CAPÍTULO 3. INTERFACES GRÁFICAS DE USUARIO

3.1 PRESENTACIÓN DEL GUIDE

El **GUIDE** (*Graphical User Interface Development Environment*: Entorno de desarrollo de Interfaz Gráfico de Usuario) es un conjunto de herramientas que proporciona MATLAB para la creación de **GUI**s (*Graphical User Interface*: Interfaz Gráfico de Usuario). Un GUI es un conjunto de uno o varios paneles que están provistos de algunos de los siguientes elementos:

- ✓ Controles (botones, menús desplegables, etc.), que permitirán al usuario interactuar con el GUI y establecer el flujo de ejecución.
- ✓ Menús.
- ✓ Ejes de coordenadas, que permiten dibujar gráficos e imágenes.

Vamos a usar el GUIDE para:

- ✓ Diseñar el aspecto del GUI.
- ✓ Programar el GUI: El GUIDE genera de forma automática un fichero .m que controla el funcionamiento del GUI. Este fichero .m inicializa el GUI y contiene un marco para todos los *callbacks* del GUI (las órdenes que se ejecutan cuando el usuario interactúa con un elemento del GUI). Usando el editor de MATLAB podremos añadir código a los *callbacks* para realizar las funciones que queramos asignarles.

El beneficio que proporciona el uso de GUIs es evidente, ya que permiten al usuario ejecutar cómodamente código desarrollado en MATLAB sin necesidad de cumplir la incómoda sintaxis funcional necesaria cuando se trabaja desde la línea de órdenes. A diferencia de la ejecución de funciones o *scripts* de MATLAB, la ejecución de GUIs no predetermina el flujo de ejecución del código. Es el usuario, a través de su interacción con el GUI, el que determina el orden en que se ejecutan las diferentes órdenes y funciones desarrolladas. Otra diferencia importante es que la ejecución no termina cuando finaliza la ejecución del *script* o función, sino que el GUI permanece abierto, permitiendo al usuario invocar la ejecución de ese u otro código desarrollado.

El desarrollo de GUIs se realiza en dos etapas:

- o Diseño de los componentes (controles, menús y ejes) que formarán el GUI.

- Codificación de la respuesta de cada uno de los componentes ante la interacción del usuario.

Para empezar, en la Figura 16 se muestra la ventana principal del GUIDE con las herramientas que componen el entorno:

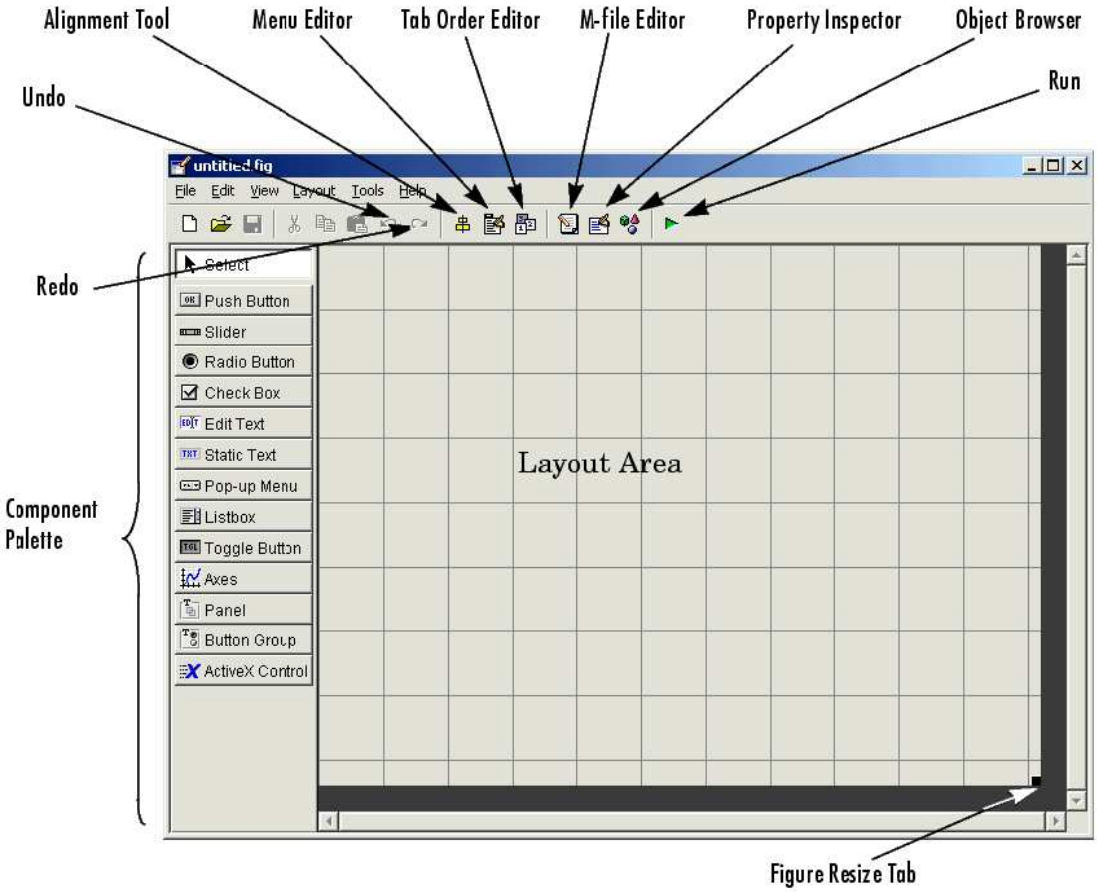

Figura 16. GUIDE

- Barra de menú, en la que podemos encontrar las funciones de edición de GUIs.

- Paleta de componentes (*Component Palette*), que contiene los componentes que podemos añadir a nuestro GUI:
  - ✓ Push Button
  - ✓ Check Box
  - ✓ Edit Text
  - ✓ Static Text
  - ✓ Pop up menú
  - ✓ Axes

- ✓ Toggle Button
- ✓ Radio Button
- ✓ Slider
- ✓ List Box

➢ Barra de herramientas, en la que podemos encontrar botones para activar las herramientas más importantes del GUIDE:

- ✓ Herramienta de alineación (*Alignment Tool*), que nos permitirá establecer opciones de alineación entre los diferentes componentes del GUI.
- ✓ Editor de orden de tabulación (*Tab Order Editor*), que nos permite establecer el orden en que se accede a los componentes al pulsar el tabulador (en lugar de usar el ratón).
- ✓ Editor de ficheros .m (*M-file Editor*), con el que podremos añadir código a las funciones *callback* correspondientes a cada componente.
- ✓ Editor de menús (*Menu Editor*), que nos permitirá definir las entradas que compondrán el menú del GUI.
- ✓ Inspector de propiedades (*Property Inspector*), que nos ofrece la posibilidad de examinar y cambiar las propiedades de los componentes.
- ✓ Navegador de objetos (*Object Browser*), especialmente útil cuando tenemos un número elevado de componentes, nos permite examinar la lista de objetos de nuestro GUI, organizándolos según su jerarquía.
- ✓ Botón de ejecución (*Run Button*).

3.2 ORGANIZACIÓN DE LOS OBJETOS GRÁFICOS EN MATLAB

El programa MATLAB ofrece una serie de funciones de dibujo y visualización de datos, como es el caso de imshow, imagesc o plot, por ejemplo. Cuando llamamos a una función gráfica, MATLAB crea el gráfico requerido usando objetos como ventanas, ejes de coordenadas, líneas, texto, etc.

Podemos trabajar con gráficos en MATLAB a tres niveles diferentes:

- Realizando llamadas a funciones de dibujo y visualización. MATLAB muestra todas las gráficas en un tipo de ventanas especiales conocidas como *figures*, en las que se sitúan unos ejes de coordenadas. Estos ejes son los que proporcionan el sistema de coordenadas necesario para realizar la visualización de los datos.

- Trabajando a "bajo nivel", haciendo las llamadas necesarias a funciones de MATLAB para ir creando los objetos gráficos que sean necesarios para hacer la visualización. A este nivel se realizarán múltiples llamadas a la función set y otras similares para establecer las propiedades de los distintos objetos gráficos que se vayan creando o para modificarlas durante la ejecución del programa. Este modo de trabajar es muy similar al uso tradicional de una biblioteca gráfica que se puede hacer en un lenguaje de programación convencional.

- Empleando el GUIDE, el entorno de desarrollo de GUIs, que nos permitirá definir todos los componentes gráficos que deseemos, establecer sus propiedades e incorporar código de respuesta a cada una de las acciones del usuario a través de una herramienta gráfica de cómodo manejo. Además, tiene la ventaja de que en cualquier momento podremos elegir si deseamos trabajar a alto o bajo nivel, pudiendo acceder al código asociado a través del editor de MATLAB.

Como podemos ver, en cualquiera de estos tres casos estamos haciendo uso del sistema de objetos gráficos de MATLAB. Estos objetos gráficos son los elementos básicos empleados por MATLAB para visualizar datos, y están organizados en una jerarquía como la de la siguiente figura, en la que se muestran los tipos de objetos gráficos empleados más frecuentemente:

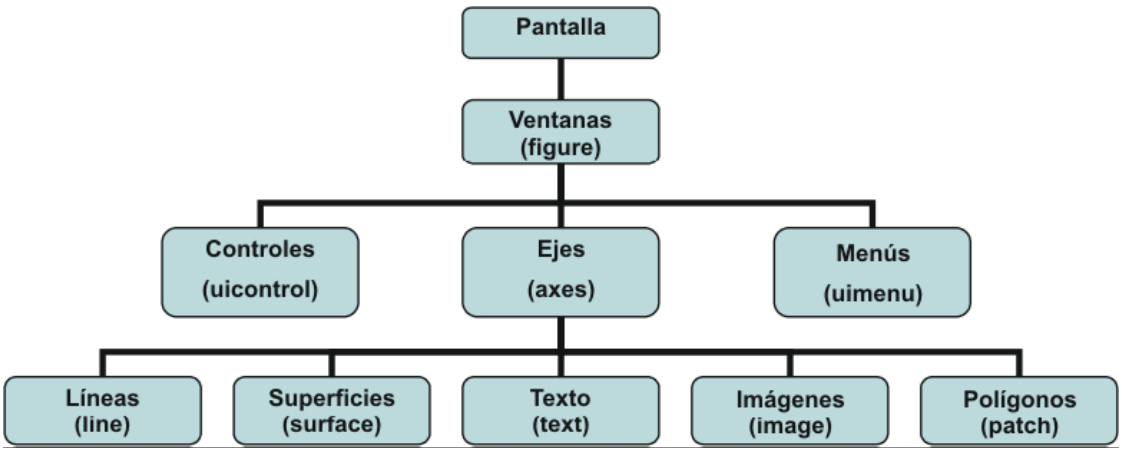

Figura 17. Jerarquía de objetos gráficos de Matlab

Como se ve en la Figura 17, el objeto más general es la pantalla. Es la raíz de la jerarquía. Una pantalla puede contener una o varias ventanas (*figure*). A su vez, cada una de las ventanas podrá contener controles (*uicontrol*), como son los botones, menús desplegables, etc., menús (*uimenu*) y uno o más ejes de coordenadas (*axes*) en los que se podrán representar objetos de nivel inferior. Los ejes pueden incluir cinco tipos de elementos gráficos: líneas (*line*), polígonos (*patch*), texto (*text)*, superficies (*surface*) e imágenes de mapa de bit *(image*).

Al establecerse relaciones padre-hijo entre los objetos gráficos se facilita su gestión, ya que, por ejemplo, cuando se borra un objeto, se borrarán automáticamente todos los descendientes de éste.

Cada objeto está asociado a un identificador (*handle*) único desde el momento de su creación. A través de este identificador podremos modificar las características (llamadas propiedades del objeto) de un objeto gráfico. Naturalmente, también podremos establecer las propiedades de un objeto en el momento de su creación (cambiarlas con respecto a los valores por omisión). El identificador del objeto raíz, la pantalla, es siempre cero. El identificador de las distintas ventanas (*figure*) es un entero que aparecerá en la barra de título de la ventana. Los identificadores de los demás objetos gráficos son números reales. Cualquier identificador se puede obtener como valor de retorno de una función y almacenarse en una variable.

Como hemos dicho, puede haber varias ventanas abiertas, pero sólo una de ellas es la ventana activa en cada momento. De la misma forma, una ventana puede contener varios ejes de coordenadas, pero sólo unos son los ejes activos. El objeto activo será el último objeto creado o sobre el que se haya hecho clic con el ratón.

Podemos obtener los identificadores de la ventana, los ejes y el objeto activos con las órdenes:

gcf: devuelve un entero, el identificador de la ventana activa.

gca: devuelve el identificador de los ejes activos.

gco: devuelve el identificador del objeto activo.

La principal utilidad que tiene conocer los identificadores de los objetos gráficos es que a través de ellos podremos modificar las propiedades de los objetos o incluso borrarlos:

set(id): muestra en pantalla todas las propiedades del objeto al que corresponde el identificador id.

get(id): produce un listado de las propiedades y de sus valores.

set(id, 'propiedad', 'valor'): establece un nuevo valor para la propiedad del objeto con identificador id. Se pueden establecer varias propiedades en la misma llamada a get incluyendo una lista de parejas 'propiedad', 'valor' en la llamada.

get(id, 'propiedad'): obtiene el valor de la propiedad especificada.

delete(id): borra el objeto cuyo identificador es id y todos sus hijos.

3.3 PRIMEROS PASOS CON EL GUIDE

(A) *Comenzando con el GUIDE*

Al llamar al GUIDE con la orden guide aparecerá el diálogo GUIDE Quick Start

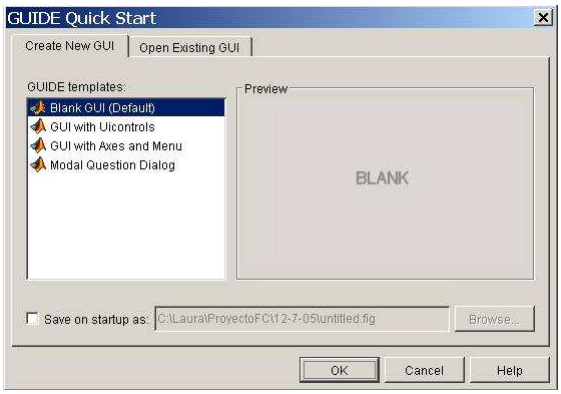

Figura 18. GUIDE Quick Start

y podremos:

- Crear un nuevo GUI en blanco o a partir de las plantillas (*templates*) disponibles.
- Abrir un GUI ya existente.

(B) *Editor de diseño:*

Cuando se abre un GUI, éste se muestra en el Editor de diseño (*Layout Editor*), que es el panel de control de todas las herramientas del GUIDE.

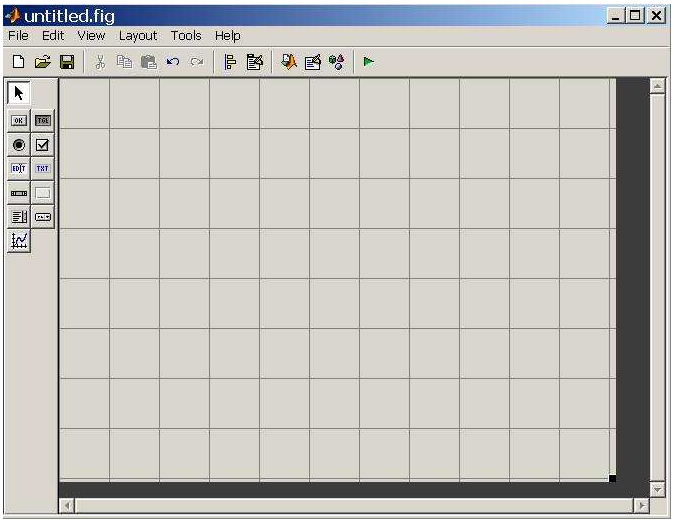

Figura 19. Editor de diseño

Para especificar el tamaño del GUI podemos:

- Redimensionar el *grid* arrastrando la esquina inferior derecha con el ratón.
- Establecerlo de forma exacta accediendo a las propiedades del GUI a través del Editor de propiedades, usando las propiedades de Posición y Unidades.

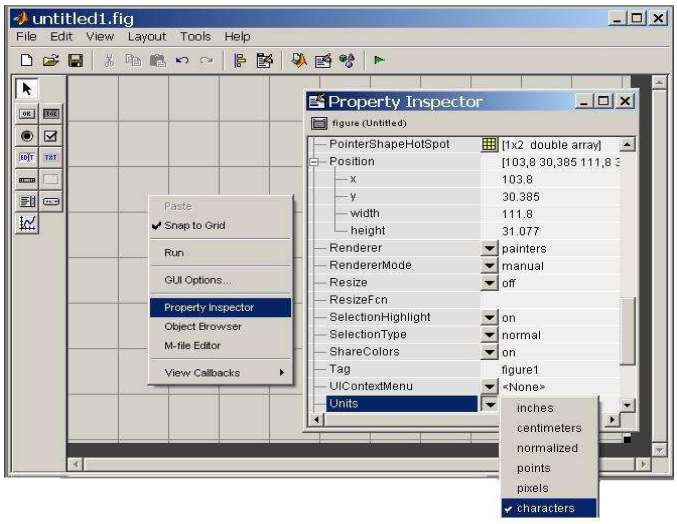

Figura 20. Editor de propiedades

Una buena idea es modificar las preferencias para que nos muestre los nombres de los componentes que podemos añadir a las GUIs, para ello seleccionamos: **File→Preferences→GUIDE→Show names in component palette**, de forma que la paleta de componentes contenga no sólo el icono, sino el nombre de cada componente.

Podemos diseñar un GUI arrastrando elementos (botones, menús desplegables, ejes, etc.) desde la paleta de componentes, que se encuentra en el lado izquierdo del editor. Por ejemplo, en la Figura 21 se ha colocado un menú desplegable.

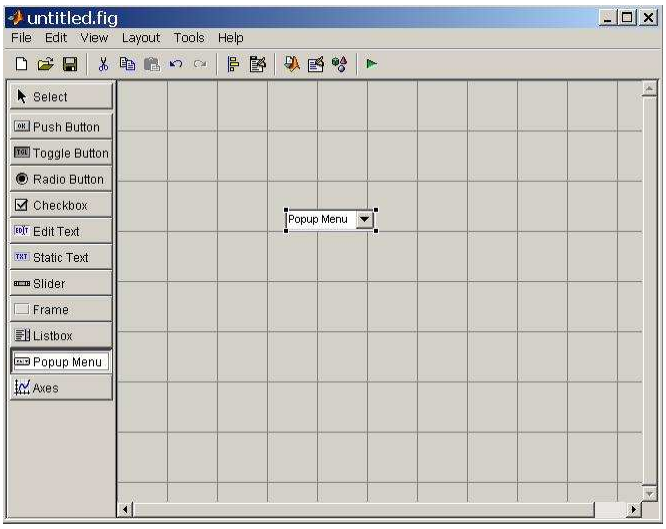

Figura 21. Ejemplo de componente

(C) ***Establecimiento de propiedades:***

Para establecer las propiedades de los diferentes componentes del GUI, seleccionaremos **View→Property Inspector**. Cuando seleccionemos un componente en el editor de diseño, el Inspector de propiedades mostrará las propiedades de ese componente. Si no tenemos seleccionado ningún componente, nos mostrará las propiedades del GUI.

Dependiendo del objeto que hayamos seleccionado, las propiedades que nos mostrará el Inspector de propiedades serán diferentes. Algunas propiedades básicas serían:

- **Nombre** (*Name*) : El valor de la propiedad **Name** será el título que muestre la ventana del GUI cuando se ejecute.

- **Título** (*Title*): El título de un panel permite establecer el título que aparece en la parte superior del mismo. Mediante la propiedad **TitlePosition** podemos controlar dónde debe aparecer el título del panel.

- **String** (*Cadena*): Podemos elegir la etiqueta de algunos componentes, como es el caso de los botones, mediante esta propiedad. En el caso de los menús desplegables, la propiedad **String** controla la lista de opciones del menú. Para establecer las opciones que se ofrecerán a través del menú desplegable, haremos clic en el icono, lo que abrirá una ventana de edición, donde escribiremos en cada línea las opciones a incluir.

▪ **Propiedades de los callbacks**: Los componentes del GUI usan *callback*s para realizar su tarea. Los *callbacks* son funciones que se ejecutan cuando el usuario realiza alguna acción concreta sobre el componente, como es hacer click sobre un botón o seleccionar una opción de un menú desplegable. Cada componente del GUI y cada opción de menú tiene propiedades que especifican sus *callbacks*. Cuando creamos un GUI será preciso que programemos los *callbacks* para controlar el funcionamiento del GUI. Un componente podrá tener varias propiedades de *callback*, pero la más importante es la propiedad **Callback**. El código que incluyamos en esta propiedad realizará la tarea principal del componente en cuestión.

▪ **Etiqueta** (*Tag*): Esta propiedad nos proporciona un identificador único para cada componente. Este identificador se emplea, entre otras cosas, para que GUIDE genere nombres de *callbacks* únicos para los diferentes componentes del GUI. Inicialmente, los componentes tienen nombres predefinidos (por ejemplo, pushbutton1). Si el componente tiene propiedad *callback*, el GUIDE establece como valor *%automatic*.

Cuando se salva o ejecuta el GUI, el GUIDE crea un nombre de función único para cada función *callback* del fichero .m prefijando el valor de la etiqueta *Tag* a la *cadena _Callback* (por ejemplo, pushbutton1_Callback).

Es recomendable redefinir el valor de la propiedad **Tag** para que resulte más descriptiva. Debemos recordar que este valor debe ser único. El GUIDE se encargará de redefinir de forma consecuente las funciones *callback* de los componentes.

(D) ***Almacenamiento de GUIs:***

Para almacenar un GUI, seleccionamos **File→Save As,** o bien, pulsando el botón  . El GUIDE instantáneamente guarda dos ficheros:

• Un fichero .fig, que contiene una descripción del diseño del GUI y de sus componentes.

• Un fichero .m, que contiene el código que controla el GUI, incluyendo los *callbacks* de los componentes.

Estos dos ficheros se corresponden con las dos etapas de creación de un GUI: la de diseño y la de programación.

(E) ***Ejecución del GUI:***

Para ejecutar un GUI, podemos seleccionar **Tools→Run** o hacer clic en el botón de ejecución  . Al ejecutarlo, podremos verlo funcionar en una nueva ventana, fuera del editor de diseño.

(F) ***Programación del fichero .m del GUI:***

Después de diseñar el GUI, podemos programar el fichero .m asociado en el editor de MATLAB. El GUIDE genera automáticamente este fichero a partir del diseño la primera vez que se salva o ejecuta el GUI. El fichero .m del GUI se encarga de las siguientes tareas:

- Inicializa el GUI.
- Contiene código para realizar las tareas necesarias antes de que aparezca el GUI en pantalla, como la creación de datos o gráficos.
- Contiene las funciones *callback* que se ejecutan cada vez que el usuario hace clic en un componente.

Cada *callback* es una subfunción que inicialmente es sólo una definición de función vacía. Nosotros debemos añadir código a los *callbacks* para proporcionarle operatividad al GUI.

Para abrir el fichero .m del GUI podemos hacer clic en el icono de la barra de herramientas del Editor de diseño  . Si abrimos este fichero en el editor de MATLAB, podemos acceder directamente a la función *callback* de cualquier elemento del GUI a través del botón  .

I. **Cómo compartir datos entre los callbacks**: Podemos hacerlo almacenando los datos en la estructura *handles*. Todos los componentes del GUI comparten la misma estructura *handles*. Todos los *callbacks* generados por el GUIDE reciben esta estructura como argumento. Por ejemplo, para almacenar un vector, X, en la estructura *handles*:

   - Elegimos un nombre para el campo de *handles* donde queremos almacenar los datos (por ejemplo, handles.datos).

- Añadimos el campo a la estructura *handles* y le asignamos el valor X:

```
handles.datos = X;
```

- Salvamos la estructura *handles* con la función *guidata*:

```
guidata(hObject,handles)
```

hObject es el identificador (*handle*) del objeto componente que ejecuta el *callback*. Todos los *callbacks* generados por el GUIDE reciben el identificador del componente como argumento.

Para recuperar los datos en otro *callback*, haremos:

```
X = handles.datos;
```

Podemos acceder a los datos de *handles* en cualquier *callback* porque *hObject* y *handles* se pasan como argumento a todos los *callbacks* generados por el GUIDE.

II. Añadir código a la función de apertura: La función de apertura es la primera llamada del fichero .m del GUI. Podemos usarla para realizar tareas que hay que llevar a cabo antes de que el usuario tenga acceso al GUI (por ejemplo, leer datos de un fichero, crear algunos datos, cargar una imagen y visualizarla...). El código de esta función se ejecuta justo antes de que se visualice el GUI, pero una vez que se han creado todos los componentes. El nombre de la función de apertura se forma con el nombre del fichero .m seguido de *_OpeningFcn*. El GUIDE genera automáticamente dos líneas de código en la función de apertura, que deben seguir al código que añadamos:

- handles.output = hObject almacena el identificador del GUI para que pueda ser usado posteriormente por la función de salida (*output function*). Resultará muy útil si queremos devolver el identificador del GUI a la línea de órdenes de MATLAB.

- guidata(hObject,handles) almacena la estructura *handles*.

III. Añadir código a los callbacks: Cuando se ejecuta el GUI y el usuario hace clic en uno de los controles, como puede ser, por ejemplo, un botón, MATLAB ejecutará el *callback* especificado por la propiedad Callback del componente. A continuación de añadir el código deseado en la función es importante no olvidar incluir la llamada guidata(hObject,handles), que hace que las modificaciones que hemos hecho queden almacenadas.

3.4 CREACIÓN COMPLETA DE LAS GUIS DE ESTE PROYECTO

A continuación se procede a la explicación detallada de cómo se han realizado las Interfaces Gráficas de Usuario: medida de un tono, dos tonos, captura de la traza y ventana principal. Para cada una de ellas, primero diseñaremos el GUI y luego programaremos el fichero .m para que hagan exactamente lo que perseguimos.

★ Caso de la Interfaz para el **barrido de potencia de una señal de un tono**.

Para empezar con la realización de la primera Interfaz, como ya se ha comentado, abrimos un nuevo GUI en blanco, elegimos su tamaño y vamos seleccionando el elemento deseado de la barra de componentes y marcando en el GUI la posición y tamaño del mismo, aunque siempre podremos redimensionarlo arrastrando las esquinas. De esta manera, añadimos cuatro paneles (*frame*), con *static text* les llamamos: Polarización, Excitación de entrada, Configuración de la medida y Corrección de pérdidas, respectivamente, y añadimos unos *edit text* observando que al acercarlos a cada panel, el GUIDE lo marca, indicando que éste es el padre potencial del componente en la jerarquía de componentes. Para poder elegir las unidades deseadas, nos hacen falta dos menú desplegables (*pop-up menu*) y un *checkbox* para activar o no la corrección de pérdidas. Podemos usar la herramienta de alineación (*Alignment Tool*) para alinear los componentes entre sí. Para ello:

- Seleccionamos los objetos pulsando *ctrl* y haciendo clic sobre ellos.
- Seleccionamos **Tools→Align Objects** para abrir la herramienta de alineación.

Establecemos las opciones de alineación.

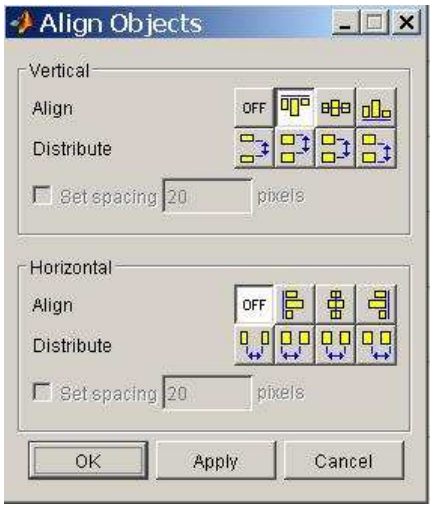

Figura 22. Alineación de objetos

El GUI va tomando la forma de la Figura 23.

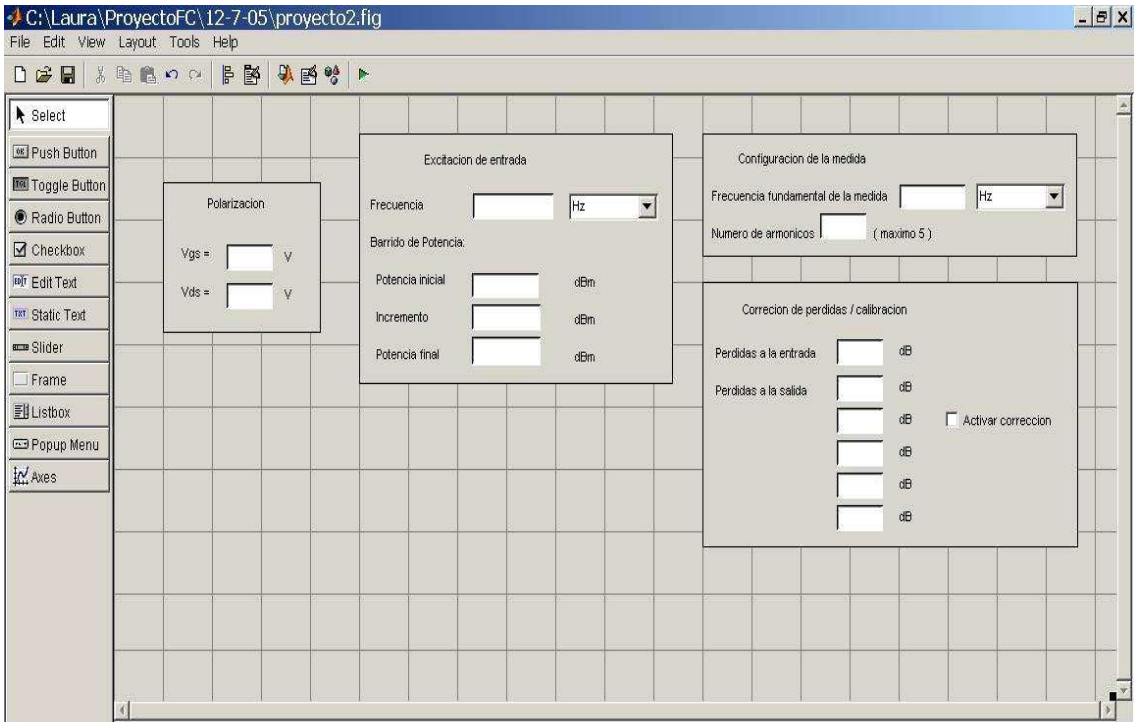

Figura 23. GUI

A continuación, se añade un botón (*push button*) que será el encargado de iniciar la medida una vez el usuario haya rellenado todos los campos anteriormente nombrados. El color y nombre del botón se elegirá a gusto del diseñador.

En cualquier momento, podemos guardar o ejecutar lo que llevamos realizado. Ejecutándolo, obtenemos la Figura 24.

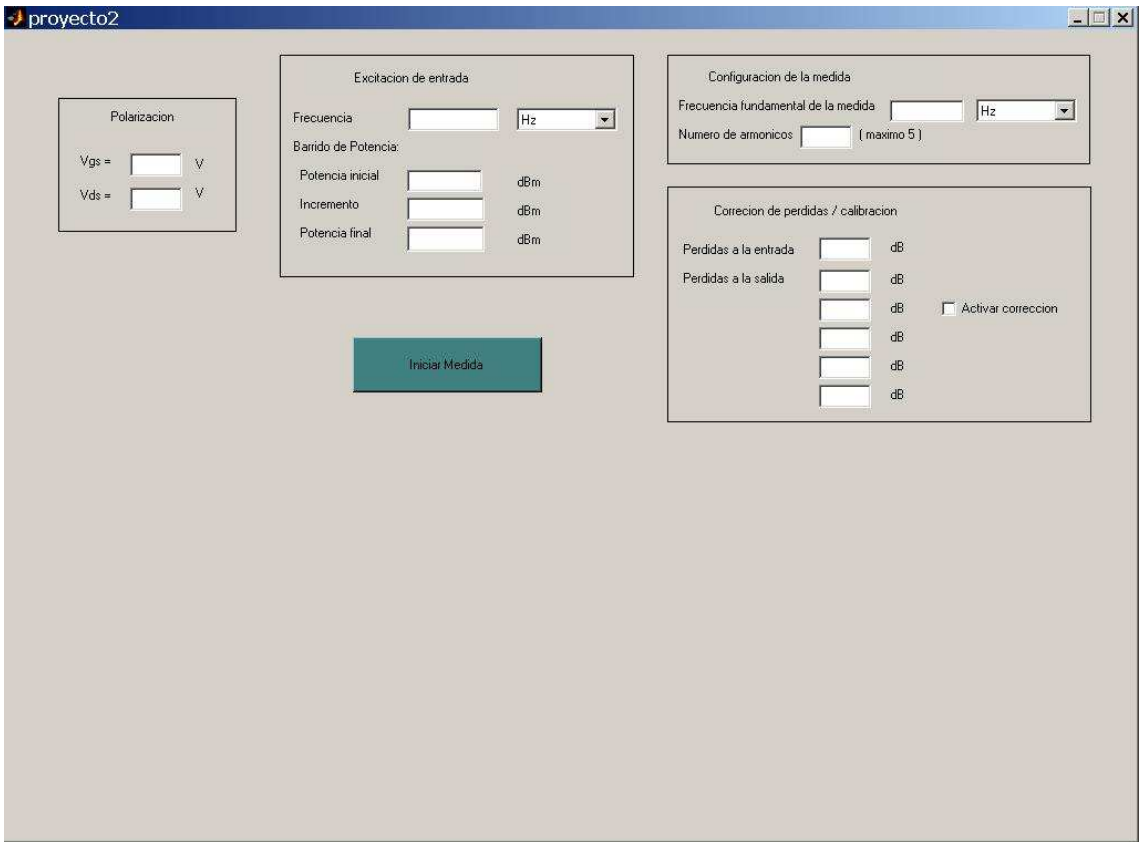

Figura 24. GUI

Con todo esto, ya tendríamos diseñado el barrido de potencia de la medida de un tono deseada. Ahora añadiendo otro *frame* con sus correspondientes *static text* y *edit text* podremos, cuando lo programemos, almacenar los resultados si pulsamos el botón de Guardar resultados.

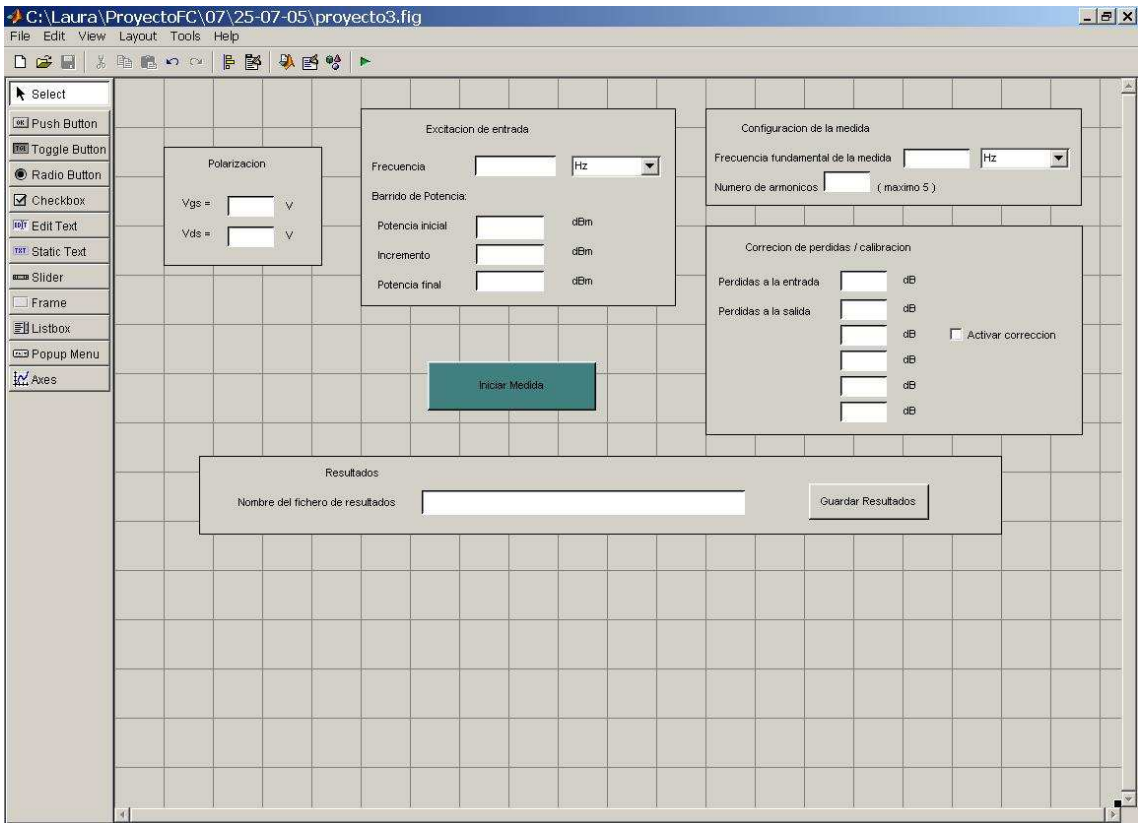

Figura 25. GUI

Los últimos pasos para terminar con la primera Interfaz serían:

Ampliar la parte de Polarización, es decir, añadir *edit text*, con sus respectivos *static text*, para los valores de polarización reales medidos por los equipos del laboratorio.

Sombrear las celdas (*edit text*) dedicadas a las pérdidas, para dar opción al usuario a activar o no la corrección de las mismas, así como las de los valores de polarización medidos, ya que serán resultados devueltos de la ejecución de funciones. Esto se realizará accediendo al Editor de Propiedades, cambiando el color de la propiedad *BackGroundColor*.

Añadir otro *frame* para la representación de los armónicos deseados, así como de la ganancia. Para ello es necesaria la creación de unos ejes (*axes*).

Todo esto queda mostrado en la Figura 26.

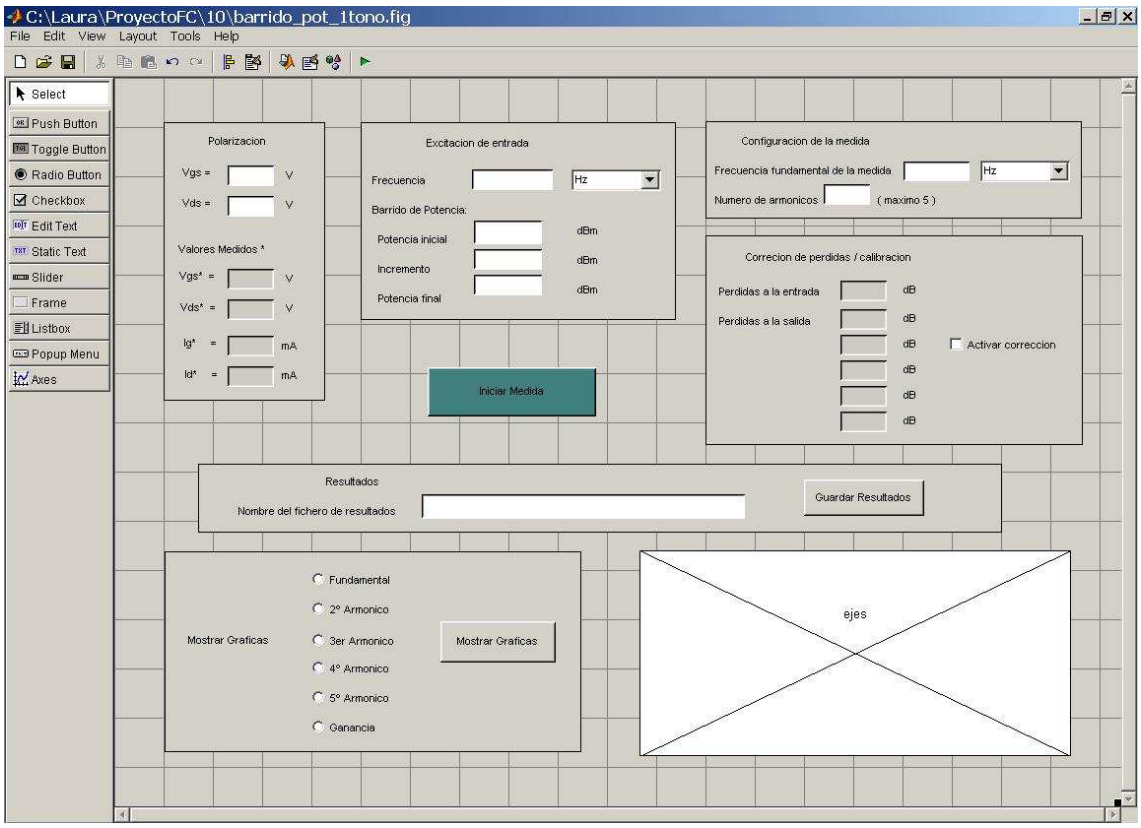

Figura 26. GUI

Presionando el botón de ejecución:

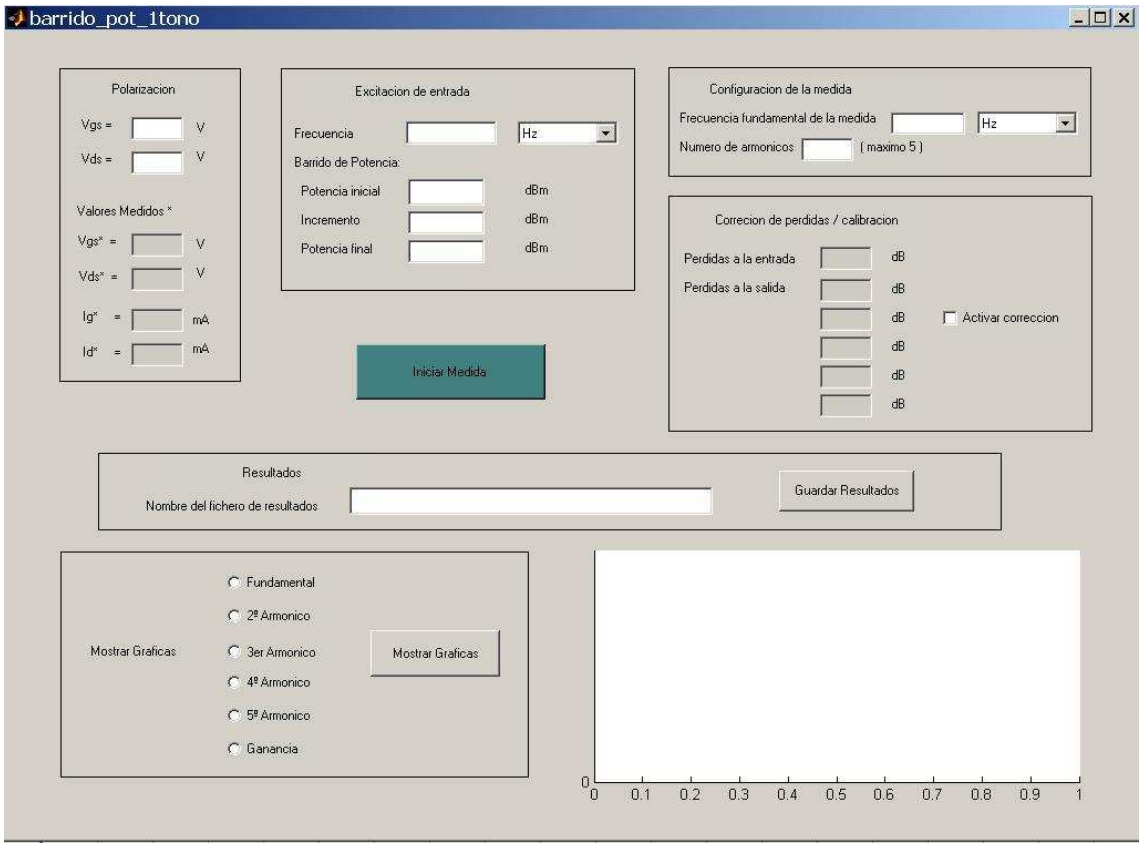

Figura 27. GUI

Una vez diseñado el GUI, la programación del fichero .m dependerá de los objetivos que pretendamos alcanzar con cada componente de la interfaz. Todos los parámetros de configuración de la medida serán pasados posteriormente a las funciones que realizan el control remoto de los equipos de medida disponibles en el laboratorio.

- En primer lugar, la **Polarización**: Un amplificador basado en un tipo MESFET queda polarizado con unos valores de tensión puerta-fuente (*Vgs*) y drenador-fuente (*Vds*) que vienen dados en V (voltios).

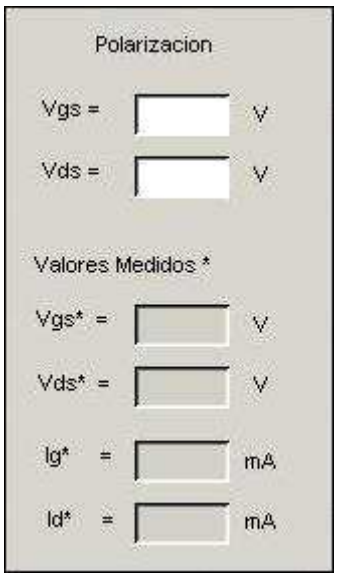

Figura 28. Polarización

Para tomar los valores numéricos de los *edit text* tenemos que escribir en las funciones *callback* correspondientes, la primera instrucción, por ejemplo para el valor de Vgs. Como ya se dijo en la introducción teórica, acto seguido, habrá que guardar este valor en un campo de la estructura *handles* y proceder a salvarla con *guidata*.

```
Vgs = str2double(get(hObject,'String'));
handles.vg = Vgs;
guidata(hObject,handles)
```

Para deshabilitar las celdas de los valores medidos, hemos tenido que añadir el siguiente código pero esta vez, a la función CreateFcn correspondiente.

```
set(hObject, 'Enable', 'off')
```

- La **excitación de entrada**: Este panel se encarga de configurar la frecuencia central de la señal portadora y a su vez, especifica la potencia de inicio, el incremento y la potencia final (dadas todas en dBm) para el barrido de potencia del tono de entrada.

Aparece aquí un menú desplegable para poder elegir entre las diferentes unidades posibles para la frecuencia de la portadora.

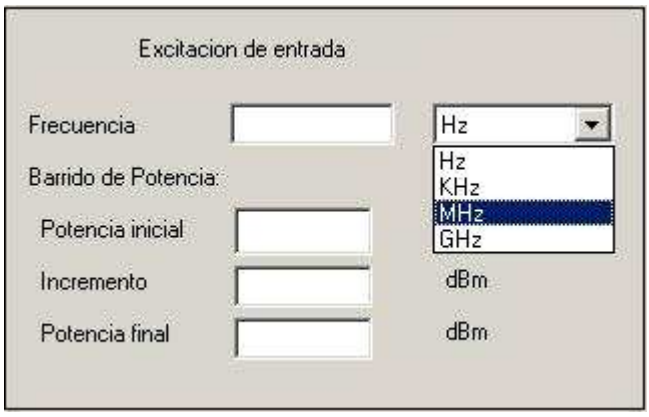

Figura 29. Excitación de entrada

Así, la primera línea devolverá un valor entre 1 y 4, correspondientes a Hz, KHz, MHz y GHz respectivamente.

```
frecuencia = get(hObject,'Value');
handles.unit = frecuencia;
guidata(hObject,handles)
```

- El panel de **Configuración de la medida** nos sirve para definir la frecuencia fundamental de la medida (que será la misma que la frecuencia central ya que queremos medir la amplificación de la frecuencia de entrada) y el número de armónicos que deseemos obtener. Como podemos ver, en lo que respecta a componentes, no encontramos nada distinto a lo anteriormente comentado.

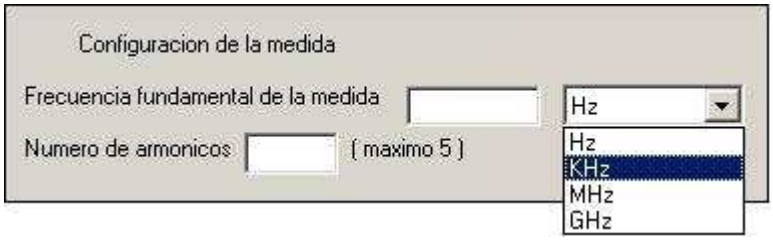

Figura 30. Configuración de la medida

- Para poder iniciar la medida, sólo nos queda elegir si queremos activar o no la **corrección por pérdidas** en los cables. Esta parte no es independiente, sino que depende del valor que se haya dado al campo número de armónicos. De esta forma, si el usuario activa la corrección, quedarán habilitados automáticamente la casilla de pérdidas a la entrada y de las pérdidas a la salida, tantas, como armónicos se hayan especificado. La Figura 31 muestra un ejemplo cuando hemos elegido sólo dos armónicos.

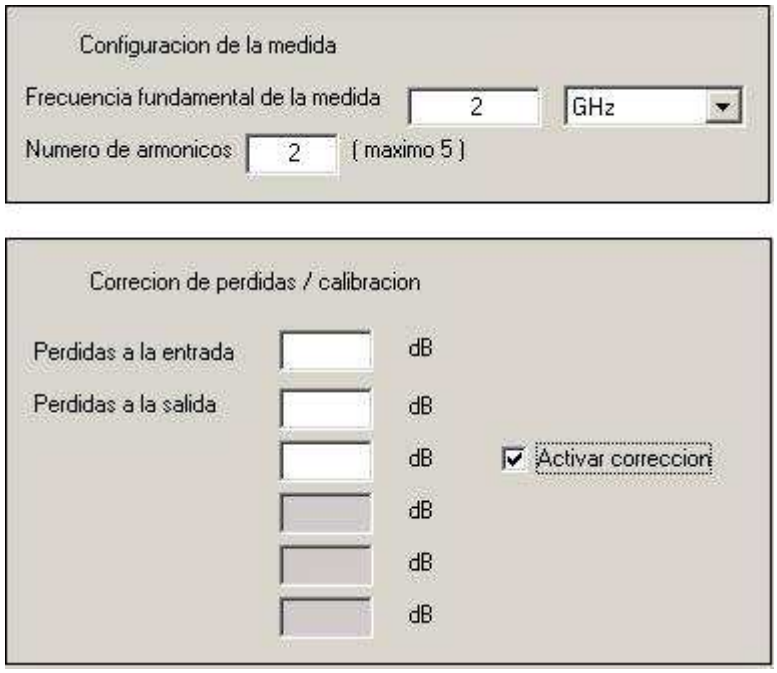

Figura 31. Corrección de pérdidas

Para programar el *check box* de *activar la corrección* habrá que poner un bucle **if** de manera que si el usuario activa la corrección, la variable *correc_activa* se ponga a '1' y luego vendrá un **switch** según el número de armónicos seleccionado.


```
if get(hObject,'Value')==get(hObject,'Max')
correc_activa =1;
```

- Ya estamos preparados para **iniciar la medida**:

Cuando se pulse el botón de *iniciar medida*, 

se empezará a ejecutar su función *callback* que podíamos decir que es la principal, en la que, primero, almacenaremos todos los valores recogidos anteriormente, accediendo a los campos correspondientes de la estructura **handles**. Una vez obtenidos estos valores, pondremos todas las unidades en MHz mediante bucles **switch**. Comprobaremos que el número de armónicos está entre uno y cinco, si no, saldrá un mensaje en la pantalla diciendo: "*número de armónicos erróneo*" y realizaremos la llamada a la función *medida_1t_potRe* que realiza el control remoto de los equipos de medida del laboratorio (la explicación y código de esta función están detallados en el capítulo cuarto), la cual devolverá, por un lado, el conjunto de valores de la potencia de entrada (*Pin*) y la potencia de salida (*medida_1t*) que serán guardadas en sus respectivos campos de la estructura **handles**, y por otro, los valores de polarización reales medidos que se representarán automáticamente en las celdas correspondientes del panel de Polarización, a través de instrucciones del tipo:

```
set(handles.vgsmed, 'Enable', 'on')
set(handles.vgsmed,'BackgroundColor','white')
set(handles.vgsmed,'String',Vgsm);
```

Donde las dos primeras habilitan y quitan el sombreado de la casilla y la tercera añade el valor.

- Ahora podremos **almacenar los resultados** de la medida en un fichero .mat si pulsamos el botón de *guardar resultados* :

Figura 32. Almacenar resultados

Como se ve en el ejemplo, hemos llamado al fichero de resultados "*medida1tono*", y como queremos la cadena de caracteres bastará con poner en el *callback* del *edit text*

```
nombre = get(hObject,'String');
```

y en la función de *Guarda Resultados*

```
eval(['save ' nombre ' Pin medida_1t']);
```

de este modo, aparecerá un fichero .mat conteniendo los resultados obtenidos, al que podemos acceder escribiendo en la pantalla de comandos de Matlab lo siguiente:

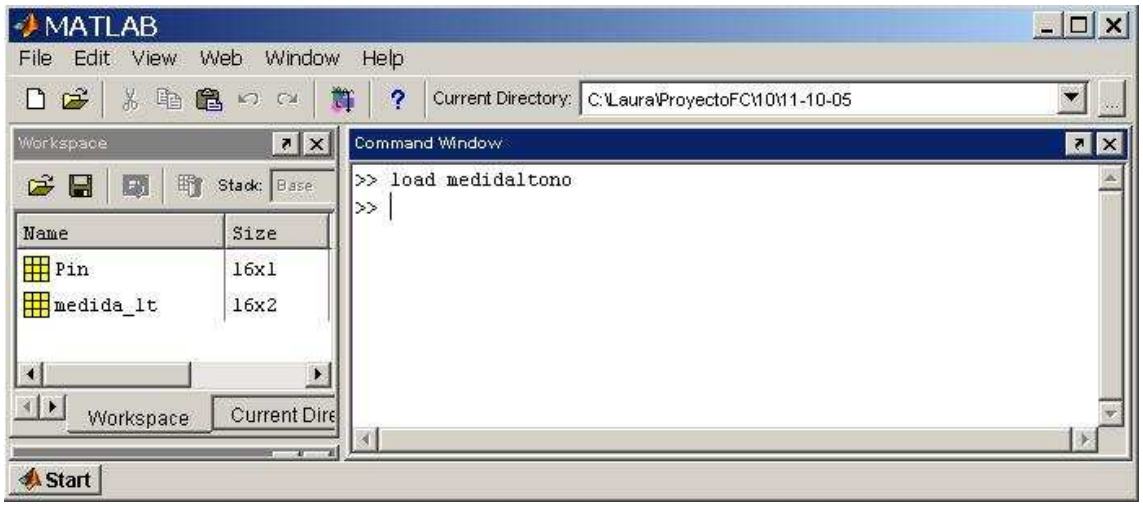

Figura 33. Pantalla de comandos de MATLAB

Apareciendo instantáneamente las variables guardadas *Pin* y *medida_1t* en el espacio de trabajo como se aprecia en la Figura 33.

- Por último se encuentra la opción de **Mostrar los resultados** obtenidos o parte de ellos, según prefiera el usuario. En la Figura 34 hemos marcado el armónico fundamental, el segundo y la ganancia, por ejemplo.

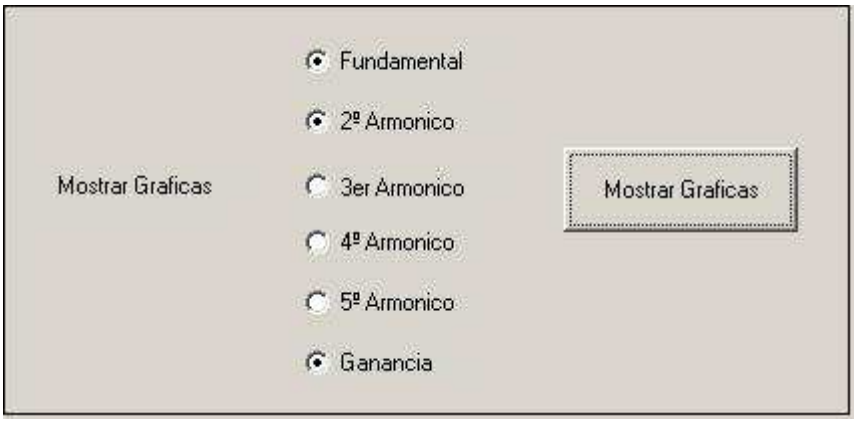

Figura 34. Mostrar resultados

Este panel consta de nuevos componentes, los llamados *Radio Buttons*, que como ya se ha explicado se programan de una manera similar a los *Check Box*, de tal modo que tendríamos que escribir para el fundamental entre otros:

```
if (get(hObject,'Value') == get(hObject,'Max'))
   armofund_act=1;
else
   armofund_act=0;
end
handles.armofundact = armofund_act;
guidata (hObject,handles)
```

Al pulsar el botón de **Mostrar Gráficas** la función que se realizará será un bucle donde para cada armónico seleccionado se irá ejecutando este código:

```
plot(Pin,Pout(1:tam,1),'r')
xlabel('Pin (dBm)')
ylabel('Pout (dBm)')
hold on
```

★ Interfaz para la medida del **barrido de potencia de una señal de dos tonos**

El diseño de esta segunda GUI sigue las mismas pautas que el anterior, aunque ahora habrá dos tonos separados una cierta distancia en frecuencia, por lo que tendremos una frecuencia superior y otra inferior, y aparecerán los comentados productos de intermodulación correspondientes. Haremos especial hincapié en la parte de programación.

Así, abriremos un nuevo GUI, elegiremos su tamaño y seleccionaremos un panel para la parte de polarización (que será exactamente igual al anterior), otro para la excitación de entrada que se diferencia del primero en la aparición de un *edit text* encargado de definir la separación de frecuencias que tendrán los tonos, seguido de su respectivo menú desplegable para especificar las unidades, y de dos celdas que irán sombreadas y mostrarán, al ser programadas, las frecuencias superior e inferior de los tonos en cuestión (debidamente calculadas en MHz). El tercer panel estará dedicado a la configuración de la medida, con el cambio del *edit text* correspondiente al número de armónicos por un menú desplegable que indica el orden máximo de los productos de intermodulación que deseamos obtener. En este trabajo se ha decidido limitar la medida hasta los productos de intermodulación de quinto orden, al presentar los órdenes superiores niveles poco significativos. Para la parte de pérdidas, al contar ahora con dos tonos, habrá que añadir las pérdidas correspondientes al tono superior, tanto a la entrada como a la salida, así como las pérdidas debidas a los productos de intermodulación de tercer y quinto orden.

El botón de *iniciar medida* y el panel de *guardar resultados*, los hemos diseñado análogos a la primera interfaz.

Y por último, la representación de los resultados tendrá que contener los *radio buttons* correspondientes a todos los tonos que se han podido medir, es decir, tono inferior, tono superior, IM3 inferior, IM3 superior, IM5 inferior e IM5 superior.

En la Figura 35, podemos observar el diseño completo del GUI y en la Figura 36 la pantalla de ejecución del mismo.

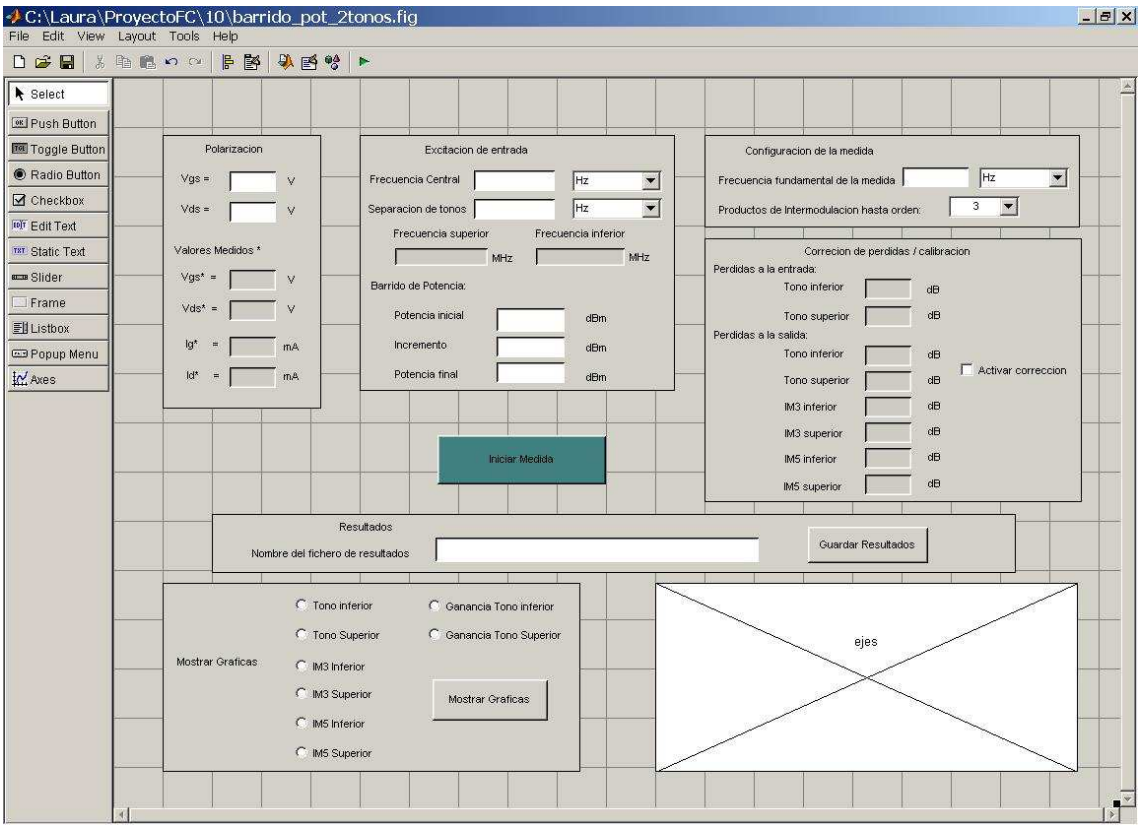

Figura 35. GUI

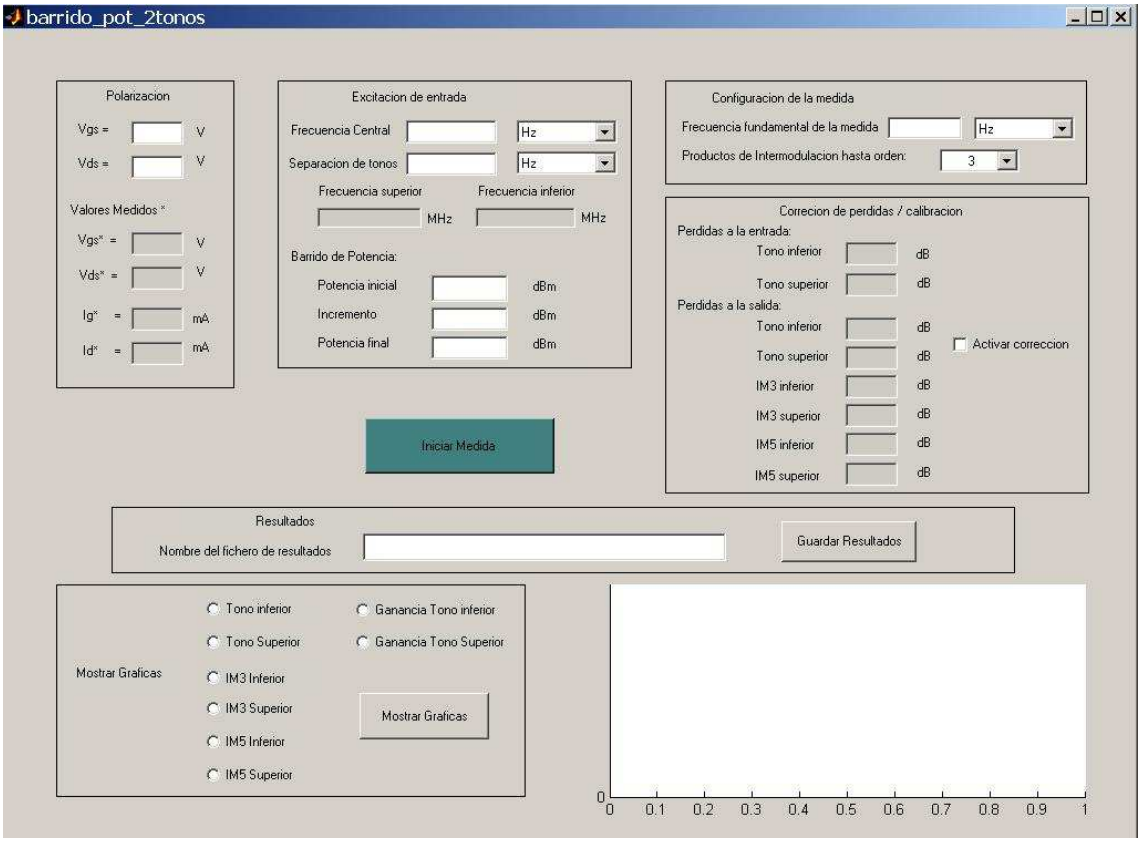

Figura 36. GUI

En lo que respecta a la programación del fichero .m:

- La parte de **polarización** y de **guardar resultados** no sufre ningún cambio.

- La **excitación de entrada** consta de la frecuencia central entre los tonos que estarán separados el valor que indique el usuario en la casilla correspondiente a *Separación de tonos*. Una vez se especifiquen estos valores, se activarán las celdas sombreadas mostrando los valores de la frecuencia inferior y superior de los tonos. El barrido de potencia no cambia.

Figura 37. Excitación de entrada

Esto se realiza mediante el siguiente código, donde *frec_i* es la frecuencia central o intermedia y *Df* la separación de los tonos.

```
set(handles.Frec_sup, 'Enable', 'on')
set(handles.Frec_sup,'BackgroundColor','white')
set(handles.Frec_inf, 'Enable', 'on')
set(handles.Frec_inf,'BackgroundColor','white')
frec_i = handles.frec_i;
frec_sup= frec_i +(Df/2);
frec_inf= frec_i -(Df/2);
set(handles.Frec_sup,'String',frec_sup);
set(handles.Frec_inf,'String',frec_inf);
```

- En la **configuración de la medida**, tenemos como antes, la frecuencia fundamental de la medida, que también coincidirá con el valor de la frecuencia central y la posibilidad de mostrar los productos de intermodulación hasta orden 3º ó 5º.

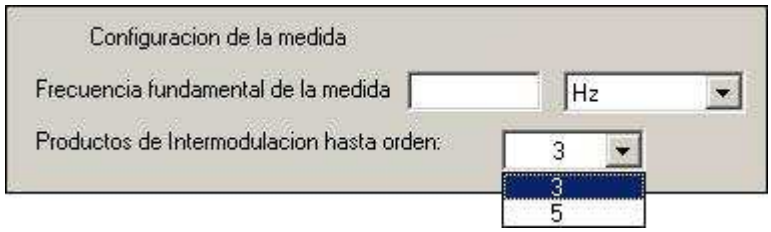

Figura 38. Configuración de la medida

- La parte de la **corrección de pérdidas** depende igualmente de la selección anterior. De manera que si se activa la corrección por parte del usuario, se habilitarán las celdas correspondientes para que el usuario pueda introducir las pérdidas respectivas.

En la Figura 39, hemos elegido obtener productos de intermodulación hasta orden 3 y hemos activado la corrección, por lo que como se observa, se han habilitados las casillas correspondientes.

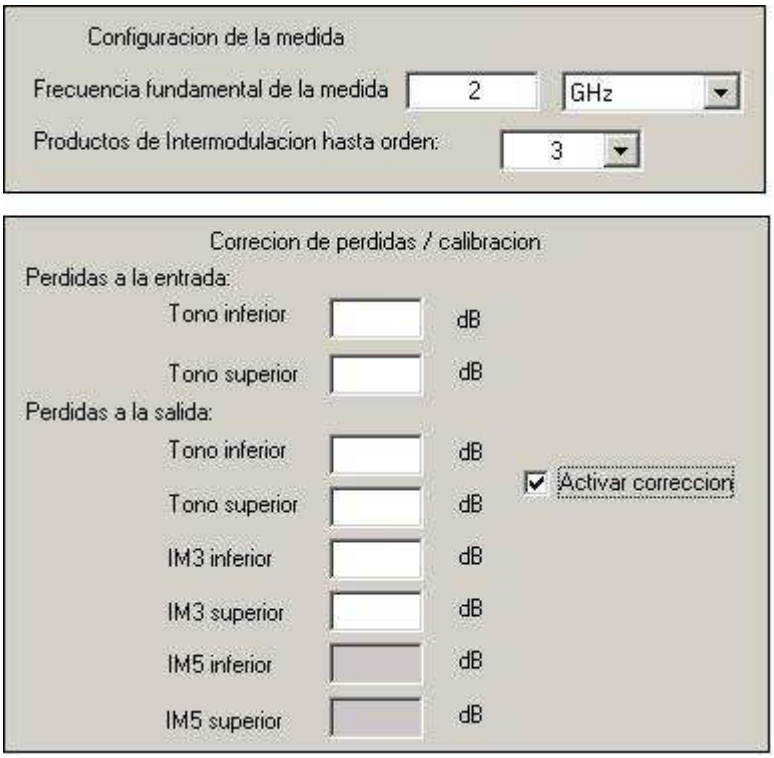

Figura 39. Corrección de pérdidas

- Cuando pulsamos el botón de **iniciar medida,** se empezará a ejecutar su función *callback*, en la que, primero, almacenaremos todos los valores recogidos anteriormente, accediendo a los campos correspondientes de la estructura **handles**. Una vez obtenidos estos valores, pondremos todas las unidades en MHz mediante bucles **switch**y realizaremos la llamada a la función *medida_2t_potRe* que controla de forma remota los equipos de medida del laboratorio(la explicación y código de esta función están detallados en el capítulo cuarto), la cual devolverá, por un lado, el conjunto de valores de potencia de entrada (*Pin*) y potencia de salida (*medida_2t*) que serán guardadas en sus respectivos campos de la estructura **handles**, y por otro, los valores de polarización reales medidos que se representarán automáticamente en las celdas correspondientes del panel de Polarización.

- A la hora de **Mostrar las Gráficas**, disponemos de los ya conocidos *radio buttons* donde hemos completado los tonos que faltaban. En la Figura 40 hemos marcado para visualizar los tonos inferior y superior, así como los productos de intermodulación de tercer orden.

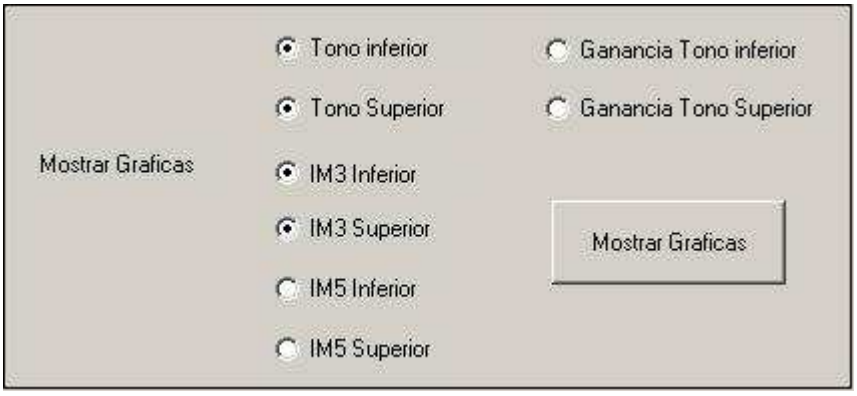

Figura 40. Mostrar resultados

★ Interfaz para la **captura de la traza**:

En esta tercera interfaz vamos a capturar la traza de la densidad espectral de potencia para señales de 1 tono, 2 tonos o señales QPSK del tipo 3GPP-WCDMA. La interfaz sigue la forma de las anteriores, a diferencia de que aquí no realizamos un barrido de potencia, sino que vamos a representar el espectro de la señal (Potencia de salida (dBm) en función de la frecuencia (MHz)) para valores de potencia de entrada especificados por el usuario.

La interfaz quedaría de la siguiente forma:

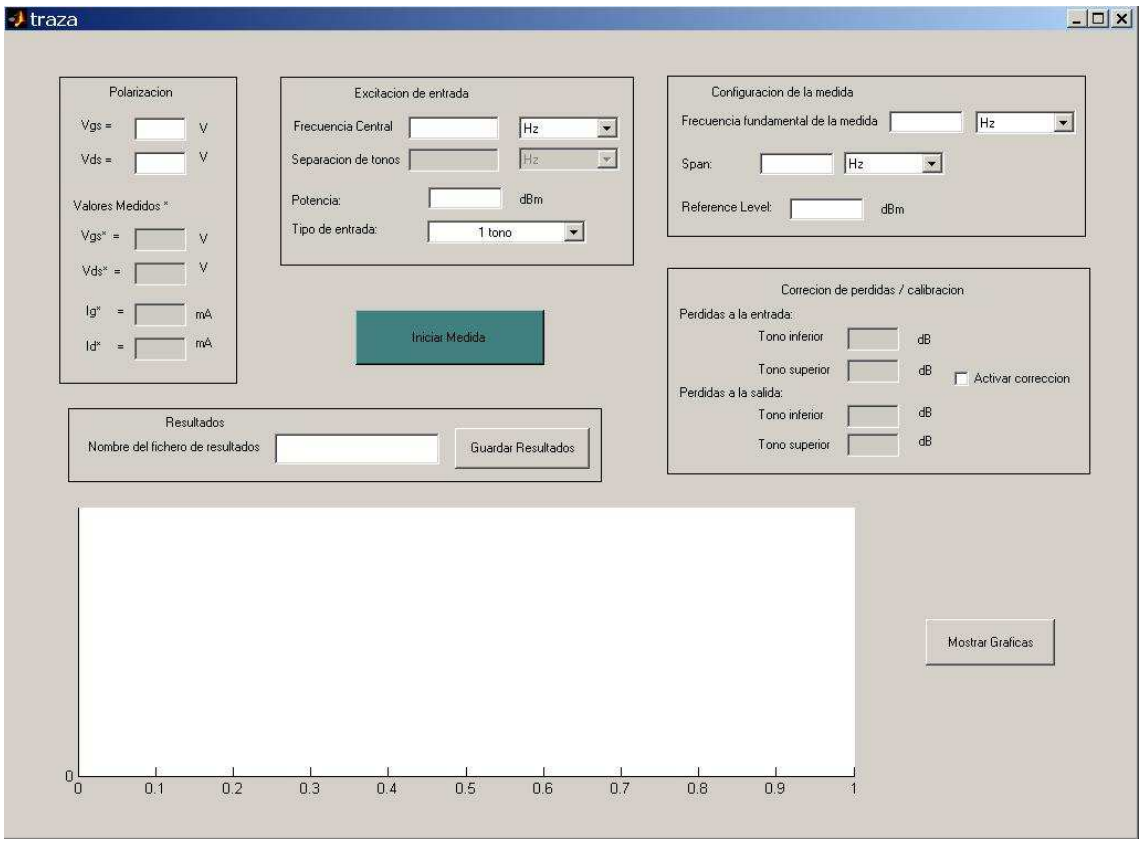

Figura 41. GUI

Como podemos apreciar, en el panel de **excitación de entrada** hemos añadido un único valor de *potencia* de entrada y un menú desplegable para poder seleccionar el *tipo de entrada*, ya sea de 1 tono, 2 tonos o WCDMA. Así mismo, los componentes que definen la separación de los tonos, están sombreados y deshabilitados con el fin de activarlos sólo en el caso en que la entrada sea una señal de dos tonos, como se puede ver en la Figura 42.

Figura 42. Excitación de entrada

Para ello, escribiremos en la función *callback* de *tipo de entrada* el siguiente código, donde primero tomaremos el valor de *tipo_entrada*, siendo 1, 2 ó 3, y si es igual a 2 estaremos en el caso de una señal de dos tonos, por tanto procedemos a activar tanto la separación de frecuencias (*Df*) como sus unidades (*uniDf*). Si estuviéramos en cualquier otro caso, tendríamos que volver a desactivar las casillas, es decir, deshabilitarlas y sombrearlas usando ese rango de colores [R G B].

```
tipo_entrada = get(hObject,'Value');
handles.tipoentrada = tipo_entrada;
guidata(hObject,handles)

if (tipo_entrada==2)
   set(handles.Df, 'Enable', 'on')
   set(handles.Df,'BackgroundColor','white')
   set(handles.uniDf, 'Enable', 'on')
   set(handles.uniDf,'BackgroundColor','white')
else
   set(handles.Df, 'Enable', 'off')
   set(handles.Df,'BackgroundColor',[0.8 0.78 0.78])
   set(handles.uniDf, 'Enable', 'off')
   set(handles.uniDf,'BackgroundColor',[0.8 0.78 0.78])
end
```

Cuando vamos a **configurar la medida**, estableceremos la *frecuencia fundamental* de tal modo que podamos visualizar en la gráfica lo que deseemos. Por ejemplo, si hablamos de una señal de un tono de frecuencia central 2 GHz, tendremos que poner una frecuencia fundamental de unos 4GHz (2º armónico) para poder observar los armónicos segundo y tercero que aparecen. Así mismo, fijaremos un *Span* y un *Reference Level* adecuados.

Figura 43. Configuración de la medida

La **corrección de pérdidas** dependerá del tipo de excitación de entrada que se produzca. Si se activa la corrección y tenemos una señal de dos tonos, se activarán todas las celdas existentes, en cambio, si ponemos como entrada una señal de un tono o 3GPP-WCDMA el panel se quedaría como muestra la Figura 44.

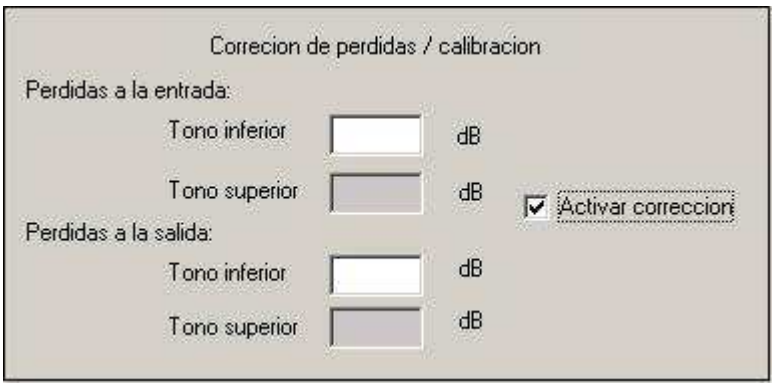

Figura 44. Corrección de pérdidas

Para **iniciar la medida** volvemos a pulsar el botón y se empezará a ejecutar su función *callback*, en la que, al igual que antes, almacenaremos todos los valores recogidos anteriormente, accediendo a los campos correspondientes de la estructura **handles**, pondremos todas las unidades en MHz mediante bucles **switch** y realizaremos la llamada a la función *captura_trazaR* que controla de forma remota los equipos de medida (la explicación y código de esta función están detallados en el capítulo cuarto), la cual devolverá, por un lado, la potencia de entrada (*Pin*), la traza (*tr1*) y la frecuencia (*f*) que serán guardadas en sus respectivos campos de la estructura **handles**, y por otro, los valores de polarización reales medidos que se representarán automáticamente en las celdas correspondientes del panel de Polarización.

La función de **guardar resultados** se programaría como ya hemos dicho:

```
eval(['save ' nombre ' frect traza Pin']);
```

Obteniéndose así un fichero .mat con la frecuencia, la traza y la potencia de entrada. Para **mostrar** el espectro de la señal sólo nos queda por hacer:

```
plot(frect,traza,'r')
xlabel('Frecuencia')
ylabel('Pout')
```

★ Interfaz **raíz**:

Es la ventana principal pero a su vez, la menos complicada de realizar. Su misión es llamar a cada una de las anteriores. Su diseño consiste en un panel compuesto de tres subpaneles, cada uno con un botón de *pulsar* dedicado a un tipo de medida, ya sea un barrido de potencia de una señal de un tono, un barrido de potencia de una señal de dos tonos, o bien, la captura de la traza de una señal. La hemos llamado raíz por ser la interfaz de partida. La podemos ver en la Figura 45.

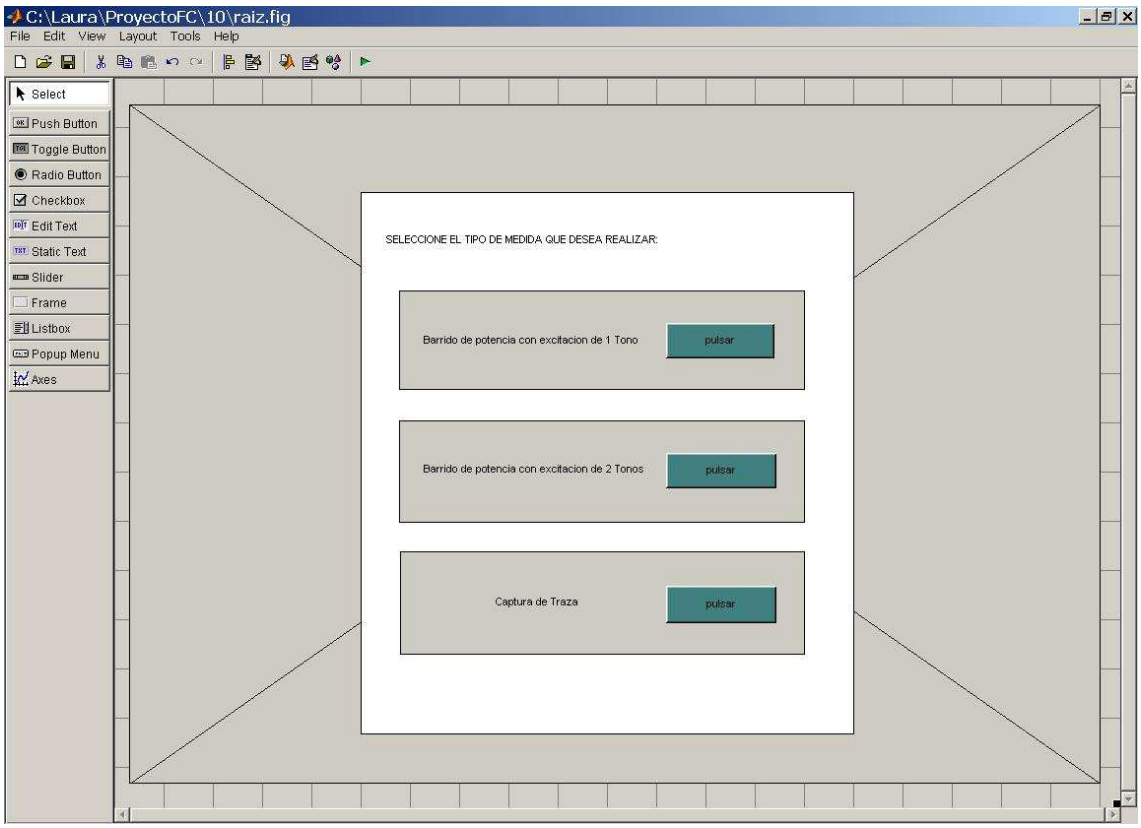

Figura 45. GUI

Como novedad podemos decir que hemos creado esos ejes inmensos que se ven, para añadir un archivo de dibujo (.tif) con el título. Es una manera particular de hacerlo, pero existen otras perfectamente válidas también. Esto se deja a la elección del diseñador. Las líneas de código, dentro de la función que se indica, serían las siguientes.

```
function raiz_OpeningFcn(hObject, eventdata, handles, varargin)

handles.imagen=imread('figura3.tif');
imagesc(handles.imagen), colormap(gray(256)), axis off;
```

Obteniendo como resultado al ejecutarlo la Figura 46.

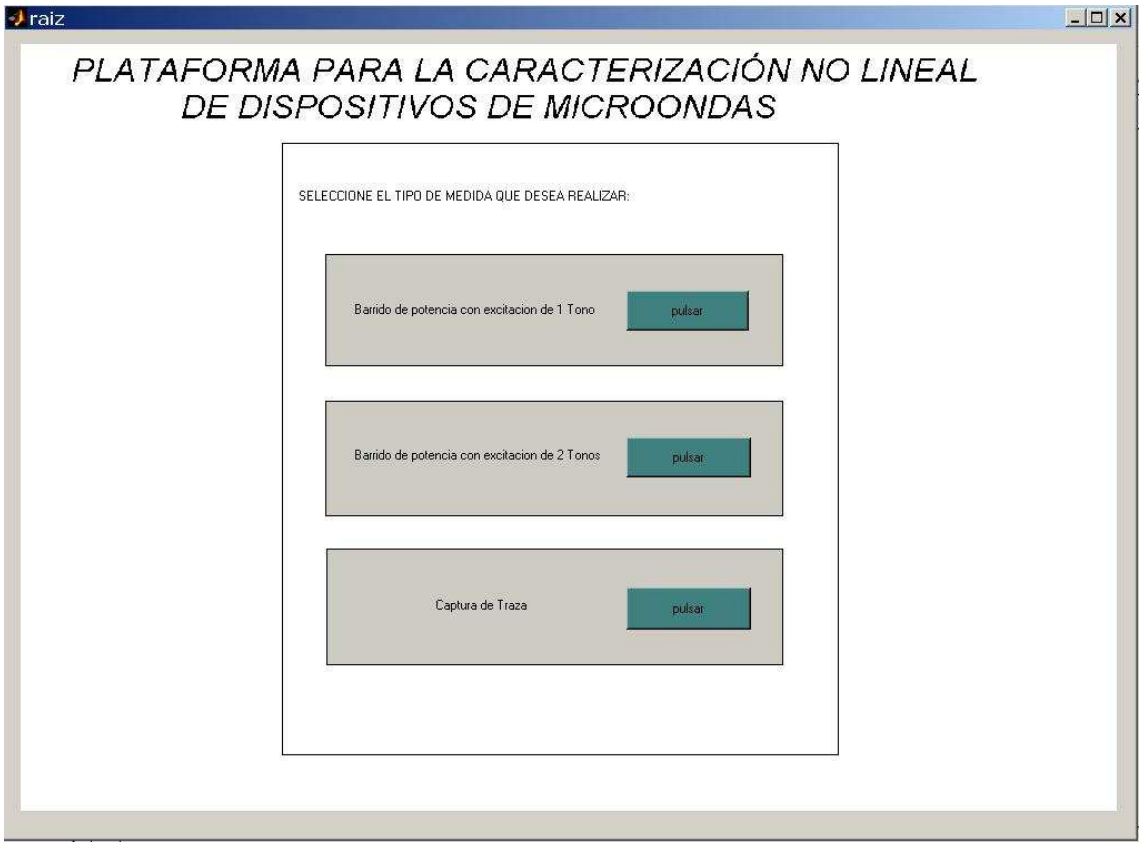

Figura 46. GUI