Graphical User Interfaces (GUIs)

- A User Interface (UI) is the way a person interacts with a program (for example, to exchange information). The user supplies input and the program supplies output (based on an analysis of the input).
- The program displays text, graphics, sound on a computer.
- The user uses the keyboard, mouse, microphone, etc to interact with the program.
- A Graphical User Interface (GUI), pronounced "gooey", is a user interface that uses graphical objects such as windows, icons, buttons, menus and text. Most commonly, the user uses a mouse to control the move-

ment of the cursor on the computer screen. Pressing the mouse button signals action selection, etc.

- We show the use of predefined MatLab dialog boxes and the use of Handle Graphics *uicontrol*, *uimenu*, *uicontextmenu* and *uitoolbar* objects to add graphical user interfaces to MatLab programs.

- *uimenu* objects create drop down menus in Figure windows.

- *uicontrol* objects are used to create buttons, sliders, popup menus and text boxes.

- *uitoolbar* objects contain pushbutton (*uipushtool*) objects and toggle button (*uitoggletool*) objects.

- MatLab has a online tutorial to show how to build a GUI. Simply type `demo` in a MatLab command window, chose "Creating Graphical User Interfaces" and then select "Creating a GUI with Guide" and listen to a 10 minute tutorial. GUIDE (Graphical User Interface Design Environment) is a GUI tool that allows you to interactively build GUIs quickly and easily. However, the user still needs knowledge of all the nuances of building a good GUI. For example, the user must understand Handle Graphics properties such as 'HitTest', 'Interruptible' and 'BusyAction' as well as button-clicking sequencing, the event queue and callbacks.

- GUIs features and details are so extensive that entire books are devoted to them. Some MatLab GUI references include:

    1. "Graphics and GUIs with MatLab", 3rd edition, Patrick Marchand

and O. Thomas Holland, Chapman and Hall/CRC, 2003.

2. "MATLAB Advanced GUI Development", Scott T. Smith, Dog Ear Publishing, 2006.

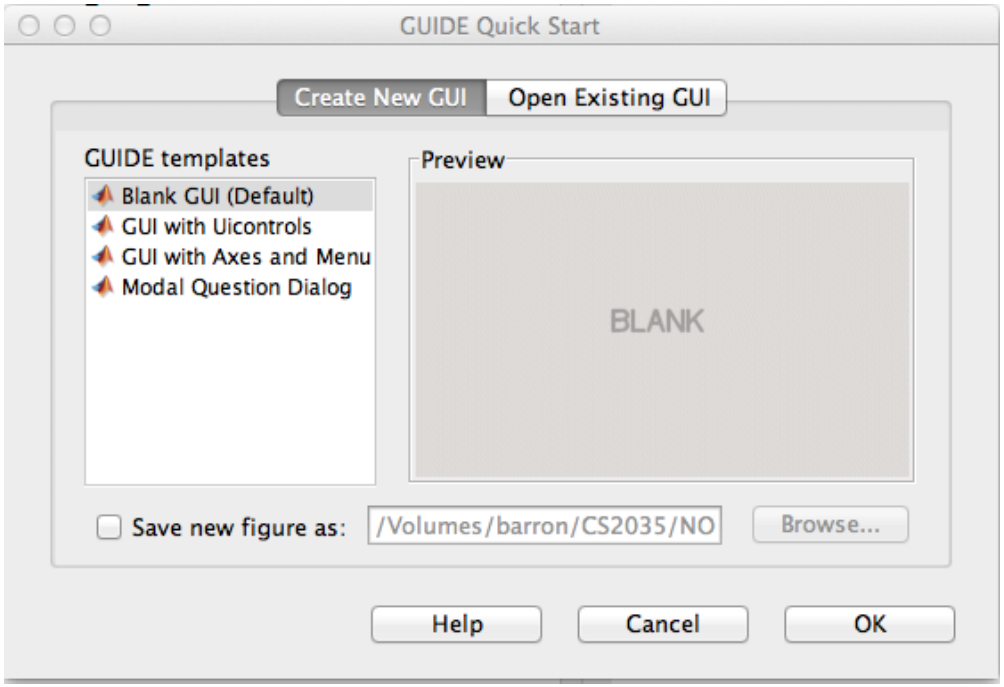
- In this lecture, we will keep things simple, and just introduce GUIDE.

GUIDE - Example

- We'll implement a modified version of the GUI described in the MatLab GUI GUIDE video (see above). This GUI lets you plot one of: the **peaks**, the **membrane** or the **sinc** functions (actually the latter is called a **jinc** function in 3D) as a **surf**, a **mesh** or a **ribbons** plot. [In the MatLab demo GUI the **ribbons** plot replaces a **contour** plot.] The surface is plotted in a window on the GUI. Some buttons that allow manipulation of these objects (3D rotation, zooming, pan) and a data cursor to examine 3D coordinates) are included at the top of the GUI.

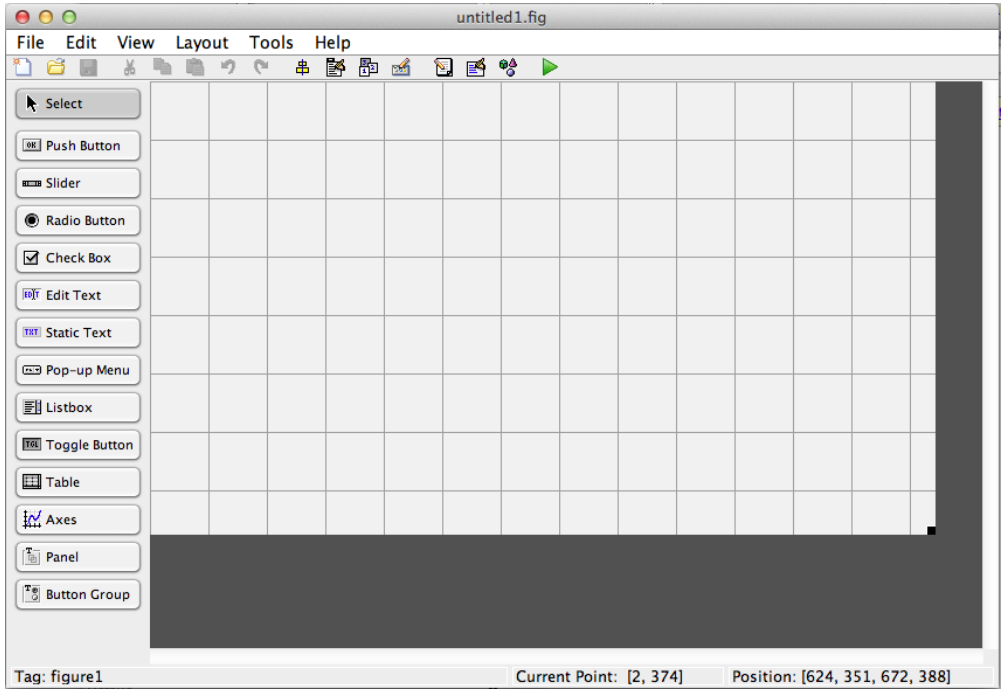
- Open MatLab in the right directory and type "guide" in the command prompt window. [Or select "new" on the home tab and then "Graphical

User Interface".] The following window pops up:



GUIDE quick start window.

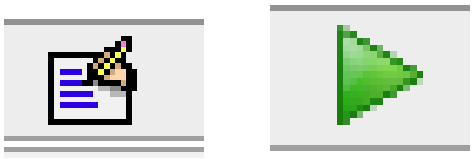
Select the first option "Blank GUI (default)" and click "OK'. The following window appears:



Default GUI GUIDE window (drag on the lower right corner to make it a bit bigger).

- If the names don't appear in the lefthand list, select "file/preferences" and then tick "Show names in component palette" in the GUIDE preferences window. It is better to have the names displayed as the icons may not be meaningful to the beginner.

- Advisable: select the "Inspector Properties" button (2^{nd} button from the left of the green arrow). Set units to pixels (corresponds to the units on

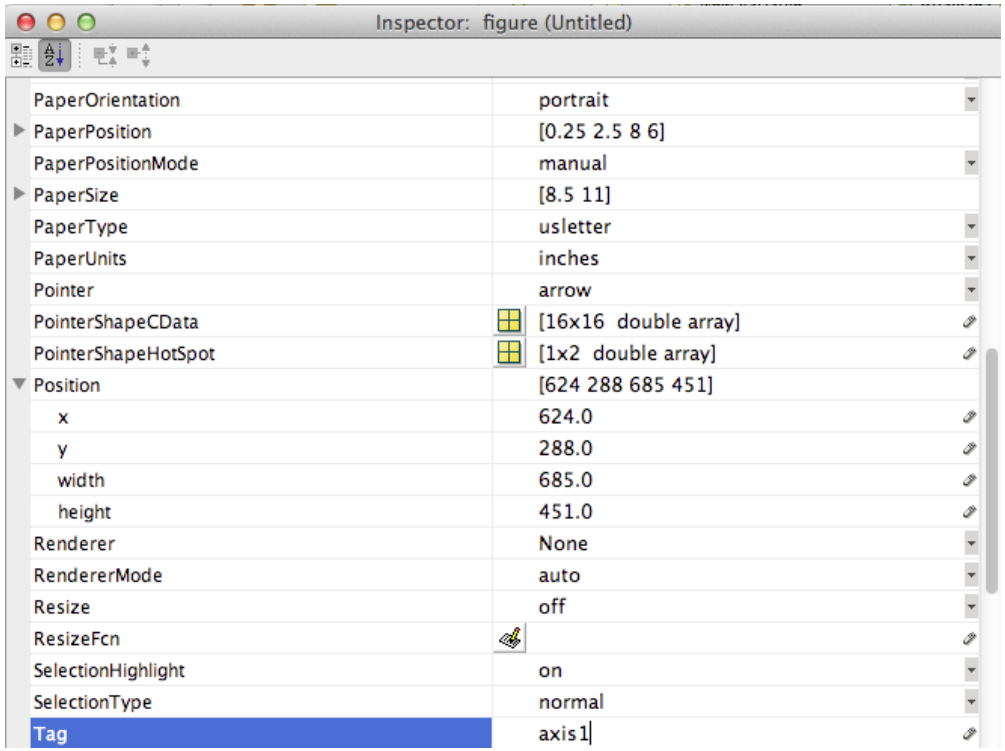


(a) (b)

(a) The property inspector button (a) is the 2^{nd} buttom left of the green arrow shown in (b). This latter button creates the m and fig files for the GUI when clicked.

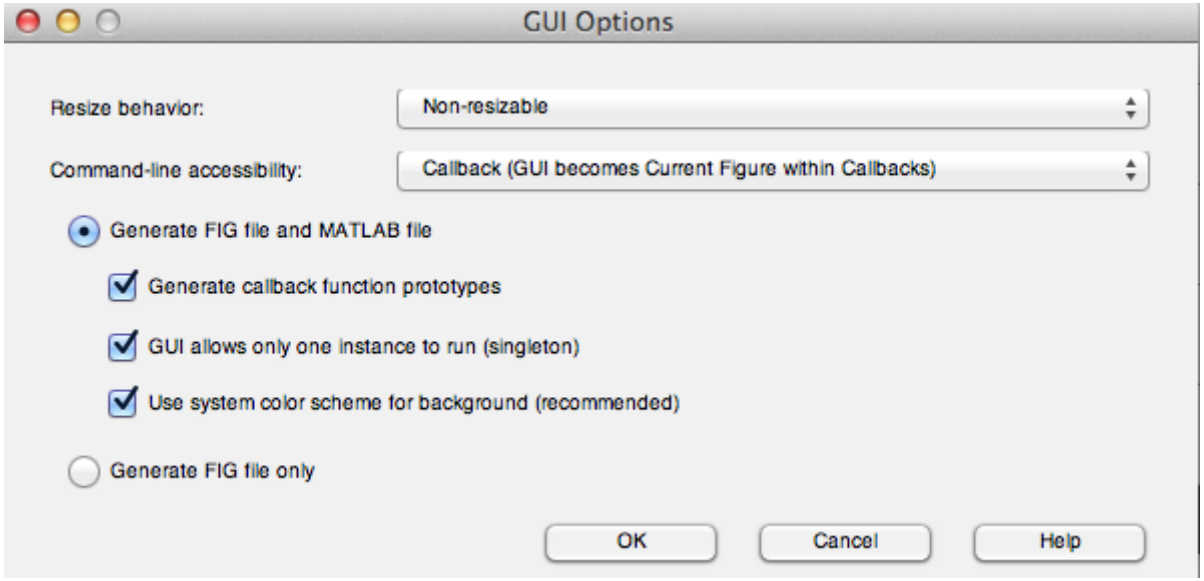
your screen. Left double click on the "position" property and fields "x",

"y", "width" and "height" will be visible and available to change. To change, type new numbers in pixels or just drag the lower right point of screen layout. Currently $x = 624.0$, $y = 288.0$, $width = 685.0$ and $height = 451.0$. The figure below shows a screen shot of the property inspector menu:



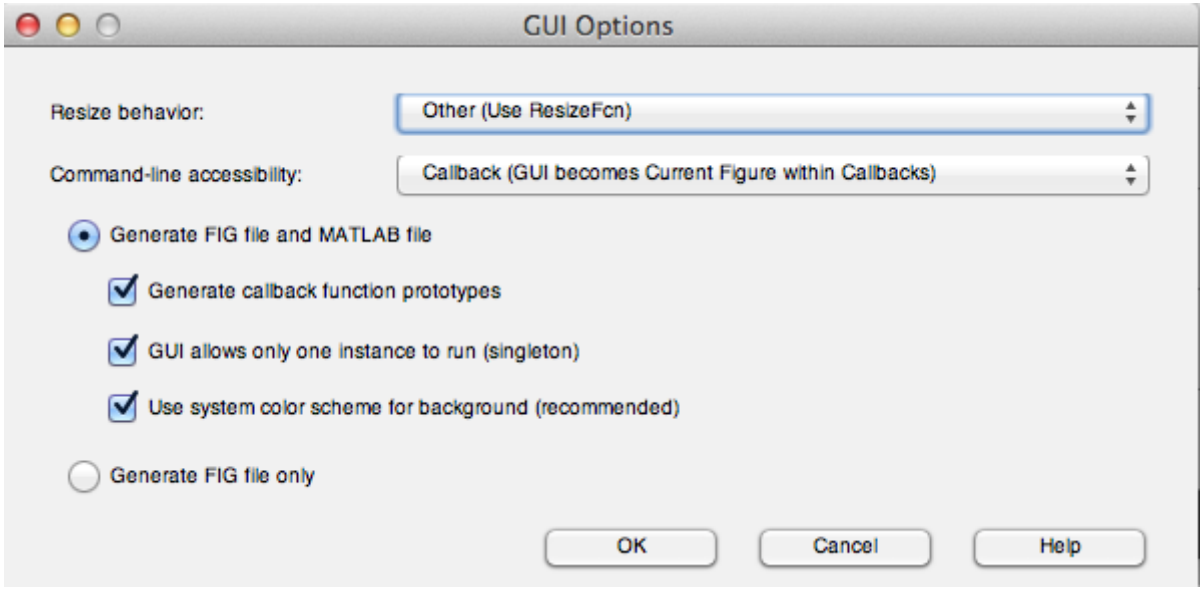
The Properties Inspector menu: note that the tag field for the plotting area has value "axis1" (we don't change this) and the position field has been left double clicked to yield the x, y, width, height fields in pixels. Of course, the Units property was first set to pixels from characters.

- Initially, the layout window will be of fixed size, We can select "Tools" and then "GUI Options" to get: Select "Other (ResizeFcn)" as shown



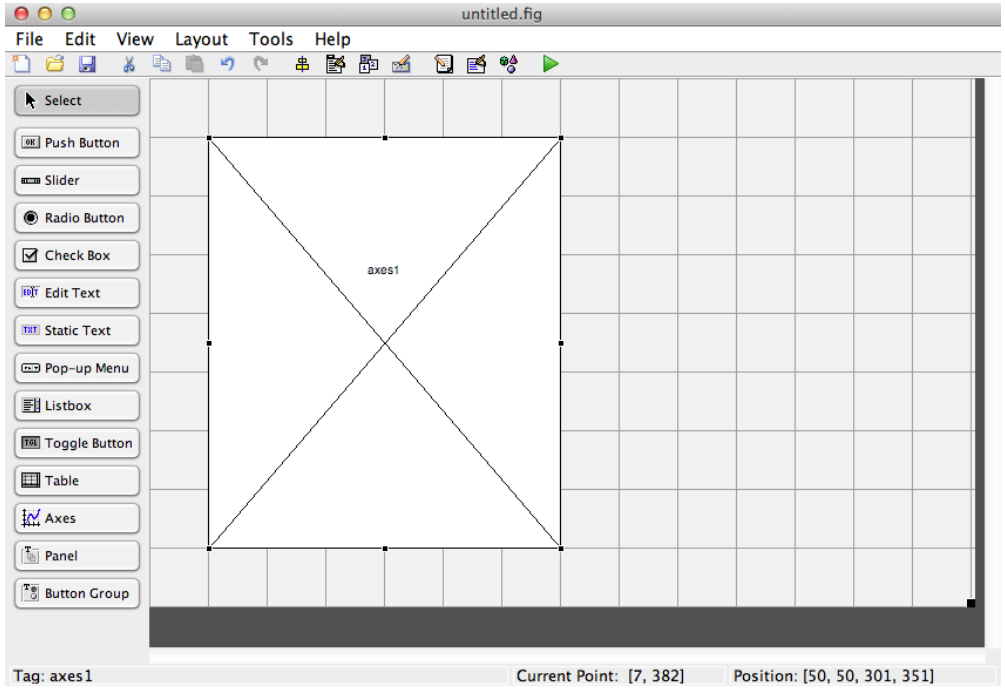
Resizeable window.

below to get a resizeable GUI:



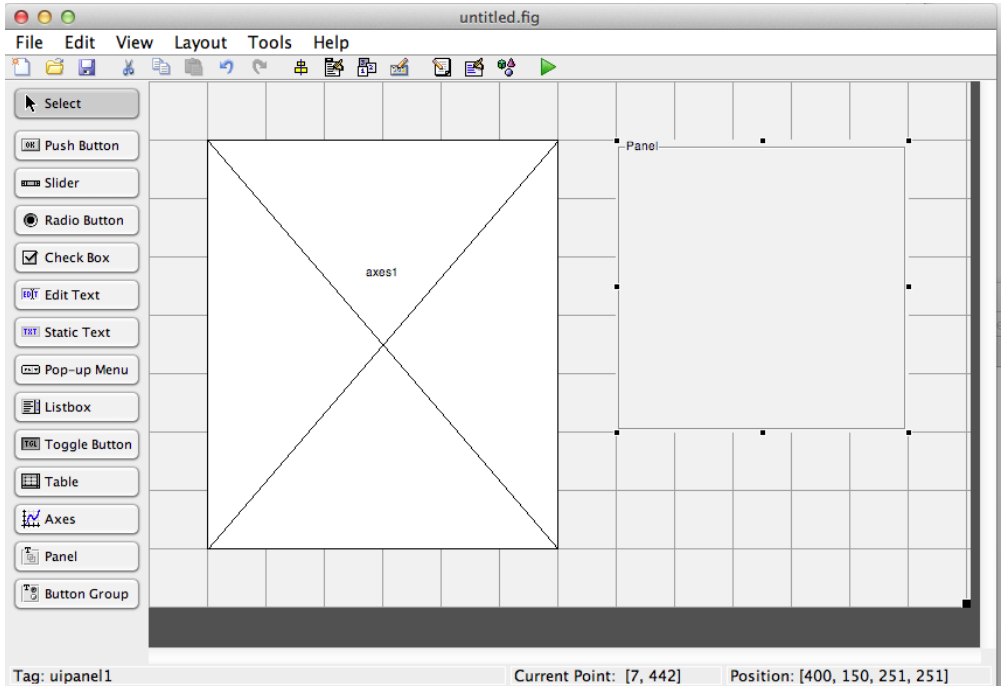
Now the GUI is resizeable.

- We will add an axis object to the layout, by selecting "axis" from the buttons on the left. We get:



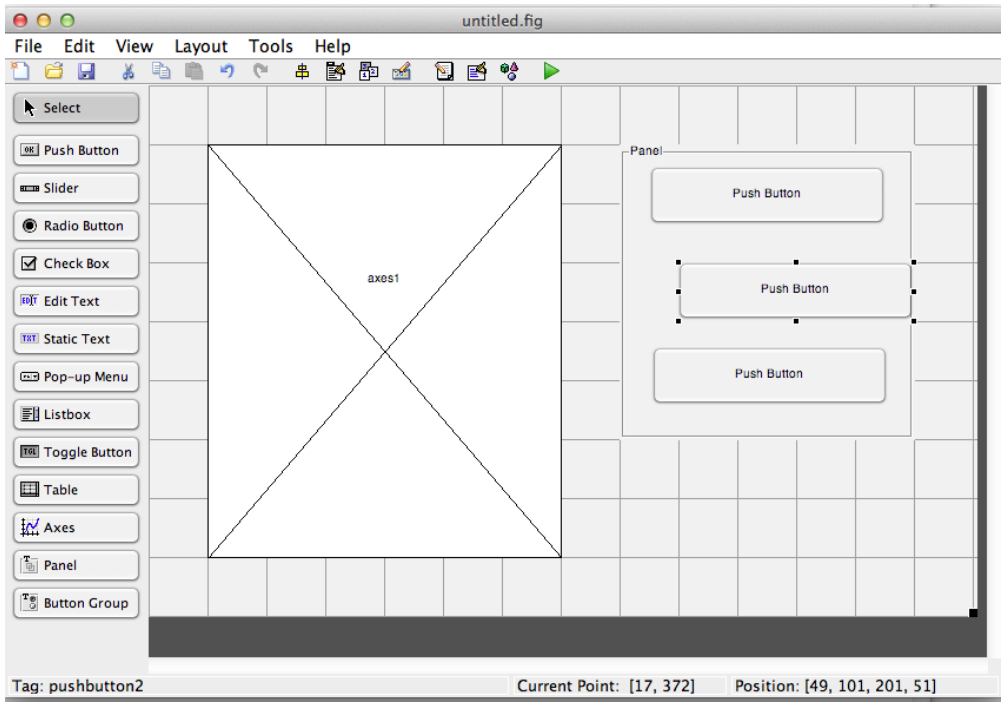
GUI layout with an axis object added.

- Next, we add a **panel** to contain our buttons. This makes our buttons manipulatable as a group. The result is shown below:



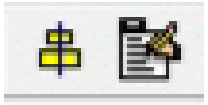
GUI layout with panel added.

- Next, we add some **pushbuttons** to allow execution of the 3 graph types. We add one pushbutton to the upper part of the panel. Then we duplicate this button twice by twice clicking on the right mouse button and selecting "Duplicate". We then roughly place these second to pushbuttons in their correct positions in the panel (select each button with a left click and drag). At this point we have:



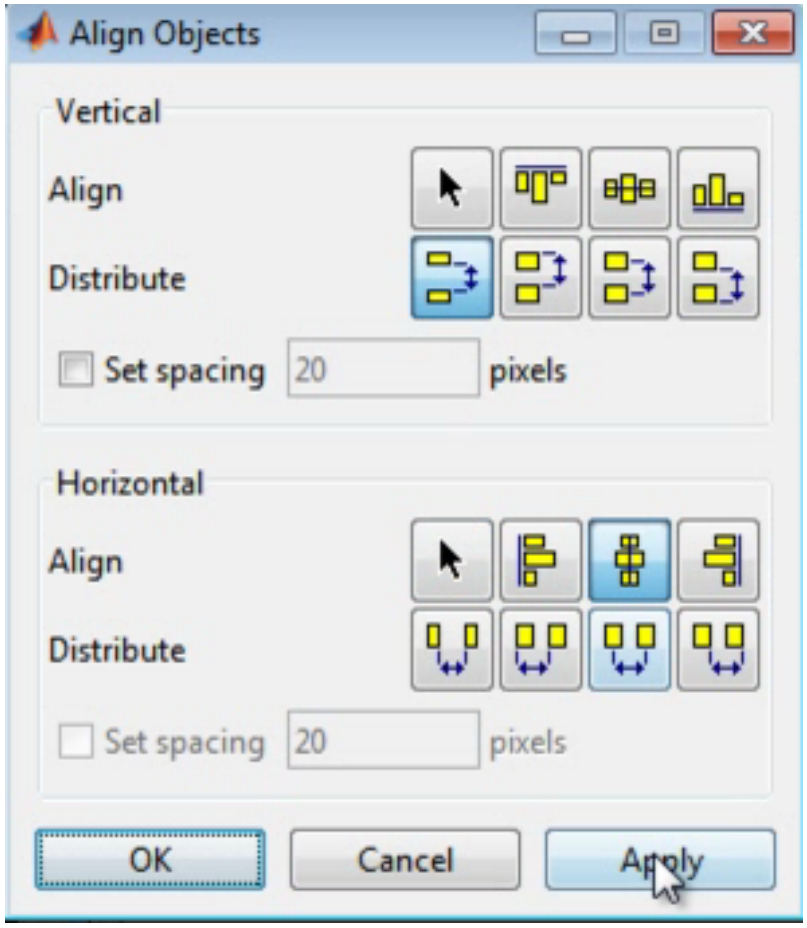
GUI layout with three non-aligned pushbuttons.

- Then we finalize the placement of these buttons by using the alignment tool. When the alignment button, shown here, is clicked,



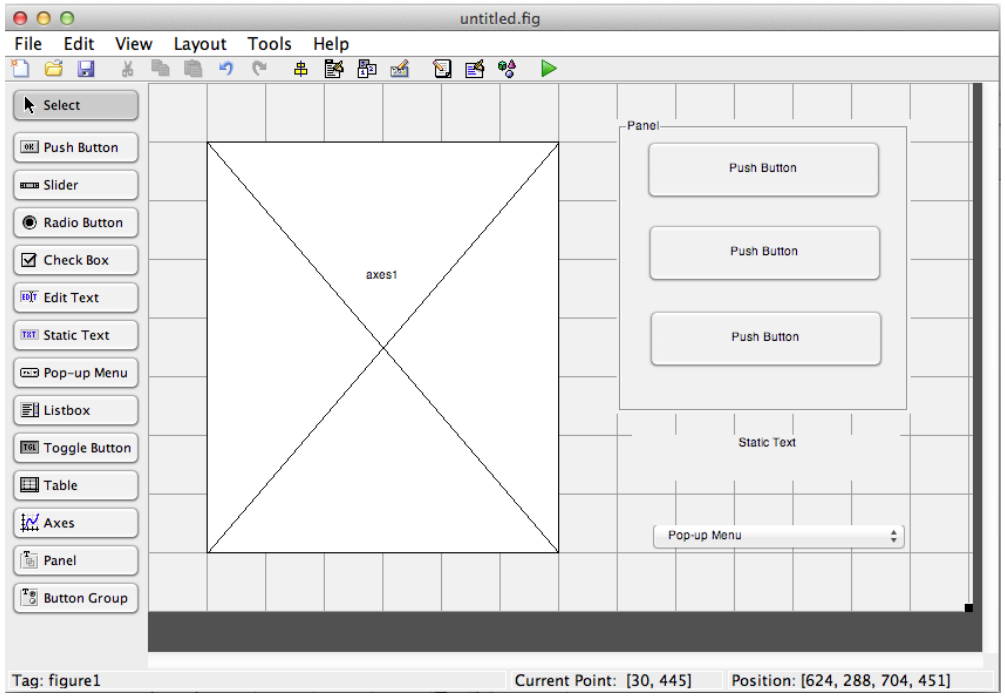
The alignment button on the left.

the alignment window appears:



The alignment window with the correct buttons applied.

- We add the other components of the GUI, a static text box and a popup menu. We have:



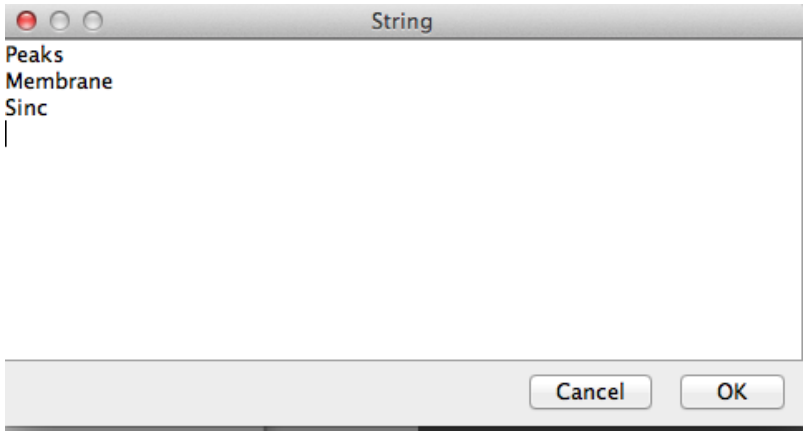
GUI layout with static text and popup menu.

- We access the property inspector again. We change the "Name" title to **simple_gui**. This changes to name on the title bar from "untitled.fig" to "simple GUI". We click on the panel object and change the panel Title property to "Plot Types".

- To set the text we want to appear on the pushbuttons, click on each push-button in turn, go to the String property in the property inspector for that button and type the text you want the pushbutton to be labelled with. We will use "Surf", "Mesh" and "Ribbons". Note that the later label different from the MatLab GUI, which uses "contour".

- We set the String property for a popup menu to contain the data sources. We will use **peaks**, **membrane** and **sinc**. The former two data items are built into MatLab but we will have to program the **sinc** data. The figure

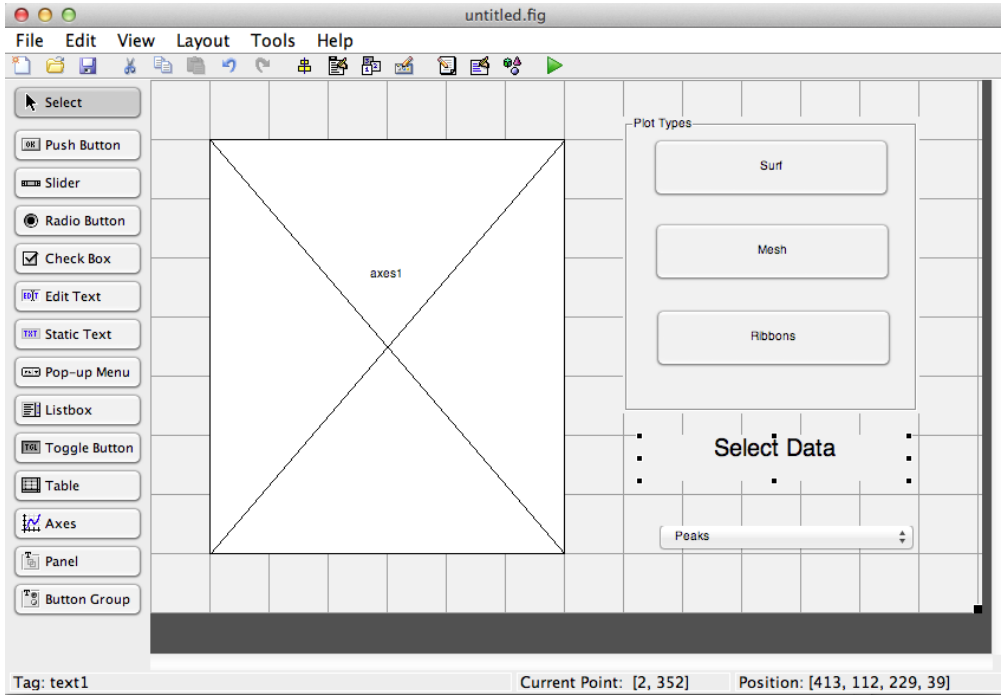
below shows this string window: go to the String property, click on the small icon with 6 lines of various lengths on it and the String window will popup. Type “peaks”, ”membrane” and ”sinc” (without quotes) in it. This window is shown as:



Text for the popup window options.

- Just below the String property is the Tag property. MatLab uses the tag name as a unique identifier to remember which popup menu we have (important is we have more than one). We repeat this for the other controls (the pushbuttons) with tag names "surf_pushbutton", "mesh_pushbutton" and "sinc_pushbutton".

- We set the text for the empty static text box to "Select Data" with fontsize 20. At this point we have the GUI layout:

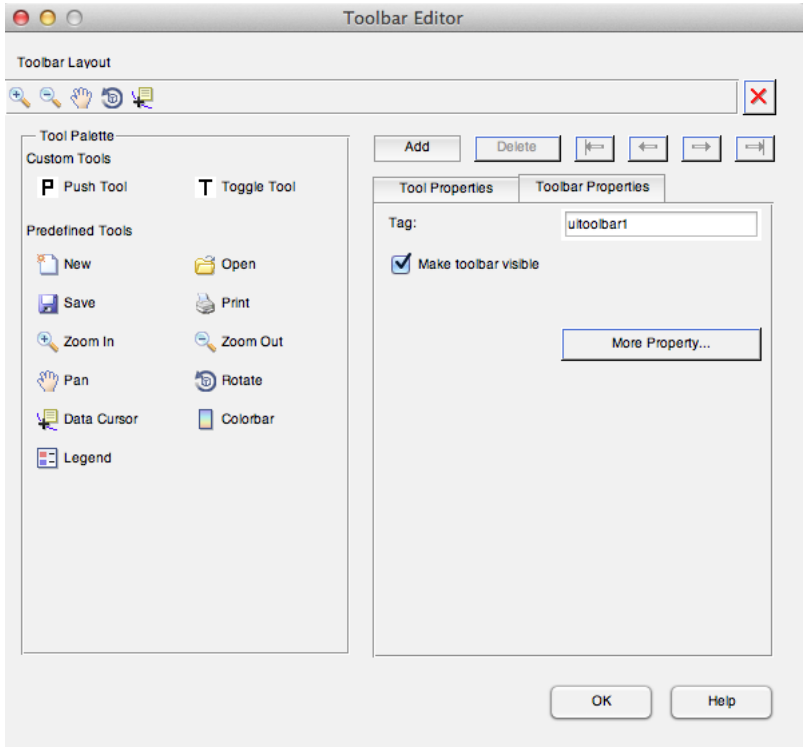


GUI layout with all buttons labelled and tags set.

- We can also add some toolbar functions by using the toolbar editor, indicated by the icon:

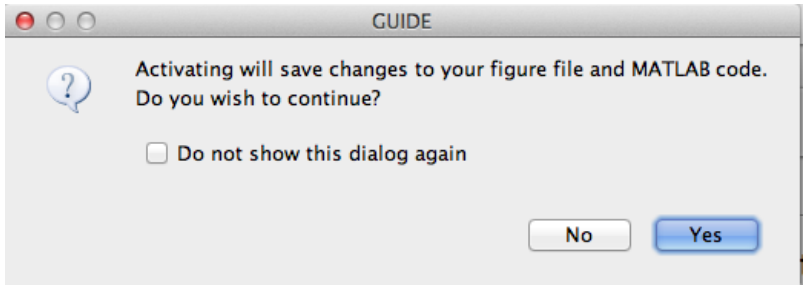
Toolbar editor icon.

- This pops up the toolbar editor with allows us to add some predefined buttons (or we can define our own as well). We choose 5 buttons: zoom+ (zoom in), zoom- (zoom out), pan (move around the image in xy plane), rotate (the graphs in 3D) and data cursor (examine the 3D coordinate data on the graph) and put them in the top toolbar available for this purpose. The figure below shows the toolbar now:



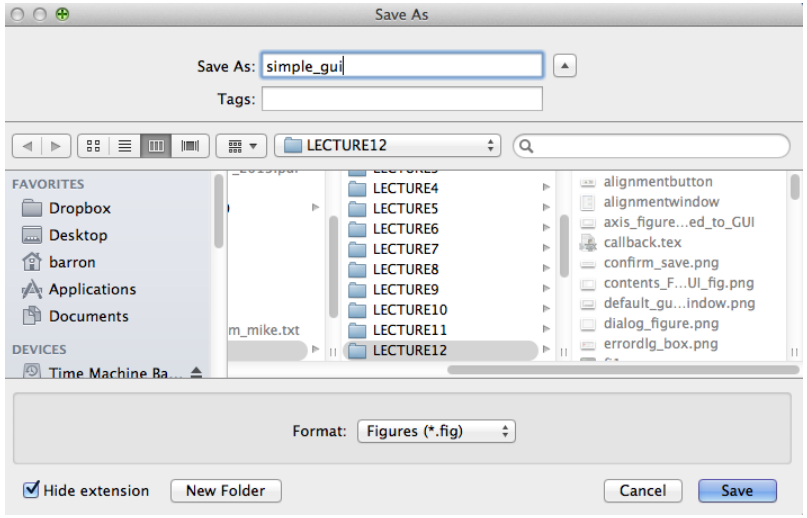
Toolbar editor with 5 predefined tools selected: zoom+, zoom-, pan, rotate and data cursor.

- At this point, you can save the layout by clicking on the right most **green** arrow (as mention above when discussing the property inspector). The following window pops up:



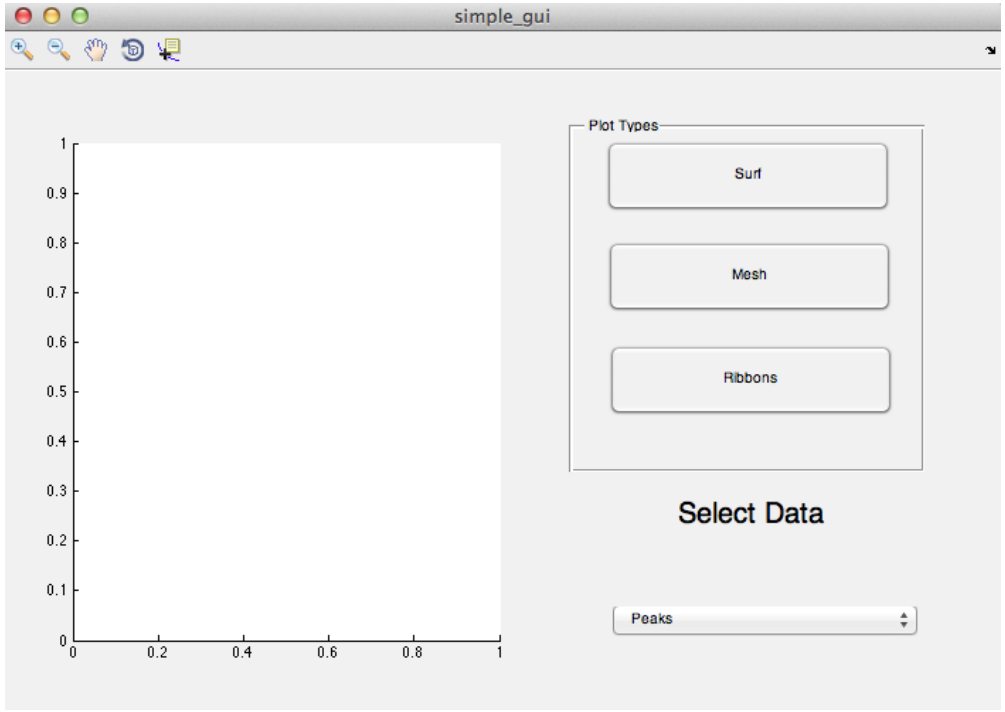
GUIDE prompt to save the layout, generating m and fig files.

- We select "yes" and then specify "simple_gui" as the stem names for the 2 files "single_gui.m" and "simple_gui.fig" that MatLab generates automatically.



Save the GUI layout as simple_gui.m and simple_gui.fig.

At the point the MatLab editor pops up with simple_gui.m as the file currently being edited and the GUI layout displayed as:



Current GUI layout.

- Note that the x and y axes of the figure are displayed as coordinates $\in [0, 1]$. Also note that the toolbar is displayed as it was set up.

- File **L13simple_gui.m** contains automatically generated MatLab code to control the GUI. You do NOT modify this code; rather you add you code to the various callback functions.
- Navigate to the opening callback function, simple_gui_OpeningFcn, which will executes first. After the code:

```
% --- Executes just before simple\_gui is made visible.
function simple\_gui\_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 simple_gui (see VARARGIN)
```

we add:

```
handles.peaks=peaks(35);
handles.membrane=membrane;
% 35*35 x and y arrays
[x,y]=meshgrid(-8:0.5:8);
r=sqrt(x.^2+y.^2)+eps;
sinc=sin(r)./r;
handles.sinc=sinc;
handles.current_data=handles.peaks;
surf(handles.current_data);
```

The call to `peaks(35)` sets the array `handles.peaks` to be a 35×35 array of the builtin `peaks` MatLab function. Array `handles.membrane` is set to the 35×35 array of the builtin `membrane` Matlab function. We have to compute the 35×35 `sinc` values and save them in the array `handles.sinc`. We set `handles.current_data` to point to the `peaks` data. Note that all 3 data sets vary from -8 to 8 by 0.5 (35 values along each dimension).

- The statement `handles.output=hObject` specifies the output argument when the GUI is executed.

- The last statement `guidata(hObject, handles)` must be present in all callback functions and updates the `handles` data structure.

- If we run this code now we can actually see the peaks function displayed as a surface.

- Next we must write the code for the pushbutton callback functions.

  1. For the `surf_pushbuttion` callback function we have:

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

     after which we add:

```
surf(handles.current_data);
guidata(hObject,handles);
```

2. For the `mesh_pushbutton` callback function we have:

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

   after which we add:

```
mesh(handles.current_data);
guidata(hObject,handles);
```

3. For the `ribbons_pushbutton` callback function we have:

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

   after which we add:

```
ribbon(handles.current_data);
guidata(hObject, handles);
```

- For the callback function for the popup menu we have:

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

% Hints: contents = cellstr(get(hObject,'String')) returns plot_popup content
%        contents{get(hObject,'Value')} returns selected item from plot_popup
```

followed by:

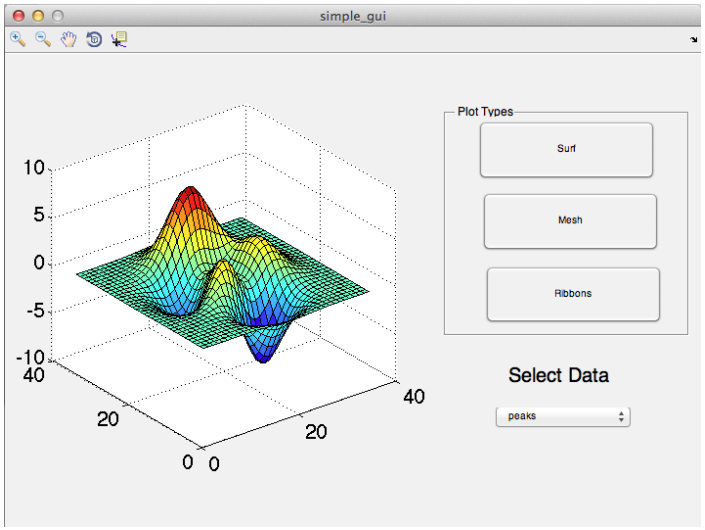
```
val=get(hObject,'Value');
str=get(hObject,'String');
switch str{val}
   case 'peaks'
        handles.current_data=handles.peaks;
   case 'membrane'
        handles.current_data=handles.membrane;
   case 'sinc'
        handles.current_data=handles.sinc;
end

guidata(hObject,handles);
```

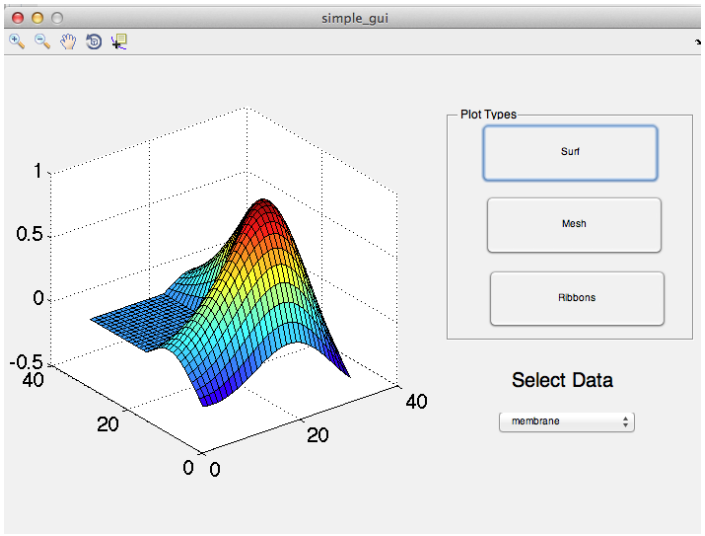
This code changes the `handles.current_data` to point to the selected 35×35 data (peaks, membrane or sinc).

- Note that you can make changes to **simple_GUI** by adding or changing buttons/code. In this case, the MatLab code you entered for other callback functions will remain unchanged. Very handy!!!

- The GUI is now ready to run. Type simple_gui in the command prompt window. Some results:

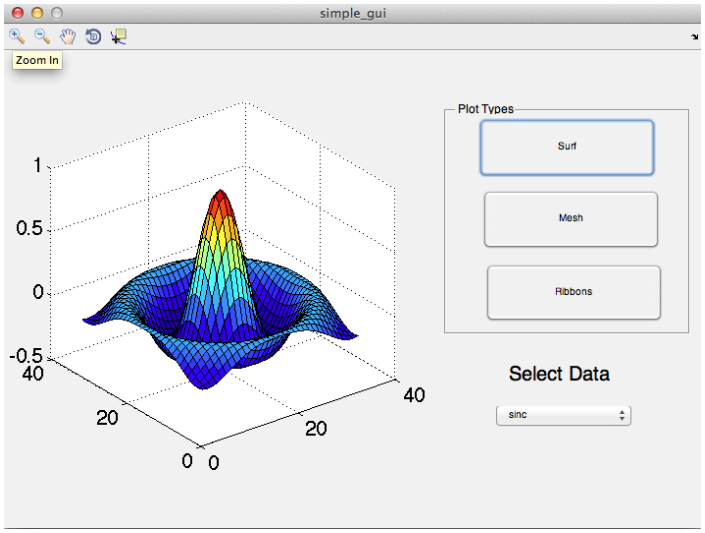


(a)

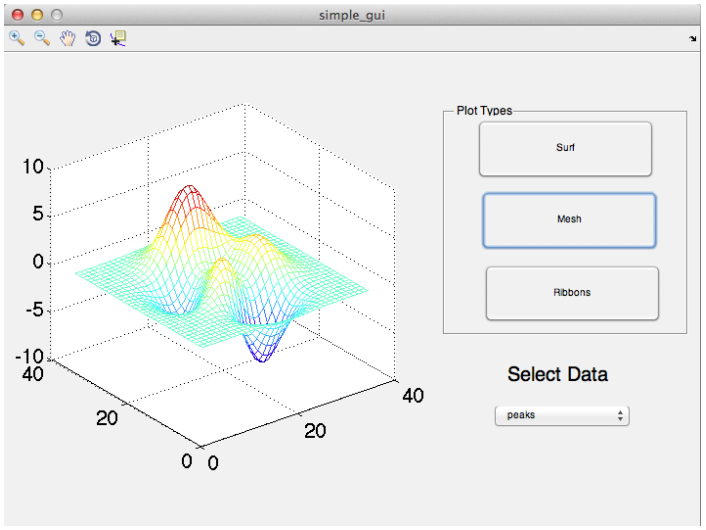


(b)

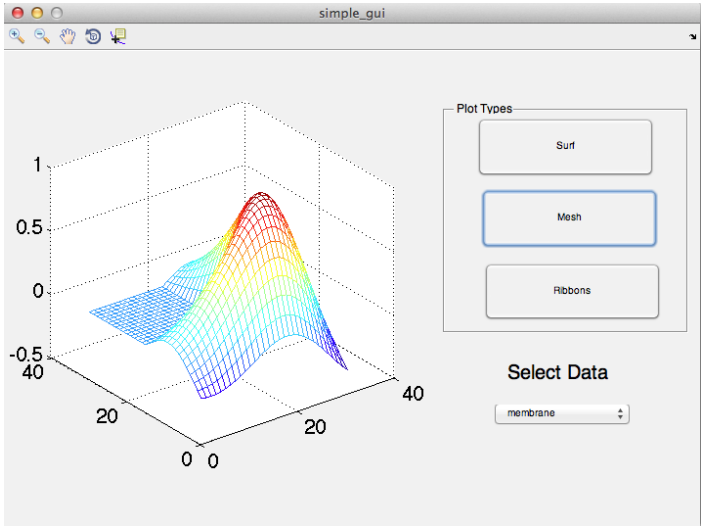
(a) Peaks surface plot and (b) Membrane surface plot.



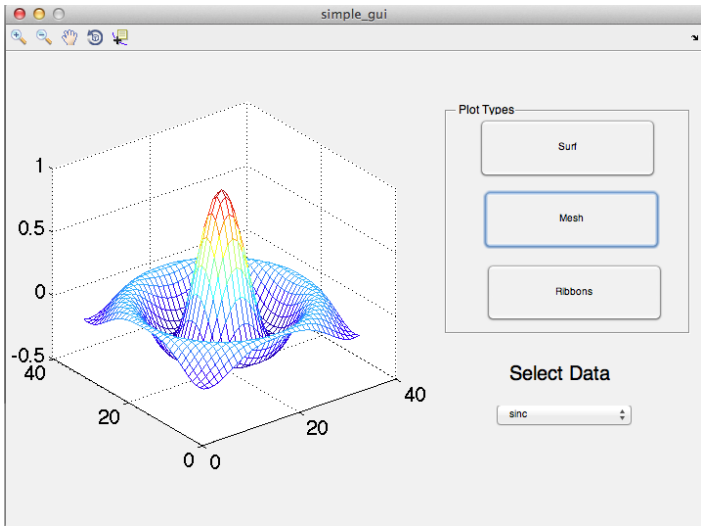
(a)



(b)

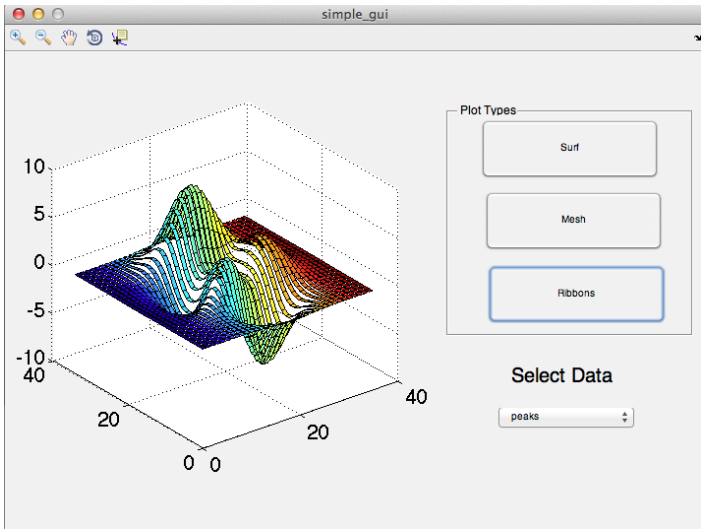


(c)

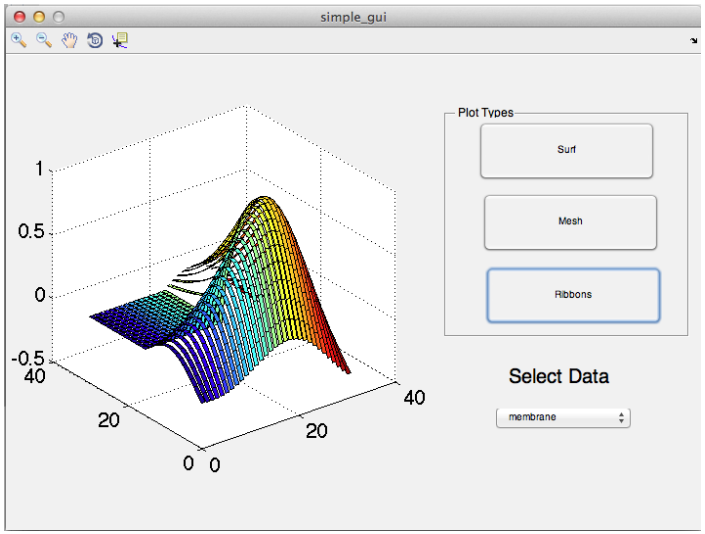


(d)

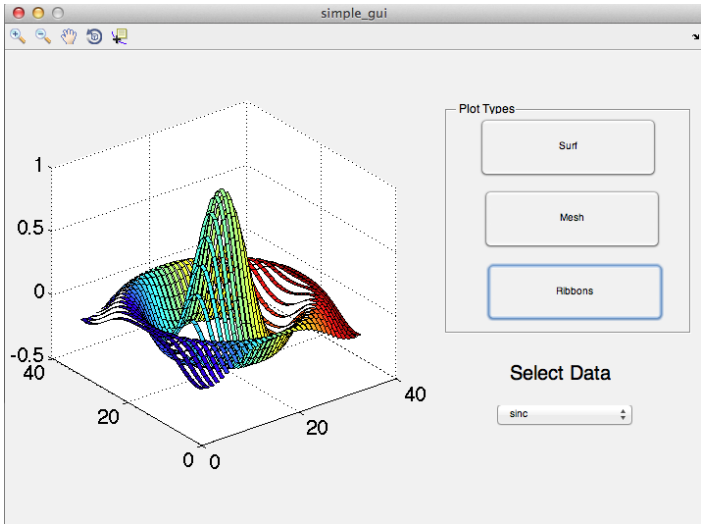
(a) Sinc surface plot, (b) Peaks mesh plot, (c) Membrane mesh plot and (d) Sinc mesh plot.



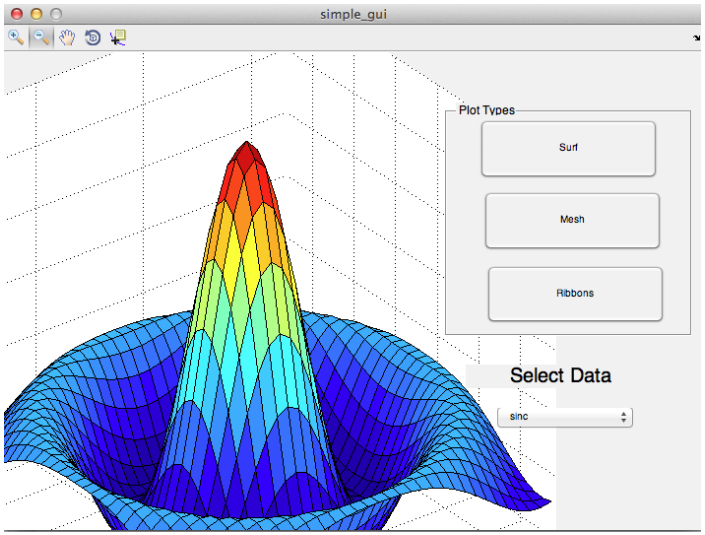
(a)



(b)

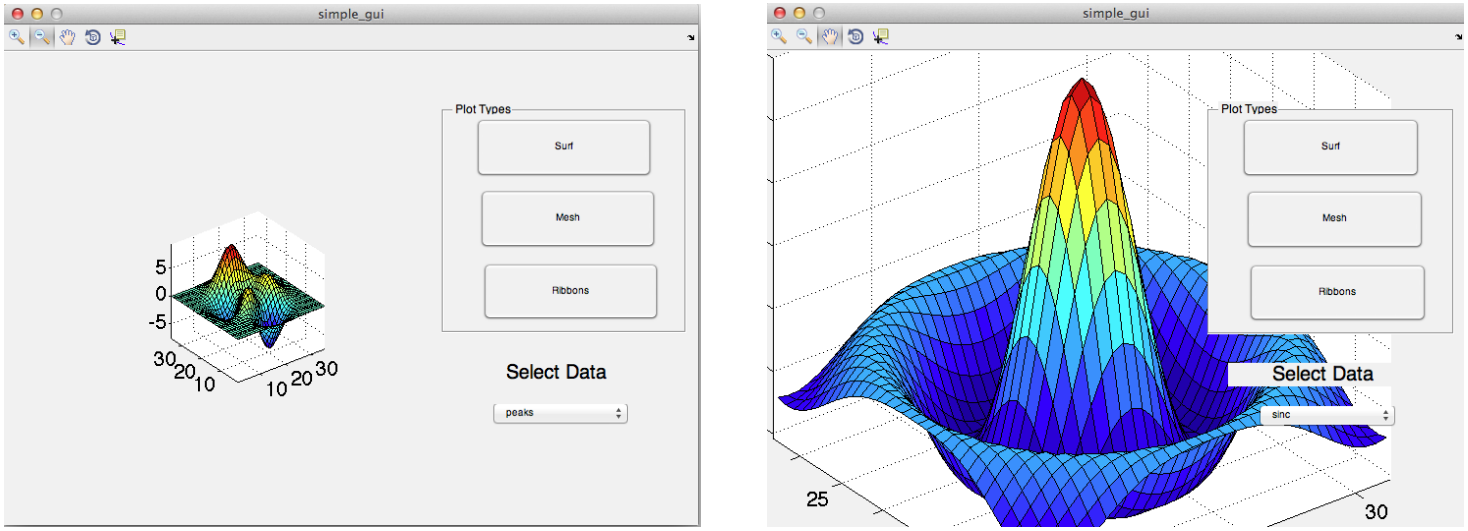


(c)



(d)

(a) Peaks ribbon plot, (b) Membrane ribbon plot, (c) Sinc ribbon plot and (d) Zoom in of sinc ribbon plot using the zoom- button.



(a) Zoom out of peaks mesh plot using the zoom+ button, (b) Pan of sinc meshplot using the pan button, (c) 3D rotation of the sinc mesh plot using the rotate button and (d) highlight of the coordinates of a surface point on a 3D plot on the rotated 3D sinc mesh plot in figure (c) using the data cursor button.