Comunicaciones Digitales 2002
Tutorial de Matlab Guide*
(SIN REVISAR)

Introducción:
Matlab Guide* es un entorno de programación visual que ofrece Matlab para poder realizar y ejecutar programas de Simulación a medida de forma simple , tiene las características básicas de todos los programas visuales como Visual Basic o Visual C++.

por ejemplo una aplicación se puede ver de la siguiente manera

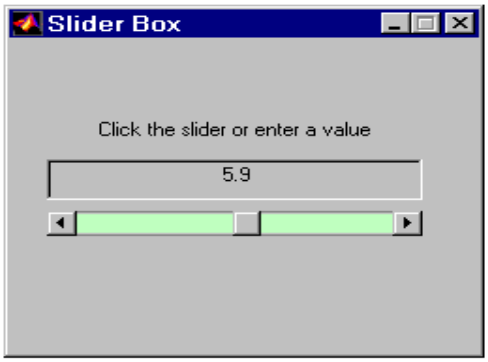


Figura 1

Ejecución : Desde la ventana de comando del Matlab se debe ejecutar el comando *guide*.
Esto abre la consola de edición de la parte grafica de la aplicación a implementar (.fig), es decir , colocar botones, cuadros de dialogo, graficas , texto, etc.

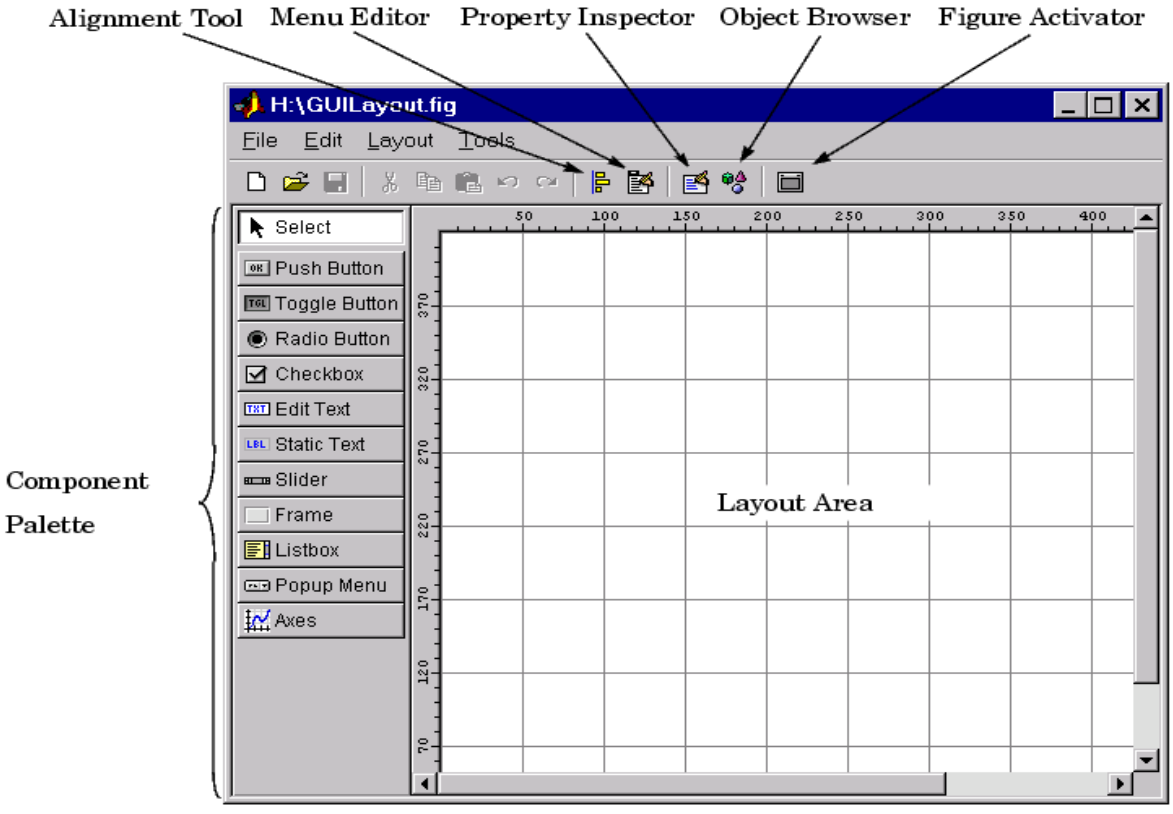


Figura 2

Cada uno de estos elementos tienen un conjunto de propiedades a las cuales podemos acceder con el botón derecho del mouse, una vez clickeado este aparece el siguiente cuadro:

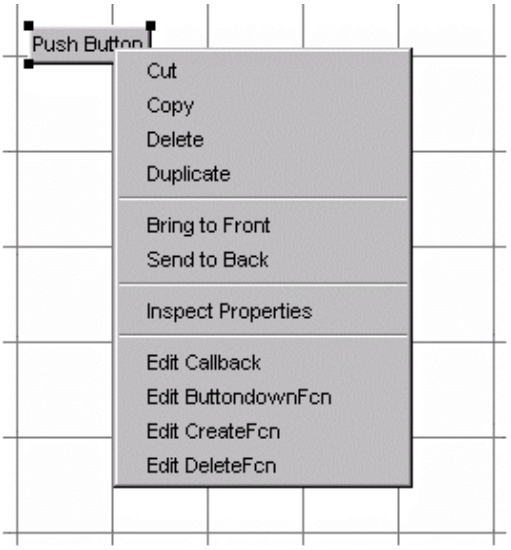


Figura 3

Para editar las propiedades de cada elemento seleccionamos la opción *Properties Inspector* y se abre una consola (la cual variará según que elemento se este editando) con todas las propiedades que podemos editar, ej color, posición, tamaño, font, etc.
Una de las opciones de mayor interés para nosotros en la figura anterior es *Edit Callback* . Esta última abre el archivo .m asociado (ejecutable Matlab) y nos posiciona en la sección del programa que corresponde a la subrutina que se ejecutara cuando se realice una determinada acción sobre el elemento que estamos editando.

Por ejemplo para el botón 1, Edit Callback nos posiciona en la siguiente parte del programa:

```
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.pushbutton1.
disp('pushbutton1 Callback not implemented yet.')
```

Como funciona una aplicación Guide? Consta de dos archivos uno .m (ejecutable) y otro . fig la parte grafica. Las dos partes están unidas a través de las subrutinas callback. Una vez que se graba los archivos desde la consola de emisión (si salvamos la .fig automáticamente lo hace el .m asociado) podemos ejecutar el programa en la ventana de comando de Matlab solamente escribiendo el nombre del archivo solamente. Por ejemplo si guardamos un archivo ej.fig y ej.m escribiendo *ej* y presionando *enter* se ejecuta el programa.
El archivo .m que se crea tiene una estructura predeterminada. Consta de un encabezado y a continuación viene el código correspondiente a las siguientes subrutinas. Por ejemplo una aplicación cuya figura tenga 3 botones, un grafico y un cuadro de edición tendrá un archivo .m con las siguiente estructura inicial (todavía no se agrega el código de la subrutina) como la siguiente:

```
function varargout = untitled1(varargin)
% UNTITLED1 Application M-file for untitled1.fig
%    FIG = UNTITLED1 launch untitled1 GUI.
%    UNTITLED1('callback_name', ...) invoke the named callback.

% Last Modified by GUIDE v2.0 20-Aug-2002 19:57:33

if nargin == 0  % LAUNCH GUI

        fig = openfig(mfilename,'reuse');

        % Use system color scheme for figure:
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));

        % Generate a structure of handles to pass to callbacks, and store it.
        handles = guihandles(fig);
        guidata(fig, handles);

        if nargout > 0
                varargout{1} = fig;
        end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end

end

%| ABOUT CALLBACKS:
%| GUIDE automatically appends subfunction prototypes to this file, and
%| sets objects' callback properties to call them through the FEVAL
%| switchyard above. This comment describes that mechanism.
%|
%| Each callback subfunction declaration has the following form:
%| <SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES, VARARGIN)
%|
%| The subfunction name is composed using the object's Tag and the
%| callback type separated by '_', e.g. 'slider2_Callback',
%| 'figure1_CloseRequestFcn', 'axis1_ButtondownFcn'.
%|
%| H is the callback object's handle (obtained using GCBO).
%|
%| EVENTDATA is empty, but reserved for future use.
%|
%| HANDLES is a structure containing handles of components in GUI using
%| tags as fieldnames, e.g. handles.figure1, handles.slider2. This
%| structure is created at GUI startup using GUIHANDLES and stored in
%| the figure's application data using GUIDATA. A copy of the structure
%| is passed to each callback.  You can store additional information in
%| this structure at GUI startup, and you can change the structure
%| during callbacks.  Call guidata(h, handles) after changing your
%| copy to replace the stored original so that subsequent callbacks see
%| the updates. Type "help guihandles" and "help guidata" for more
%| information.
%|
%| VARARGIN contains any extra arguments you have passed to the
%| callback. Specify the extra arguments by editing the callback
%| property in the inspector. By default, GUIDE sets the property to:
```

Encabezado

handles, Puntero a todas varialbes y elementos de la aplicación

guidata(), comando para guardar las variables de la aplicación

Comentario, se puede borrar

```
%| <MFILENAME>('<SUBFUNCTION_NAME>', gcbo, [], guidata(gcbo))
%| Add any extra arguments after the last argument, before the final
%| closing parenthesis.

% --------------------------------------------------------------------
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.pushbutton1.
disp('pushbutton1 Callback not implemented yet.')

% --------------------------------------------------------------------
function varargout = pushbutton2_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.pushbutton2.
disp('pushbutton2 Callback not implemented yet.')

% --------------------------------------------------------------------
function varargout = pushbutton3_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.pushbutton3.
disp('pushbutton3 Callback not implemented yet.')

% --------------------------------------------------------------------
function varargout = edit1_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.edit1.
disp('edit1 Callback not implemented yet.')
```

Subrutina Boton 1

Subrutina Boton 2

Subrutina Boton 3

Subrutina Cuadro de Edición 1

La siguiente figura muestra la interacción entre la la figura (.fig) y el archivo (.m):

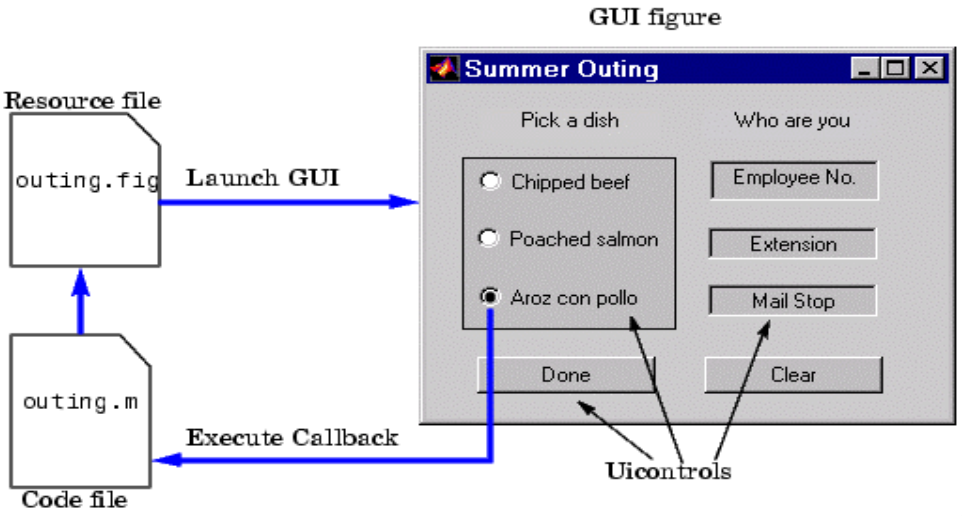


Figura 4

Observación sobre Salvar: Si guarda una modificación del archivo cuando aparece el cuadro de dialogo (ver figura)que pregunta si hacemos Replace o Append debemos elegir Append, si se selecciona Replace borra todo el archivo .m es decir lo pone a cero, no utilizar Replace.

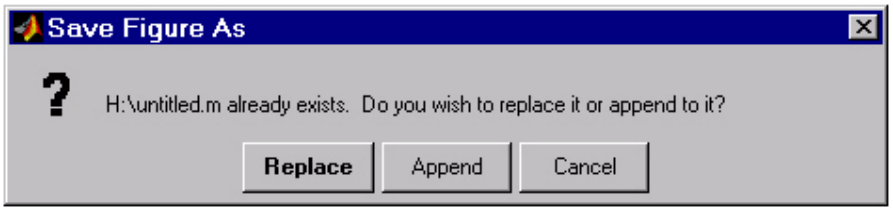
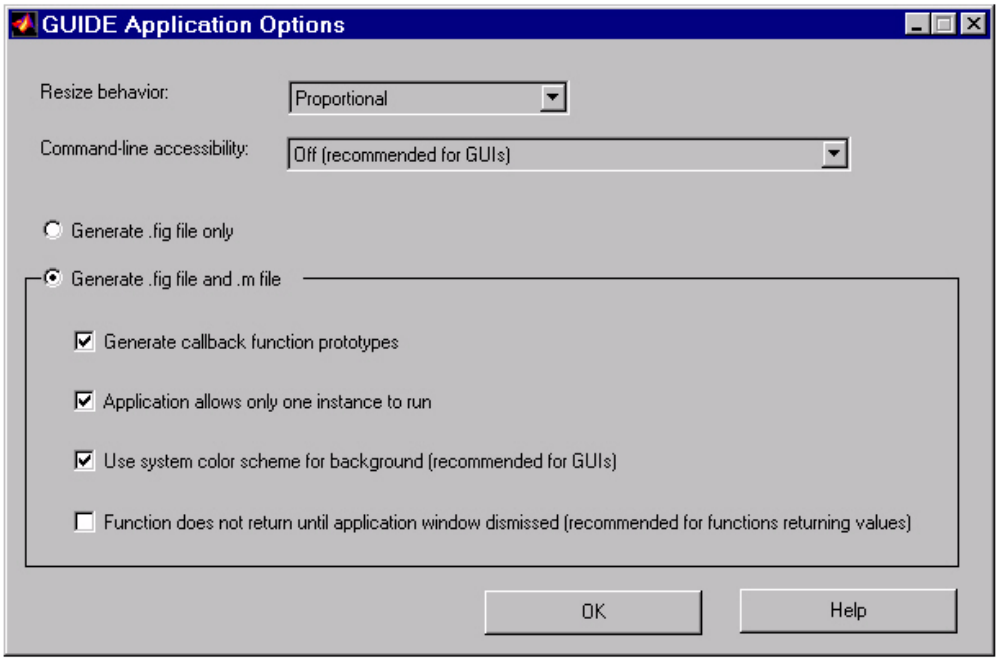


Figura 5

Configuración Estándar de la Aplicación:

Sobre las propiedades del elemento Figure:
Aplicattion Options: Al seleccionar esta opción se abrirá la siguiente ventana:



Se debe configurar como:
Resize Behaivor: User Specified
Command Line Accesibillity=on
Generate .fig and .m files
Generate Cllaback Prototype Functions
Applications allows only one instance to run
Use system color for background

Property Inspector:
Para hacer una aplicación que ocupe toda la pantalla y que nos deje ver la barra de tareas de Windows debemos seleccionar:
Position : x= 0.0
y=2.5
width= 205.6
height=54.8

Si se desea utilizar varias graficas:
Nexplot=add
Tag: Nombre interno que se le asigna

Sobre las propiedades del elemento Axes (Grafica):

Color: Red=1
Green=1
Blue=0.75
Nextplot=Replace Children
Tag: nombre interno que se le asigna .

Sobre las propiedades del elemento Edit (Cuadro de Edición):

BackgroundColor: Red=1
Green=1
Blue=0.75.
FontAngle : Italic
Fontsize: 14.00
Font Name : MS Sans Serif
Fontwight :bold
ForegrounColor:
Red=0.0
Green=0.0
Blue=0.75
Tag:Nombre interno que se le asigna.

Sobre las propiedades del elemento Button (Botón):

BackGroundColor: Red=.75
Green=.75
Blue = .75
String= Es el texto que aparece dentro del botón
Tag: Nombre interno que se le asigna.

Sobre las propiedades del elemento Slider (Barra de desplazamiento):

BackGroundColor: Red=.75
Green=.75
Blue = .75
Tag: Nombre interno que se le asigna.
Slider Step: Con esta propiedad se define el valor del paso al clickear en la fecha de incremento o decremento.

Manejo de datos entre los elementos de la aplicación y el archivo .m

Todos los valores de las propiedades de los elementos (ej, color, valor, posición, string, etc.) y los valores de las variables transitorias del programa se guardan en una estructura (structura matlab: puede almacenar matrices, string, arreglos, vectores, etc.) los cuales son accedidos mediante un único y mismo puntero para todos estos. El nombre del puntero se asigna en el encabezado del archivo .m , tomando por ejemplo el programa listado anteriormente el puntero se asigna en

```
handles = guihandles(fig);
```

handles , es entonces nuestro puntero a los datos de la aplicación .
Esta definición de puntero es salvada con la siguiente instrucción

```
guidata(fig, handles);
```

guidata, es entonces la función para salvar los datos de la aplicación.
Importante: guidata es la función que guarda las variables y propiedades de los elementos en la estructura de datos de la aplicación , entonces como regla general en cada subrutina se debe escribir en la ultima línea lo siguiente,

```
guidata(gcbo,handles);
```

esto nos garantiza que cualquier cambio o asignación de propiedades o variables quede salvado. Por ejemplo si dentro de una subrutina una operación dio como resultado una variable *fi* para poder utilizarla desde el programa u otra subrutina debemos salvarla de la siguiente manera,

```
handles.fi=fi;
guidata(gcbo,handles);
```

la primera línea crea la variable *fi* a la estructura de datos de la aplicación apuntada por *handles* y la segunda graba el valor.

get y set

La transferencia u obtención de los valores de las propiedades de los elementos se realiza mediante las funciones get y set. Por ejemplo si queremos que la variable *fi* tenga el valor del slider escribimos

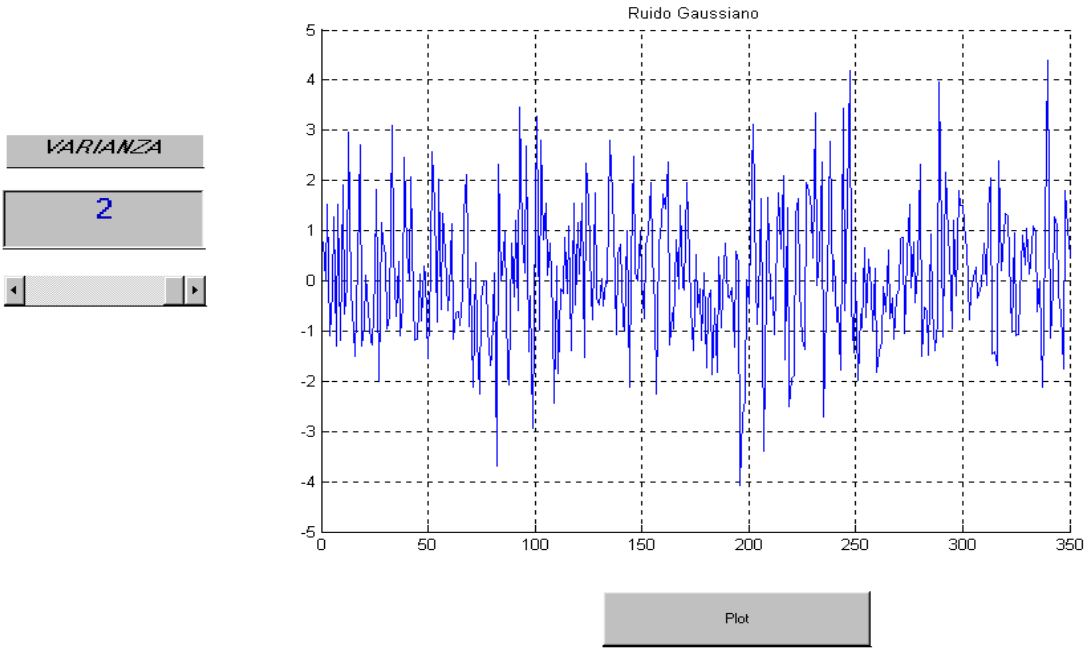
```
fi=get(handles.slider,'value');
```

Observar que siempre se obtienen los datos a través del puntero *handles* .
Ahora si quisiéramos hacer al revés y asignarle el valor la variable *fi* al slider debemos escribir

```
set(handles.slider,'value',fi);
```

Para aclarar todos los conceptos anteriores veremos una aplicación de ejemplo cuya presentación y archivo .m se listan a continuación. Lo que realiza este programa es graficar un

ruido Gaussiano (350 muestras) cuya varianza se elige con la barra de desplazamiento, el valor de esta se muestra en un cuadro de edición y luego con un botón denominado *plot* se grafica.




```
function varargout = tut(varargin)
% TUT Application M-file for tut.fig
%    FIG = TUT launch tut GUI.
%    TUT('callback_name', ...) invoke the named callback.

if nargin == 0  % LAUNCH GUI

        fig = openfig(mfilename,'reuse');
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));

        % Generate a structure of handles to pass to callbacks, and store it.
        handles = guihandles(fig);
        guidata(fig, handles);

        if nargout > 0
                varargout{1} = fig;
        end

elseif ischar(varargin{1})  % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:});  % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
```

inicialización ,aquí se pueden agregar otros valores de inicialización

```
% --------------------------------------------------------------------

function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
var=handles.var;
noise=randn(1,350);
noise=noise*sqrt(var);
%-------------------------------------------------------------------------------
%instrucciones para imprimir en la grafica del programa y que no se habra una ventana nueva
h=gca;
axes(h);
%instrucciones para imprimir en la grafica del programa y que no se habra una ventana nueva
%esto es valido cuando solo tenemos una grafica, por ahora esta bien
%-------------------------------------------------------------------------------
plot(noise);
title('Ruido Gaussiano');
grid on%
guidata(gcbo,handles);
% --------------------------------------------------------------------

function varargout = edit1_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------

function varargout = slider1_Callback(h, eventdata, handles, varargin)
var=get(handles.slider1,'value');
var=var*2;
set(handles.edit1,'string',num2str(var));
handles.var=var;
guidata(gcbo,handles);
%----------------------------------------------------------------
```

subrutina boton plot

subrutina slider