INTRODUCTION TO MATLAB INTERACTIVE GRAPHICS

EXERCISES

Eric Peasley, Department of Engineering Science, University of Oxford
version 3.0, 2017
MATLAB Interactive Graphics Exercises

In these exercises you will be building your own Graphical User Interface (GUI).

Before starting any GUI, it is a good idea to make a rough sketch of the GUI.

The purpose of the GUI is to plot the MATLAB expression entered into the Edit Text at the top of the figure. The graph will be plotted between two values in the Edit texts in the Panel in the bottom right. The Check box will toggle the grid on and off. The Pop-up Menu will be used select the colour of the graph.

The first exercise will produce the minimum required to get the GUI working. The following exercises will add more controls and extra functionality.
Exercise 1 (The Basic Program)

In this exercise you will create your GUI and start laying out the graphical objects and controls. You will also automatically generate the MATLAB program and edit it to produce a basic function plotter.

Start Guide

Enter `guide` in the command window.

Select Blank GUI

OK

Guide Preferences

Form the menu bar select

File
Preferences

Make sure that Show names in components palette is ticked. Without this set, the components palette on the left hand side will only show the icons.

Click on OK to close the preferences.

Form the menu bar select

Tools
Grids and Rulers

Make sure that Show rulers is ticked. This will add rulers to the top and left hand side of the figure that can help align objects in the figure.

Click on OK to close the Grids and Rulers window.

Setting the Figure Size

Maximize the guide window.

The grey squared object is the figure. Drag the bottom right hand corner of the figure so that it almost fills all the available space.

Reference Lines

Put the mouse on the top ruler and drag down about 1½ squares.
You should see a blue reference line.

This will be used to align the objects.
Adding a Static Text Object.

On the left of the screen is the component palette. Drag a static text from the palette onto figure. Notice the the text will snap to the grid or the blue reference line. Place the text on top of the blue reference line to the left of the figure. See drawing on page 1.

Double click on the static text to open the property inspector.

_The property inspector is used to inspect and change properties of a particular object._ You can open the property inspector in several ways.

- **Double clicking on an object.**
- **Right clicking on an object and then selecting property inspector.**
- **Selecting the object and then clicking on the property inspector icon on the toolbar.**
- **Double clicking on the object in the object browser.**

You can open the object browser by clicking on the object browser icon on the toolbar. This can be useful when it is difficult to select overlaying objects on the screen.

To obtain help on a property, right click on a property in the inspector and select _what's This?_

Change the **FontSize** property to **16pt**

Click on the icon next to the **String** property. Change the string to **Function to Plot**.

Close the property inspector.

Re-size the static text by dragging the top right hand corner so that you can see all the text.
Adding a Edit Text

To the right of the static text, on the blue reference line, add an **Edit Text**. Enlarge the edit text so that it is at least as large as the static text in both width and height.
Change the following properties.

<table>
<thead>
<tr>
<th>FontSize</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>sin(x)</td>
</tr>
<tr>
<td>Tag</td>
<td>editfunc</td>
</tr>
</tbody>
</table>

The tag is used to name the callback functions in the MATLAB program. When the figure is saved, the program will automatically be created with two callback functions called editfunc_CreateFcn and editfunc_Callback. **editfunc_CreateFcn** is run when the object is created and **editfunc_Callback** is run when text is entered into the edit text. For this reason I recommend changing the tag to something meaningful so that it is easier to know which function is associated with which object.

The program will generate a structure containing all the handles of all the objects. **handles.editfunc** will be the handle of the edit text.

Adding the Axes.

Add a Axes below the edit text. Adjust the size so that each side is about one square in from the edge and the base is about 3 squares from the bottom.

Note that the default tag for the axes is axes1.

Saving and Creating the MATLAB program.

From the menu bar select

File
  Save as

Save the figure as **funcplot**.

The figure will be saved as funcplot.fig. At the same time, the MATLAB program will be created and opened in the MATLAB editor. It will already contain several functions including editfunc_Create and editfunc_Callback.
Editing the MATLAB program

At the bottom of the program, add the following function.

```matlab
function replot(handles)
% Function to replot the graph

% Produce the x value
x = linspace(-5,5,1000);

% Read in string from the Edit Text
s = get(handles.editfunc,'String');

% Evaluate the string as a MATLAB expression
y = eval(s);

% Set the current axes to axes1 on the gui
axes(handles.axes1);

% Plot the graph
plot(x,y)
```

On the MATLAB editor toolbar, in the Navigate section, you will find Go To. Click on this and select editfunc_Callback. This is the callback function for the edit text. At the bottom of this function, add the follow line.

```matlab
replot(handles)
```

Save and run the program.

Try entering you own MATLAB expressions.

The Opening Function

*If you restart the program, you will find that the default expression at start time is not plotted. You can change this by calling replot in the opening function.*

Call `replot` in the function `funcplot_OpeningFcn`, before the `guidata(hObject,handles)`.

Test the program.
Exercise 2 (Expression Error Recovery)

Enter an erroneous expression into the function to be plotted.  
*For example, x^2 will not work because a dot is needed after the x.*

At the moment it is not obvious that an error has occurred.  
What would be better is to detect the error and report what is going wrong.

Minimize, but don't close the GUI.

In the function editfunc_Callback, replace

```matlab
replot(handles)
```

with

```matlab
try
    replot(handles);
catch err
    errordlg(err.message,'Expression Error');
end
```

Save the program.

Click on the GUI on the banner at the bottom of the screen.

Enter the expression again to see the error dialogue. The program should not crash this time.

*Notice that the GUI does not have to be shut down to take advantage of the new code.*

Close down the GUI
Exercise 3 (The Grid Check Box)
In this exercise, you will add a check box to toggle the grid on and off.

Editing the Figure
Return to the guide window. Generate another reference line about 1½ square from the bottom of the figure. Remember, drag down from the top ruler.

Drag a check box onto the figure so that the base is on the reference line and the left hand side is approximately in line with the left hand edge of axes1.

Change the following properties of the check box.

<table>
<thead>
<tr>
<th>String</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid</td>
<td>checkboxGrid</td>
</tr>
</tbody>
</table>

Save the figure.

Editing the Program
Notice the the callback function for the check box has been added to the program. Add the following to the end of this function.

```
function checkgrid(handles)
% Turn the grid on or off as required.
% Set the current axes
axes(handles.axes1);

% Read the state of the grid check box.
status = get(handles.checkboxGrid,'Value');

if  (status)
   grid on
else
   grid off
end
```

Run the program and test the check box.

Question 1
Leave the grid on and change the MATLAB expression.
What is the problem and how do you fix it?

*The answer is on page 17*
Exercise 4 (The Colour Pop-up Menu)
In this exercise you will add a pop-up menu to change the colour of the line.

Editing the Figure
Close the program and go back to the the guide window. Drag a pop-up menu onto the figure to the right of the grid check box.

Note that the tag for the pop-up menu will be popupmenu1. As there is only one pop-up menu, there is less likelihood of confusion.

Save the figure.

Editing the Program
This time we are going to use the create function to produce the menu. Add the following to the end of popupmenu1_CreateFcn.

```matlab
% Create the cell of the menu strung
s{1} = 'Red';
s{2} = 'Green';
s{3} = 'Blue';

% send the menu strings to the pop-up
set(hObject,'String',s);
```

*hObject is an argument passed to the function that contains the handle of the object itself.*

Edit replot and replace the plot line with the following

```matlab
% Read the state of the colour pop-up menu
v = get(handles.popupmenu1,'Value');

% Set the colour
switch(v)
    case 1
        col = 'r';
    case 2
        col = 'g';
    case 3
        col = 'b';
end

% Plot the graph
plot(x,y,col)
```
Save and run the program. Select a different colour using the pop-up menu. **Don’t worry if the colour does not change straight away.** At this stage the colour will only change after a new expression has been entered. So enter a new expression.

**Question 2**

The reason that the colour only changes when you enter a new expression is that replot only runs at the start of the program and in the callback function of the edit text. It does not run when you select a new colour with the pop-up menu.

What can you do to force the graph to plot when you select a new colour?

*The answer is on page 17*
Exercise 5 (X Axis Limits)

In this exercise you are going to add a panel with two text edits that specify the limits of the X axis.

Editing the Figure

Close funcplot and return to guide. Drag a Panel onto the figure. Resize the Panel so that it is approximately 4 squares wide by 3 squares high. Don't worry about the position of the Panel yet.

Change the properties of the panel to

| Title | limits |

Drag a edit text into the bottom right hand corner of the panel. Change the following properties.

| String | 5 |
| tag | editmax |

Add a static text above the edit. Change the properties to

| String | max |

Move the static text so that it sits just above the edit text without touching it. Select both the static text and the edit by click on one, then holding down control and click on the other.

Then click on the align icon on the icon bar.

In the box marked Horizontal, to the right of Align, click on the icon to align the centres.

Then click on OK

With both objects still selected, right click on top of them and select Duplicate. Using the cursor keys, move the objects so they are to the left of the original pair and in line with them. Change the properties of the new edit text to

| String | -5 |
| tag | editmin |

Change the properties of the new static text to

| String | min |

Now select all four objects in the Panel (using the control key) and drag them into the top left hand corner of the Panel. Then drag the bottom right hand corner of the panel in so that the four object are nicely position in the panel. Then drag the panel so that it is below the bottom right hand corner of axes1.
Run the program by hitting the **run figure** icon, which is the **green right arrow** on the icon bar.

Check that the Panel does not obscure the X axis. Correct if required.

**Editing the Program**

Edit the function `replot` and replace the line with the `linspace` with the following.

```matlab
% Get the limits of the X axis
mxs = get(handles.editmax,'String');
mx = eval(mxs);

mns = get(handles.editmin,'String');
mn = eval(mns);

% Produce the x values
x = linspace(mn,mx,1000);
checkgrid(handles)
```

Save and run the program.
Set the limits and enter an expression.

**Question 3**

What can you do to force the graph to plot when you select a new limit?

**Question 4**

Try entering a limit of 2*pi.
What do you do to allow the program to work with pi?

*The answers are on page 17*
Optional

The last three exercises are optional as there may not be time to complete them within the seminar. If you want to try them on your own computer, then email funcplot.m and funcplot.fig to yourself. Exercise 6 and 7 each take about 20 mins. Exercise 8 takes about 5 mins.

Exercise 6 (Adding an Icon Bar)

At the moment, your GUI has no icon bar. This exercise explores various different ways of configuring an icon bar.

Using the default figure icon bar

In Guide, double click on the figure away from any objects. This will give you the figure properties.

Set the Toolbar property to figure.

Click on the run figure icon.  

Check that the zoom in icon works as expected.

Customizing the icon bar.

Not all the icons are appropriate for our GUI. We are now going select which icons we want.

Close the program.

Set the Toolbar property back to auto.

From the menu bar of guide, select

  Tools
  Toolbar Editor

At the top of the editor you will see a bar with Add tools here to create toolbar. Drag the Zoom In, Zoom Out, Pan and Data Cursor icons onto this bar.

Click on OK

Click on the run figure icon and check out the icons.
Adding your own icons.
We are going to add our own icon to toggle the grid on and off.

Open the Toolbar Editor.

Drag a Toggle Tool onto the toolbar. Then select the Toggle Tool

In the Tool Properties area on the right change

**Tag**   Griduitoggletool
**Tick**   Separator on the left.

Then click on the Edit button.

Design your own icon. When you have finished click OK on the icon editor.

Back in the Toolbar Editor, in the Tool Properties, click on the View button to the right of Clicked Callback.

The figure is saved, the guide window is minimized, a call back function is created and the MATLAB editor will by open at the new call back function.

Add the following to this function.

```matlab
%Read current status of the grid check box
state = get(handles.checkboxGrid,'Value');

%Invert the value
state = ~state;

%Change the state of the grid check box
set(handles.checkboxGrid,'Value',state);

%Update the grid
checkgrid(handles);
```

Save and test the program
Exercise 7 (Adding a Menu Bar)

It is possible to use the standard figure menu bar by setting the figure property `MenuBar` to `Figure`. However, in this exercise you will produce your own menu bar.

Creating the Menu Bar

From the menu bar of guide, select

```
Tools
  Menu Editor
```

Click on the New Menu icon.

Click on the new menu **Untitled 1**

Change the Menu properties on the right so that

```
Text  Tag  File  Tag
File  FileMenu
```

With the new menu still selected, click on the New Menu Item icon twice.

Click on each menu item in turn and change their properties

```
Top
Text  Tag  File  Tag
Load  LoadMenu
```

```
Bottom
Text  Tag  File  Tag
Save  SaveMenu
```

Click on **OK** and save the figure.
Programming the Menu Callbacks

At the bottom of the program you will find a callback function for each menu. Add the following to the save function.

```matlab
% Build a structure with the program details
FuncData.Expression = get(handles.editfunc, 'String');
FuncData.Max = get(handles.editmax, 'String');
FuncData.Min = get(handles.editmin, 'String');
FuncData.Grid = get(handles.checkboxGrid, 'Value');

% Predefined dialogue to save variable
uisave('FuncData');
```

For the load, we do the reverse with a check for the correct file type.

```matlab
% Predefined dialogue to load variable
uiopen('FuncData');

if (exist('FuncData', 'var'))
    set(handles.editfunc, 'String', FuncData.Expression);
    set(handles.editmax, 'String', FuncData.Max);
    set(handles.editmin, 'String', FuncData.Min);
    set(handles.checkboxGrid, 'Value', FuncData.Grid);

    % Plot the loaded function
    replot(handles)
else
    % Error dialogue
    errordlg('Incorrect Data')
end
```

Save and test.
Exercise 8 (Packaging a MATLAB App)

If you want somebody else to use your program, you could send them the .m and .fig files. Another simple way of distributing your program is to package the program into a MATLAB app. You can then send this package to somebody else so that they can install the app into MATLAB and use your program.

On the banner at the top of the MATLAB window, click on the APPS tab.

Then click on the Package App icon.

In the top left hand box, select Add main file.

Select funcplot.m.

Under Describe your app, the name funcplot has been inserted. Click on the icon next to the name and select an icon for funcplot.

Hit the Package button on the right. Wait for it to say Packaging Complete.

Close the Package App window.

You will see a number a new files have appeared.

funcplot.prj The project file that contains the information that you have entered in to the Package App window. Clicking on this file will reopen the project in the Package App window.

funcplot.mlappinstall Is the file to install the app into MATLAB. This is the file that you send to users that want to use the app.

To install the app into MATLAB, double click file funcplot.mlappinstall.

To run the app, click on the down arrow on the right of the Apps banner at the top of MATLAB.

Under MY APPS, click on funcplot.

To remove the app, right click on the app and select Uninstall.
Answers

Question 1
The problem is that the grid disappears when you enter a new expression. By default the grid is off when you plot a new graph.

You can fix the problem by running the checkgrid function after the graph has been plotted so that the grid is turned on if required. At the very end of function replot, after the plot command add

    checkgrid(handles);

Question 2
You call the function replot in the pop-up menu callback function, popupmenu1_Callback. So that when you select a new colour, the expression will be plotted again, this time picking up the new colour. Add

    replot(handles);

to the very end of the function popupmenu1_Callback.

Question 3
Call replot function in the callback function for editmin and editmax edit texts.

Question 4
Replace the str2double function calls with an eval function call.

version 3.0