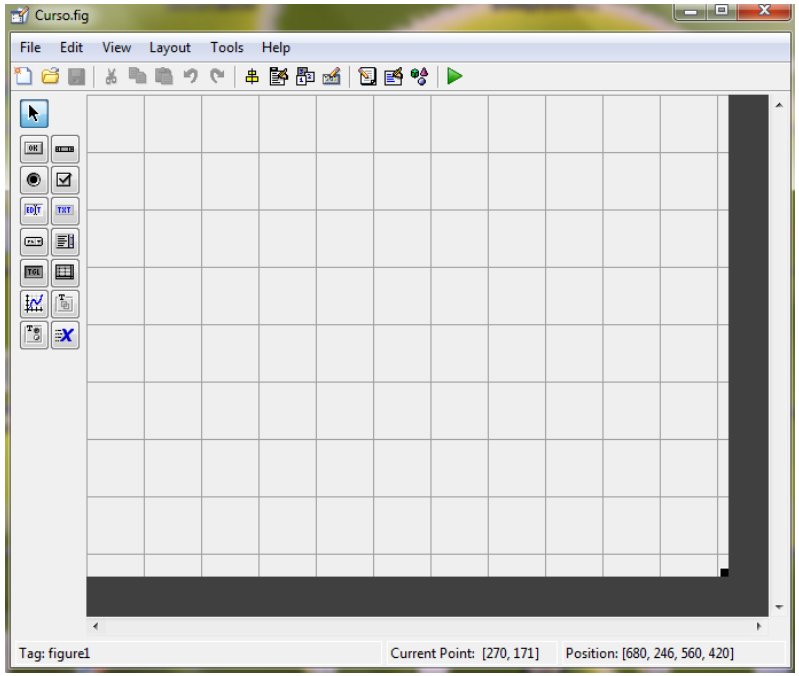
La GUI de MATLAB(2)

- La forma de empezar a escribir una interfaz mediante la GUI es incluyendo un nuevo archivo GUI desde el panel principal de MATLAB:
- Una vez que se ha generado el GUI, veremos en el directorio de trabajo dos nuevos archivos, uno será un ".m" y otro un ".fig". El primero corresponderá al codigo de la GUI mientras que el segundo archivo mencionado corresponde a la propia interfaz gráfica.
- Cualquier cosa que se haga en la interfaz tendrá repercusión en el archivo del código pero al revés, no siempre es así.



La GUI de MATLAB(3)

- En su forma básica la GUI es un lienzo (**canvas**) en color gris donde se pueden incluir elementos gráficos que permiten la interactuación con un código subyacente que es en sí la interfaz.



La GUI de MATLAB(4)

- Los elementos gráficos para el desarrollo de la aplicación se sitúan en una paleta a la izquierda del canvas. Los que vamos a tratar de una manera especial son:

Panel: Todos los elementos que se dejen dentro de un panel, se podrán mover por el canvas como si fueran uno solo

Botón: Este elemento sirve para invocar un código a ejecutar por parte de la interfaz gráfica. Se puede llamar dicho botón como se quiera y modificar determinados aspectos como el estado inicial (pulsado o no pulsado), el color…

Edit text: Este elemento sirve para introducir elementos en forma de cadena de caracteres y que serán usados como valores de variables, por parte de la interfaz.

Texto fijo: Este elemento se va a emplear para escribir texto que se genere en el código de la interfaz.

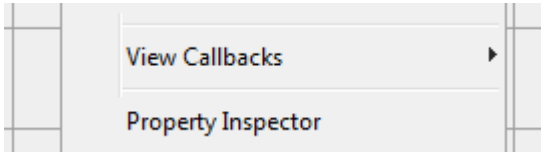
Grafico: Este elemento representa los datos que se le envíen como parámetro en ejes, empleando las funciones de representación que hemos estudiado anteriormente.

Botón radial: Actúa como selector. Esto es, plantea el valor de una variable (botón) como dos estados posibles, activado o desactivado. La diferencia con un botón normal es que el botón normal vuelve a su posición de inicio nada más dejar de pulsarlo.

La GUI de MATLAB(5)

- Una vez que se ha posicionado un elemento gráfico en el canvas, se pueden acceder a dos partes importantes asociadas a él. Por un lado las propiedades del elemento que serán propias de cada uno de ellos y de las cuales, por mencionar alguna de las más importantes:
  - Nombre: con el que se llama a dicho elemento. Es el nombre asociado a dicho objeto.
  - Color: hace referencia al color de fondo del objeto grafico.
  - Texto: hace referencia al texto que se puede leer en el objeto.
  - Características del texto: tamaño, tipo de texto, modificadores al texto (negrita, cursiva…)

- Por otro lado, se puede acceder a la función propia del objeto. Esta función será la que se ejecute cuando se interactúe (de manera específica para cada objeto) con el elemento gráfico que se ha colocado en el canvas.
- A ambos campos (propiedades y función) se puede acceder presionando el botón derecho del ratón encima del objeto.



La GUI de MATLAB(6)

- La función que subyace a la interfaz gráfica permite ejecutar el código cada vez que se interactúa de forma correcta con los elementos gráficos que se han incluido.
- La interfaz tiene una parte de código asignada a la inicialización y al cierre de la aplicación. En estas partes del código se pueden incluir rutinas que se ejecuten cuando iniciamos o cerramos la aplicación a parte de las que ya incluye MATLAB per sé.
- Una parte interesante que merece la pena mencionar, como parte de la rutina de inicialización es la inclusión de las variables globales a toda la aplicación.

```
% --- Executes just before ControlTX is made visible.
function ControlTX_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ControlTX (see VARARGIN)

% Choose default command line output for ControlTX
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ControlTX wait for user response (see UIRESUME)
% uiwait(handles.figure1);

global serialPorts
global dato1
global dato2
global dato3
```

La GUI de MATLAB(7)

- El valor de estas variables globales es accesible desde todas las partes del código, pero en cada función asociada a un objeto deben estar declaradas, también como variables globales.
- Obviamente, las variables propias sólo de una de las función estarán declaradas sólo en esa función, es decir, en el código propio de cada función.
- Las variables, tanto locales como globales, pueden obtenerse de algunos elementos gráficos que se incluyen en el canvas de la interfaz. Por ejemplo, de los objetos ***"edit text",*** se pueden obtener cadenas de caracteres que más tarde pueden convertirse en números con los que realizar funciones matemáticas.
- A continuación, vamos a estudiar alguna de las funciones básicas más importantes para el intercambio de datos entre el canvas de la interfaz y el código que ejecutará las funciones pertinentes.

Funciones principales para el intercambio de datos (1)

- Lo realmente interesante de la GUI de MATLAB es la posibilidad de que el usuario pueda interactuar directamente con el instrumento que esta manejando o del que está obteniendo información.
- La forma más sencilla de introducir datos a través de la interfaz es con el objeto ***"edit text"*** Este objeto recoge la cadena de caracteres introducidad a través del elemento gráfico y lo almacena en una variable que podrá ser modificada como se quiera.
- La forma de acceder a la información contenida en el objeto es accediendo al campo "String" del objeto:
  - *ID=get(handles.edit1,'String');*
- En el ejemplo anterior, se accede a este campo y se almacena el contenido en una variable llamada ID. El objeto en cuestión es un ***"edit text"*** llamado edit1.
- Si se incluye dicha variable asociada, por ejemplo a la pulsación de un botón, el usuario puede introducir los datos que quiera y cargarlos para ser empleados en otra función. Más adelante se verá un ejemplo de aplicación como ejemplo de esto que acabamos de ver

Funciones principales para el intercambio de datos (2)

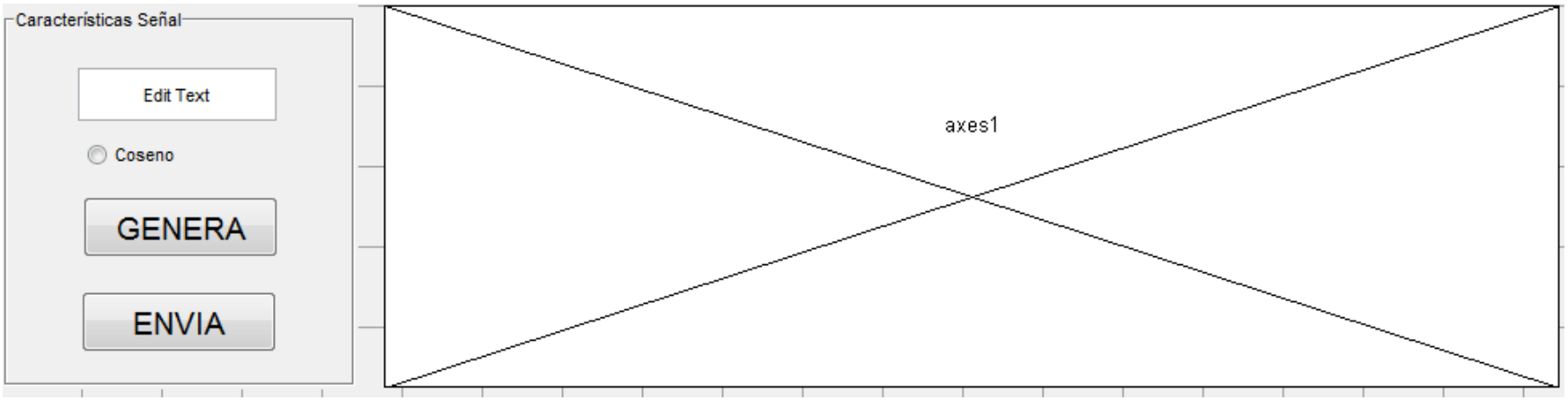
- El objeto ***"text"*** permite la visualización de datos a través de un bloque de texto donde pueden aparecer los datos que se quieran, es una de las funciones principales para la visualización de datos de los que dispone la GUI de MATLAB.
  - set(handles.text1,'String',dato);
- La orden anterior cargaría en un objeto ***"text"*** el contenido de la variable datos. Dicha variable tiene que estar declarada como cadena de caracteres para poder visualizarse.
- Las funciones asociadas a los ***"botones radiales"*** permiten al usuario programar selectores, de tal forma que, en función del valor que tengan dichos botones se puede realizar una función u otra.
  - contr1=get(handles.radiobutton1,'Value');
- La orden anterior obtiene el valor del campo "Value" del objeto y lo almacena en una variable que será booleana (debido a sus dos posibles valores que puede tomar)

Ejemplo. Programa para la generación de señales que se enviarán mediante puerto serie(1)

- Vistas las funciones más importantes, con las que se puede programar aplicaciones sencillas, vamos a desarrollar un ejemplo básico de manejo de la GUI de MATLAB, teniendo siempre, muy presente, que existen innumerables opciones y características que se escapan del contenido de este curso, principalmente por la corta extensión del mismo.
- La aplicación que va a realizarse genera una señal senoidal y el usuario podrá elegir entre un seno o un coseno en función del estado de un botón radial.
- Más tarde se enviará vía serie.
- La función podrá verse gracias a un gráfico de ejes.

Ejemplo. Programa para la generación de señales que se enviarán mediante puerto serie(2)

- Primero vamos a dibujar en el canvas. Se van a incluir dos botones, un botón radial y un gráfico de ejes, además de un objeto “edit text” para incluir la duración de la señal:



- Todos los elementos de edición se han introducido en un panel con el objetivo de tenerlos más organizados.

Ejemplo. Programa para la generación de señales que se enviarán mediante puerto serie(3)

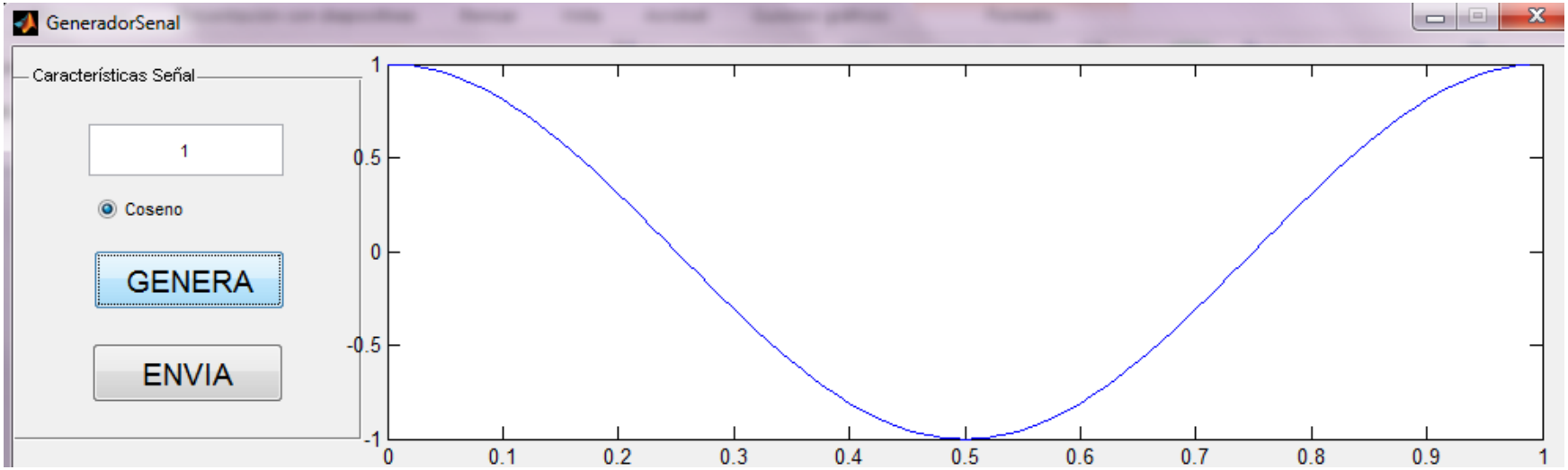
- Primero, vamos a escribir la función que genera el seno. Para ello introduciremos una duración temporal a través del “edit text” y seleccionaremos o no la opción de “coseno”.

```
function pushbutton1_Callback(hObject, eventdata, handles)
% 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)

N=100
T=str2double(get(handles.edit1,'String'));
t=[0:T/N:T-(T/N)];
coseno=get(handles.radiobutton1,'Value');

if(coseno==1)
    senal=cos(2*pi.*t);
else
    senal=sin(2*pi.*t);
end
set(handles.axes1)
plot(t,senal)
```

Ejemplo. Programa para la generación de señales que se enviarán mediante puerto serie(4)

- Establecemos un vector de muestras con un total de 100 que vaya desde la muestra 0 hasta la muestra 100 llegando (en tiempo) a la duración marcada e introducida mediante “edit text”.
- Después se consulta el estado del botón radial que actúa como selector de seno o coseno. Como el resultado de esa consulta es un valor booleano se puede incluir como discriminador en una sentencia “if”, de tal forma que si el botón radial tiene como resultado ‘1’ (se ha seleccionado coseno) la función que se genera es un coseno.
- El último paso, una vez que se ha generado la función es mostrarla a través del gráfico de ejes que se ha incluido en la interfaz. Para ello se seleccionan como gráficos activos el objeto “axes1” y se invoca un plot para la señal.

Ejemplo. Programa para la generación de señales que se enviarán mediante puerto serie(5)



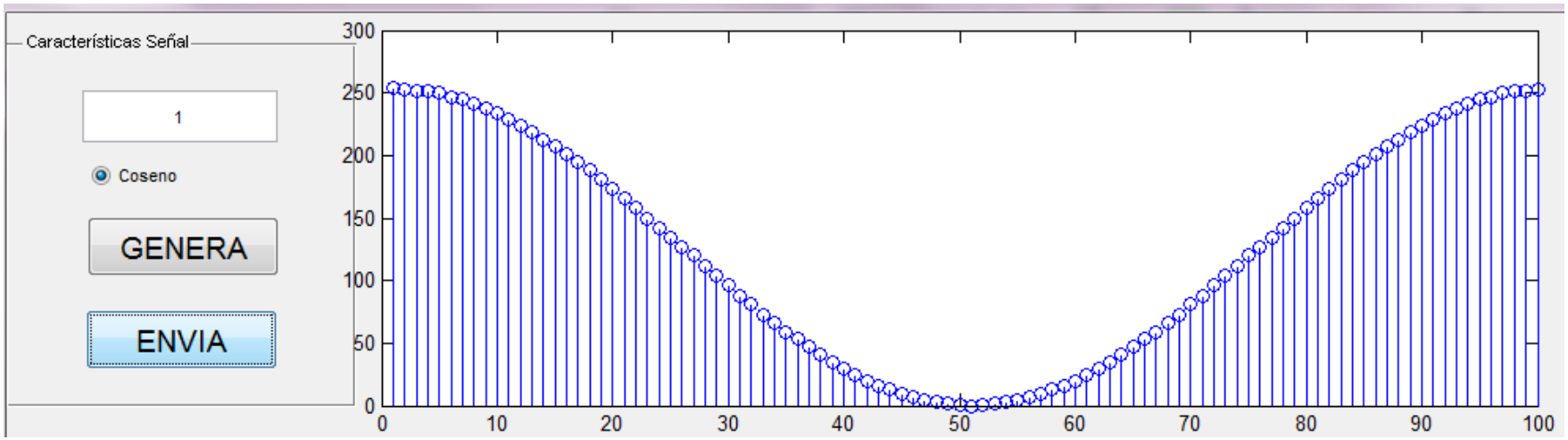
Ejemplo. Programa para la generación de señales que se enviarán mediante puerto serie(6)

- Vamos a incluir ahora una rutina para el envío a través de puerto serie. Como nuestro objetivo es cargar la señal generada en una pila de datos leibles por un microcontrolador, transformaremos los datos en un array de números comprendidos entre 0 y 255, donde 255 corresponderá al valor máximo que tenga nuestra seña, es decir 1
- Para ello usaremos el botón “ENVÍA” que previamente hemos incluido en el canvas de la interfaz.
- La primera parte de la función genera los datos adaptados al micro:

```
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global senal

N=100;
vector=[1:N];
muestras = fix(senal.*127)+127;

set(handles.axes1)
stem(vector,muestras);
```

Ejemplo. Programa para la generación de señales que se enviarán mediante puerto serie(7)

- Se ha incluido la variable “senal” como global debido a que se genera en otra parte del código y se usa invocada por la función asociada al botón “ENVÍA”
- La nueva matriz de datos se representa de nuevo en los ejes pero esta vez como muestras discretas, para saber los datos que se van a enviar mediante el puerto serie.



Ejemplo. Programa para la generación de señales que se enviarán mediante puerto serie(8)

- Los datos que deben mandarse a través del puerto serie están almacenados en el vector “muestras”.
- Ahora, para mandar los datos a través del puerto serie, lo único que tendremos que hacer es escribir la rutina de envío. Para ello usaremos las funciones ***“fopen”*** y ***“fwrite”.***
- Primero, abriremos el puerto serie. Para eso ejecutaremos la rutina siguiente:

```
serialPorts=[serial('COM1')];
fopen(serialPorts(1));
```

- Una vez que se haya ejecutado estas órdenes, ya estará abierto el puerto serie y podrá usar para enviar datos. Nótese que el nombre del puerto es ‘COM1’ por defecto, pero será importante hacer una preselección de los puertos que se van a usar, empleando el administrador de dispositivos de Windows.

Ejemplo. Programa para la generación de señales que se enviarán mediante puerto serie(9)

- Ahora procedemos a enviar los datos. Para ello usamos la función “fwrite”. Se va a enviar dato por dato lo que nos obliga a implementar un bucle “for” que vaya recorriendo el array de muestras.

```
for i=1:length(muestras)
    fwrite(serialPorts(1),muestras(i));
    pause(0.5);
end
```

- En este bucle, se recorre el array de muestras desde el principio hasta el final y se envía el elemento actual del bucle, es decir “muestras(i)”.
- Por cada envío se espera medio segundo. No es del todo necesario realizar esto pero puede ser importante si el micro que actúa como receptor de órdenes tiene algún tipo de rutina o código que ejecutar cuando recibe un dato, de esta forma, se le concede un tiempo de procesado.
- La ejecución de este código puede repetirse todas las veces que se quiera pero para evitar posibles errores conviene incluir la rutina de apertura del puerto sujeta a un objeto gráfico distinto, para que no se abra siempre que se pulsa “ENVÍA”