Tutorial 8 – Week 9

Aims:

Upon successfully completing these tutorial exercises, students should be able to:

- Create a basic Graphical User Interface (GUI) in MATLAB.
- Create multidimensional arrays.
- Create structures.
- Create more complicated functions containing arrays, conditional statements and repetitive statements.

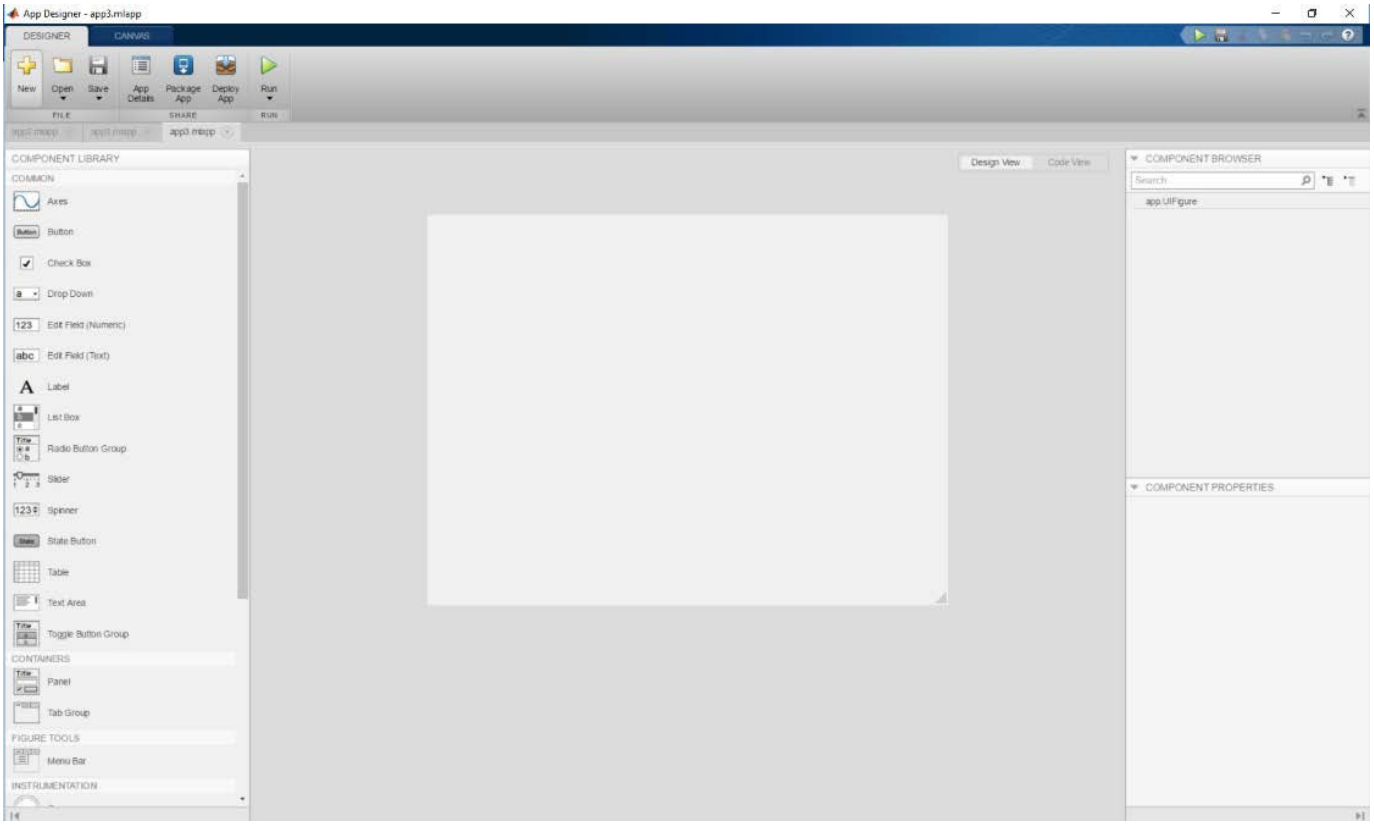
GUI

1 Create a basic Graphical User Interface (GUI) in MATLAB using App Designer

A GUI can be created using the App Designer. Refer to lecture notes & https://www.mathworks.com/help/matlab/creating_guis/create-a-simple-app-or-gui-using-app-designer.html

Start by typing 'appdesigner' in the command window.

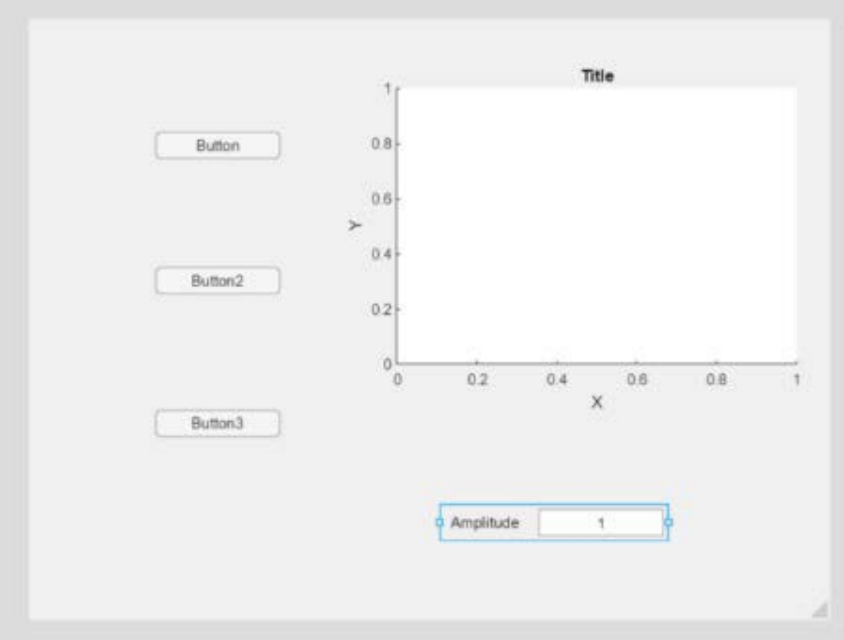
The following should appear:



Add

- three push buttons,
- an editable field (numeric) and
- an axis

Change the strings of the static and editable text to the values shown below. To change the properties of an object, use 'Edit Field Properties'.



We wish to create a GUI which plots a **sin**, **cosine** and **tan** wave upon the click of the push buttons. Double click the first push button to change the string to 'Sine'. Similar change the strings of the other push buttons using Cosine and Tan instead.

Save your file as Trig_plot

Change the component handles to app.sinewave, app.cosinewave and app.tanwave in the 'Component Browser'.

Change the component handle for the editable field to app.Amplitude.

Right click on the Sine pushbutton and select View **Callbacks>>Add**.

App Designer will take you to Code View, your code for the callback linked to Sine button is to be inputted in the white area of the browser.

Under the function **Sinewave_Callback** is where you place the code you want to run when the button is pushed. Insert the code as shown below:

```
x=0:0.01:4*pi;
a=app.Amplitude.Value;
app.UIAxes.YLim = [-a a];
plot(app.UIAxes, x,a*sin(x),'*');
```

Then run the GUI and test the functionality.

Make sure you have a value inputted for amplitude in the editable text box.

1.1 For you to do:

Now add the code for cosine and tan waves.

1.2 Extension for homework:

Add a second graph to the previous problem. The second graph should plot a function that is the inverse of the first function.

MULTIDIMENSIONAL ARRAYS

2 Create multidimensional arrays

It is possible to create arrays with as many dimensions as desired within MATLAB. However, practically, it is unusual to go beyond 4 dimensions. Let's create a numerical array of size 5x6x7 where every element is equal to 7. The matrix can be created using nested for loops.

```
for i=1:5
    for j=1:6
        for k=1:7
            a(i,j,k)=7;
        end
    end
end
disp(a)
```

Notice that i, j and k are indexes used to represent each dimension within the matrix. Let's try a more complicated example. Let's create a 2x3x4 array where each element is equal to the sum of all three indices. For example, element (2,1,2) would be equal to 2+1+2=5.

```
for i=1:2
    for j=1:3
        for k=1:4
            b(i,j,k)=i+j+k;
        end
    end
end
```

2.1 For you to do:

Create a 5x5x5 array where each element is equal to the smallest of the three indices.

For example, element (3,2,4)=2.

Hint, the function **min()** may be used to find the smallest of an array.

```
for i=1:5
    for j=1:5
        for k=1:5
            b(i,j,k)= min([i,j,k]);
        end
    end
end
```

3 Create structures

If you have multiple variables that you would like to clump together under one heading, a structure may be used. Take the example of the data associated with a student. The student will have a first name, last name, student number and average mark. We can place the individual variables all under a structure called student.

```
student.first_name='Ahmad';
student.last_name='Carlson';
student.student_no=5555555;
student.average_mark=72;
```

The full stop may be used to signify a structure. Structures are hierarchical, meaning they have layers. Data must be accessed from the top layer down to the bottom layer. Here, student is the top layer. It is possible to have more than two layers. Any layer may be an array if desired. Let's add:

```
student.subjects(1).mark=80;
student.subjects(2).mark=70;
student.subjects(1).name='ENGG100';
student.subjects(2).name='ENGG102';
```

Here student is the top layer, marks is in the middle and the subject names are the bottom layer. There exists only one structure student. However, inside the structure student exists an array of 2 structures called subjects. Inside each of these individual structures exists two variables mark and name. Try typing:

```
student

student.subjects

student.subjects(1)
```

3.1 For you to do:

Create marks and names for 6 other subjects inside the same structure. Then write code that determines which subject has the highest mark. Once found, print the name of the subject with the highest mark to the command window.

hint: Use fprintf

4 Create more complicated functions containing arrays, conditional statements and repetitive statements

Let's create a function with one input and one output. The input is a 3x3 numerical array containing only 1s and 0s. The output shall be one if at least one row contains three ones. Otherwise, the output will be zero.

```
function a=test_rows(b)
a=0;
for i=1:3
    if b(i,1)+b(i,2)+b(i,3) == 3
        a=1;
    end
end
```

Test the function with two test arrays:

```
b=[0,0,1;1,0,1;1,0,1];
```

```
b=[0,0,1;1,0,1;1,1,1];
```

The first should return 0 and the second should return 1.

4.1 For you to do:

Expand the function such that the output shall be one if any rows, column or diagonal contains three ones. Otherwise, return zero. Can you see how this function may be useful in noughts and crosses?