


Creating GUI's in Matlab

Meghan Stephens

April 2, 2007

1 GUIDE

GUIDE or the Graphical User Interface development environment is a tool for laying out and programming GUI's. A GUI uses graphics and text input to make using Matlab much more user friendly for people who are unfamiliar with it. GUI's can be created without using GUIDE but laying out the design of the window can be very time consuming. To open GUIDE click on the start button in the bottom left corner of Matlab and select START→MATLAB→GUIDE or by clicking .

2 Example GUI

This example GUI will allow the user to enter the coefficients and boundary values of a differential equation and the number of mesh points N , and then display a plot of the finite difference approximation to the equation as well as an error estimate. To start open GUIDE and select the blank GUI option. An empty GUI template opens which you can resize and add controls to, (Figure 1).

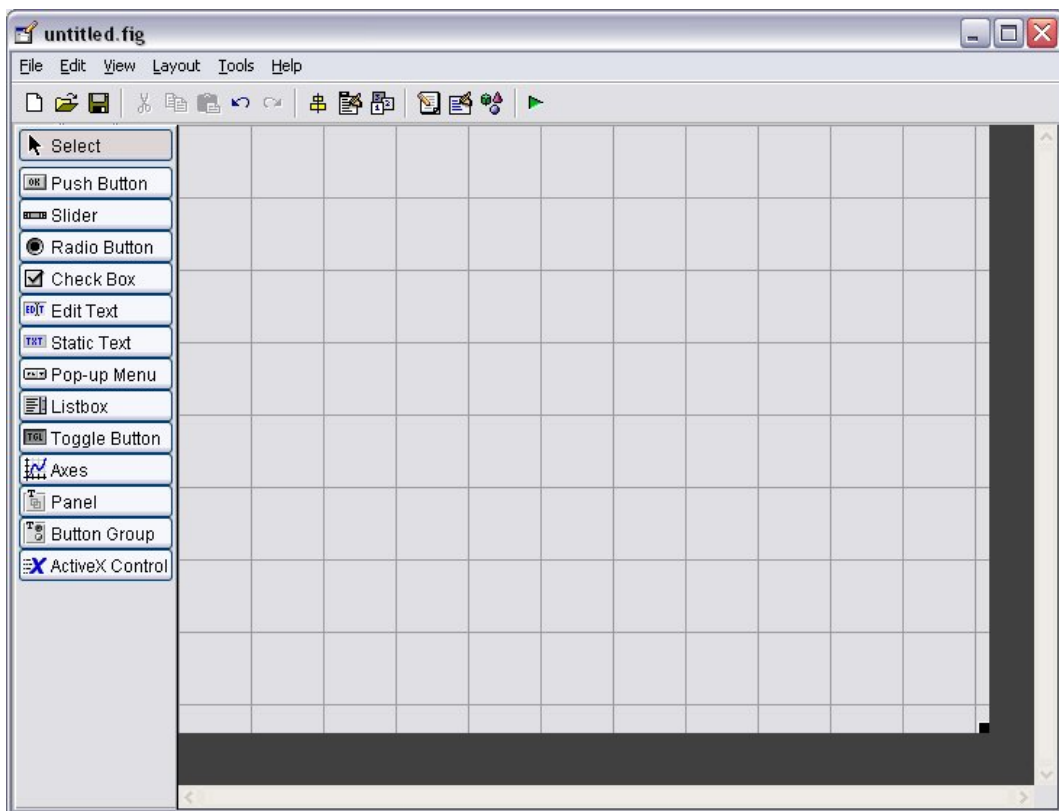



Figure 1:

Layout of the controls

To display the names of the GUI components in the component palette, select File→Preferences→ and select **Show names in component palette**.

Figures

Click on the axes button to insert a figure (Figure 2). Clicking on the blank figure window and then on the property inspector button, , opens a window displaying options which can be changed. For example scroll down to **Tag** and change this to the name which you will reference your plot with (Figure 3).

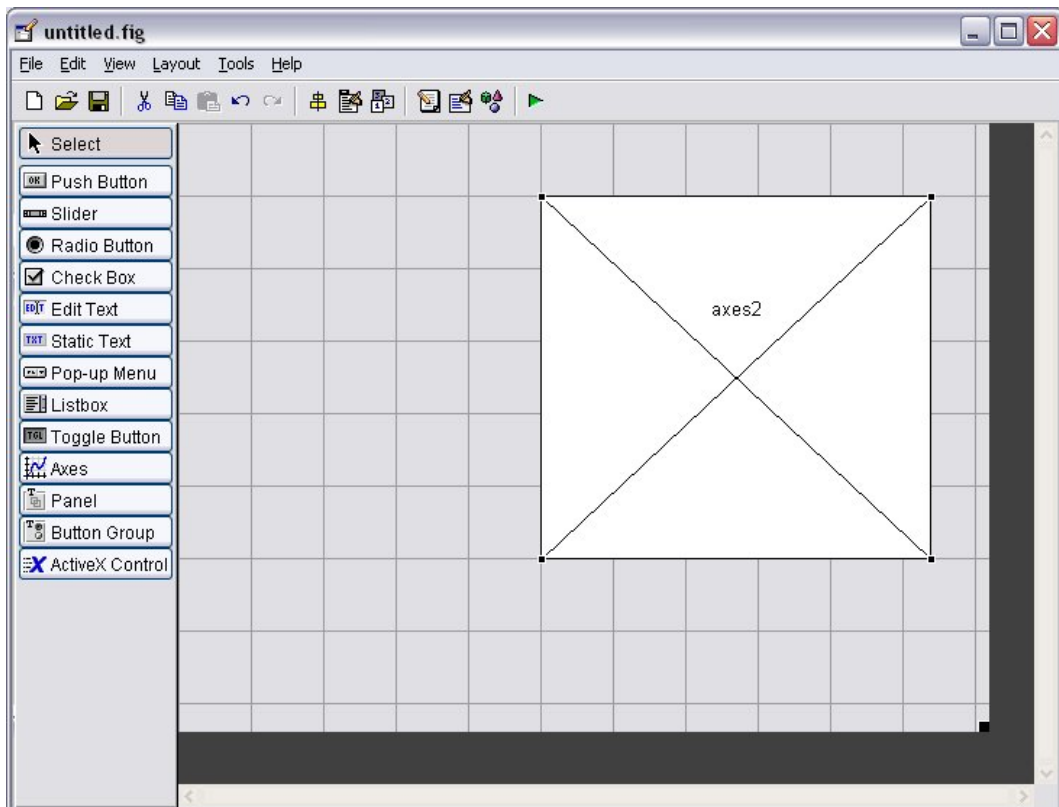


Figure 2:

Text Boxes

There are two types of text boxes, a static text box and an edit text box. To group all the controls together we can insert a panel around them using the panel button. Select the static text and insert a static text box into the GUI. The property inspector can be used to change the text that is displayed by editing the **String** option. Change this to " $u''(x) + r(x)u(x) = f(x)$ ". It is useful to change the **Tag** as well. Also insert a static text box which will display the error estimate (with a tag such as `error_text`). See Figure 4.

We have used an edit box to enter the functions $r(x)$, $f(x)$ and the boundary values $u(0)$ and $u(1)$. For example, first insert a static text box with the string " $r(x) =$ " and beside it an edit text box with a blank string. Also change the tag to a meaningful name such as `r_edit`. Repeat the process for $f(x)$, $u(0)$ and $u(1)$. See Figure 5.

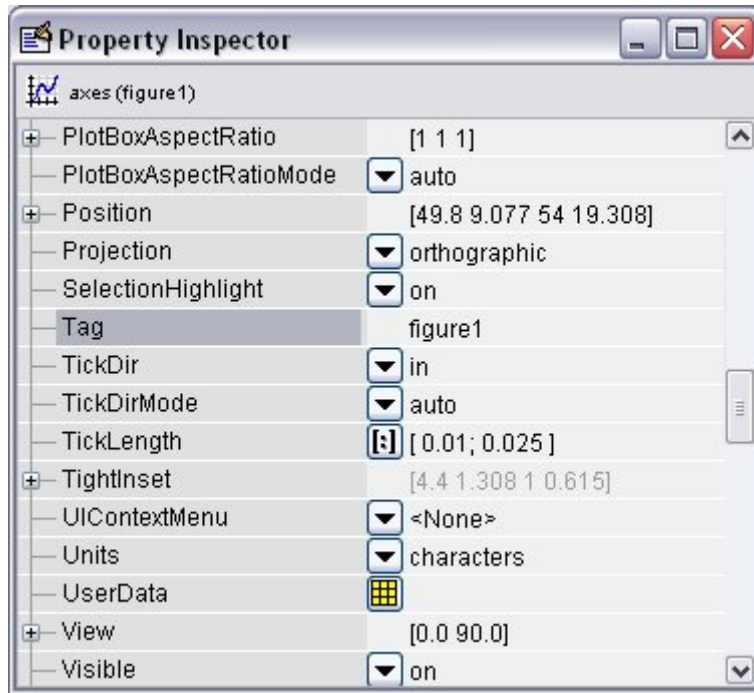


Figure 3:

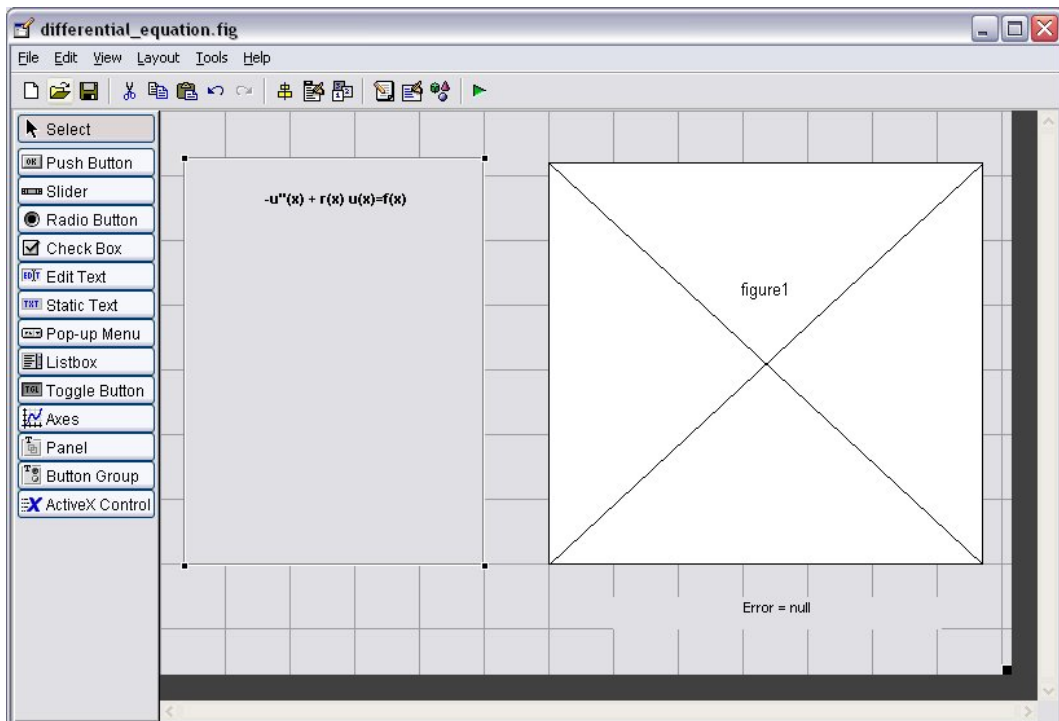


Figure 4:

Pop Up Menu

The user can select an a variable from a range of options using a pop up menu. To select N we have inserted a static text box with the string " $N =$ " beside a pop up menu, Figure 5. Change the string of the pop up menu by clicking on the button next to "string" and entering 4,8,16 etc, hitting return after each new option. Also change the tag to a name such as `N_pop_up`.



Push Button

Finally insert a push button which will execute some code when clicked. The layout should now look something like Figure 7.

Creating the GUI figure

Once all of the controls are inserted click on  in GUIDE to create the .fig and .m files. Our GUI is saved as `differential_equation`.

Programming the GUI

Clicking on the edit M-file button, , in GUIDE will open the automatically generated file which controls the actions of the GUI. You can scroll down the file to edit the functions or use the function button, , to skip to a particular function.

OpeningFcn

The opening function is the first callback in every GUI M-file. You can use it to perform tasks that need to be done before the user has access to the GUI, for example, to create data or to read data from an external source. In our example the opening function is called `differential_equation_OpeningFcn`.

At the start we want to set $N = 4$. Any data that you want to pass between functions in the GUI should be saved in the handles structure.

```
% --- Executes just before differential_equation is made visible.
function differential_equation_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 differential_equation (see VARARGIN)

handles.N = 4;

handles.x = 0:1/handles.N:1;

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

% Update handles structure
guidata(hObject, handles);
```

Note that to save any changes that you make to the handles structure, you must insert at the end of each function `guidata(hObject,handles);`

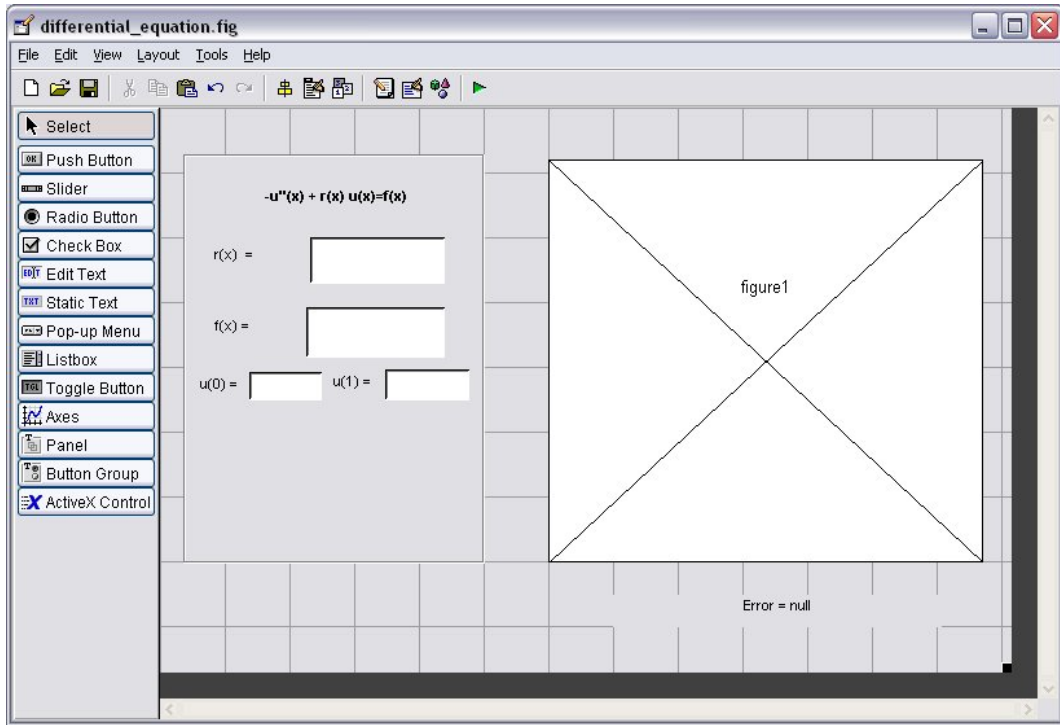


Figure 5:

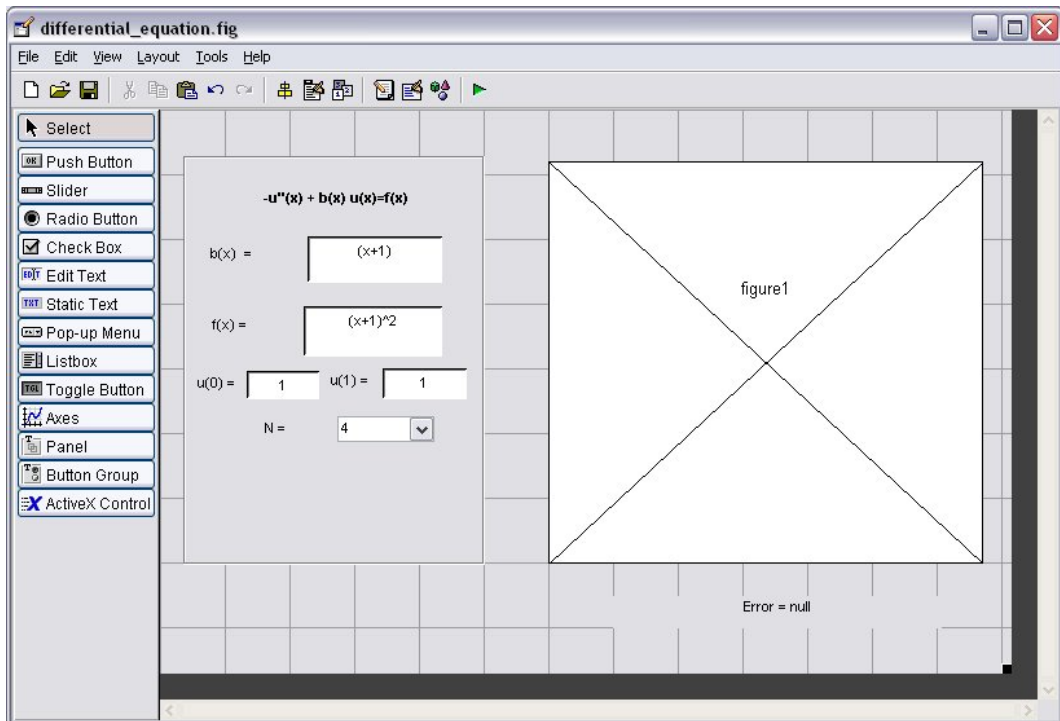


Figure 6:

Read in data from an edit text box

In order to read information from an edit box we use the command `get(hObject,'String')` inside the callback function. For example to read in the value of $u(0)$ (where the Tag used was `u0_edit`):

```
function u0_edit_Callback(hObject, eventdata, handles)
% hObject    handle to u0_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of u0_edit as text
%        str2double(get(hObject,'String')) returns contents of u0_edit as a double

handles.u0 = str2double(get(hObject,'String'));

% Update handles structure
guidata(hObject, handles);
```

To read in the values for the functions $r(x)$ and $f(x)$ we must convert the string to an inline function using

```
handles.f = inline( get(hObject,'String'), 'x');
```

2.0.1 Read in data from a Pop Up menu

In order to read in the value of N which is selected in the pop up menu and then create a mesh add this code to the callback

```
% --- Executes on selection change in popupmenu1.
function N_popup_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1

val = get(hObject,'Value');
str = get(hObject, 'String');

handles.N = str2double(str{val});
handles.x = 0:1/handles.N:1;

guidata(hObject,handles)
```

2.0.2 Submit button

In order to calculate the finite difference approximation using your own function and then plot the outcome include the following into your submit button callback.

```

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

% use the finite difference method
handles.u = fd(handles.r, handles.f, handles.u0,
handles.u1,handles.x);

plot(handles.x, handles.u);

% estimate the error by comparing to the finite difference
% solution using double the mesh points
x2 = 0:1/(2*handles.N):1; u2 = fd(handles.r, handles.f, handles.u0,
handles.u1, x2); u_interp = interp1( handles.x,handles.u, x2 );
error = max(abs( u_interp - u2' ));

% display the error
set(handles.error_text, 'String', strcat('error =
',num2str(error)));

% Update handles structure
guidata(hObject, handles);

```

When all the necessary callback functions have been edited save the M-file and run the GUI using the green button. You can also run the GUI by typing `differential.equation` in the command line of Matlab.

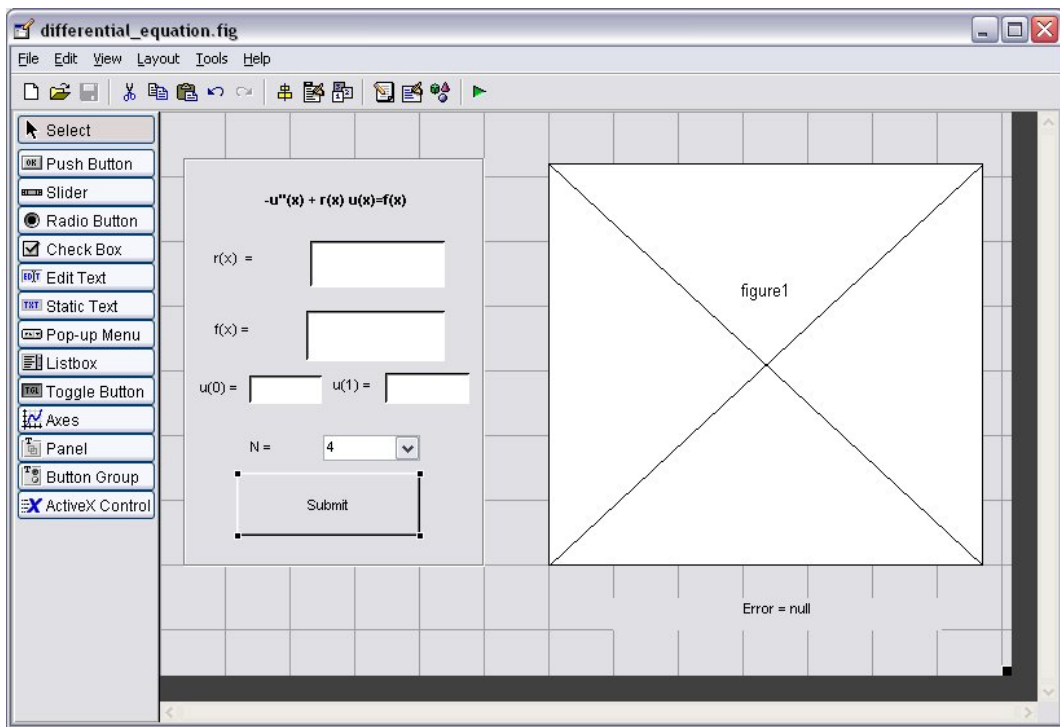


Figure 7: