

---

# Plotting using Matlab

---

Vytautas Astromskas

# Contents

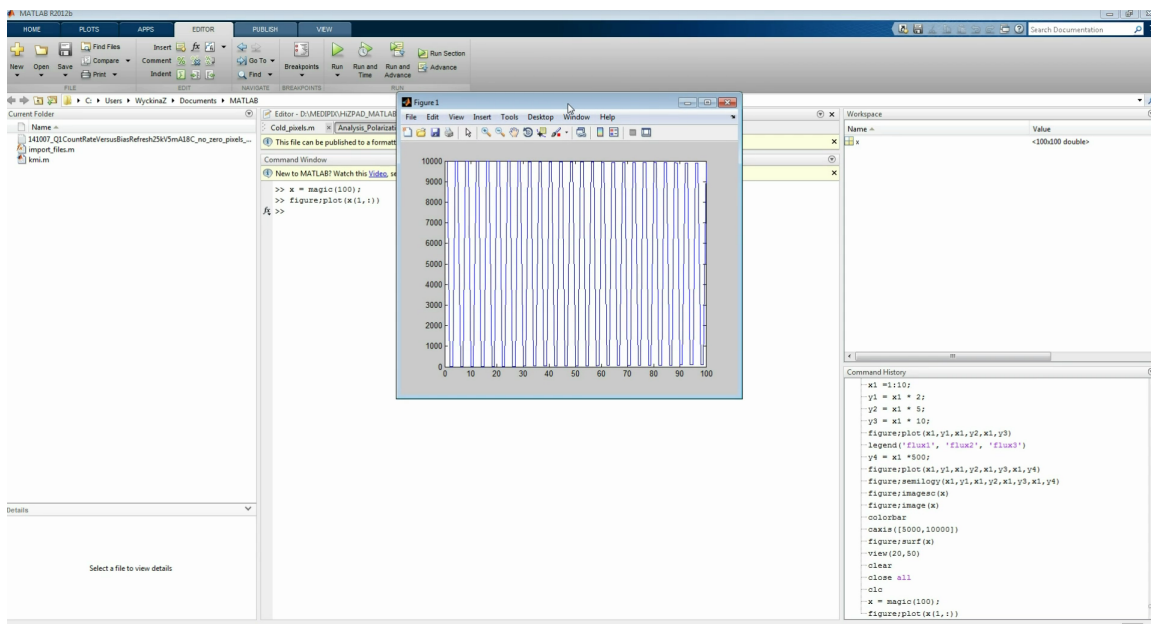
1	Manipulating a plot using GUI	2
2	Manipulating a plot using command line	4
3	Adding multiple lines to the same plot	5
4	Logarithmic axes	6
5	Plotting matrices	7
6	Plotting 3D surfaces	9

# 1 Manipulating a plot using GUI

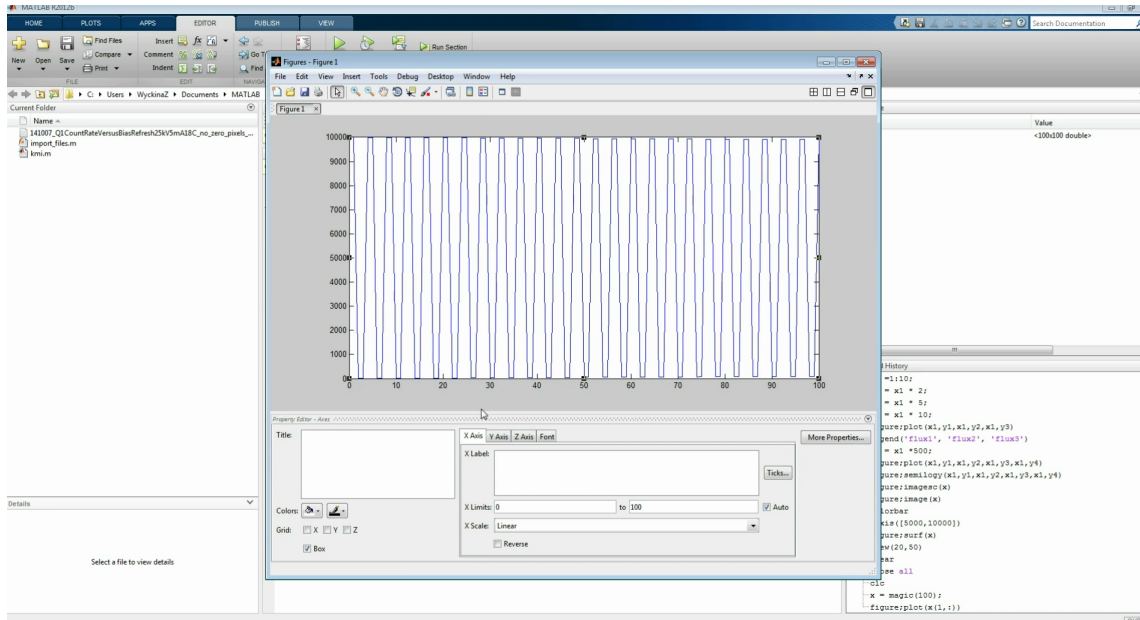
It is important to display your graph in a clear and informative manner in order for your data to be represented well. There are a couple of ways to manipulate your graph using Matlab. First one uses Matlab's Graphic User Interface (GUI). This is convenient for polishing a single plot. However, all of the manipulations can be implemented in your script so that a generated figure will have all the required labels, legends, font sizes and such already generated.

First, let us generate some data. Let us define  $x$  to be `magic(100)` matrix. This will be useful for demonstrating all the different plotting functions available in Matlab.

Generate a figure by writing `“figure; plot(x(1,:))“` in the command line. This figure will contain the first row of the  $x$  matrix.



In order to manipulate your graph using the GUI, `“Edit“` and `“Axes Properties“` in the drop down menu. A window will appear allowing you to manipulate the labels, title, legend, font sizes, linewidths and everything else you might need to tailor your plot for publication.



First, **label the axes**. You can access that in the “X axis“ and “Y axis“ tabs in the bottom of the screen. As an example, change the x label to be “Time (s)“ and y label to be “Current (mA)“. In the axis’ tabs you can also change the range that you want to be displayed as well as the type of the axis, whether it is logarithmic or linear.

Now, add the **title** by inputting “Current over Time“ inside the box on the left hand side of the window. You should see a title appearing on the top of your figure.

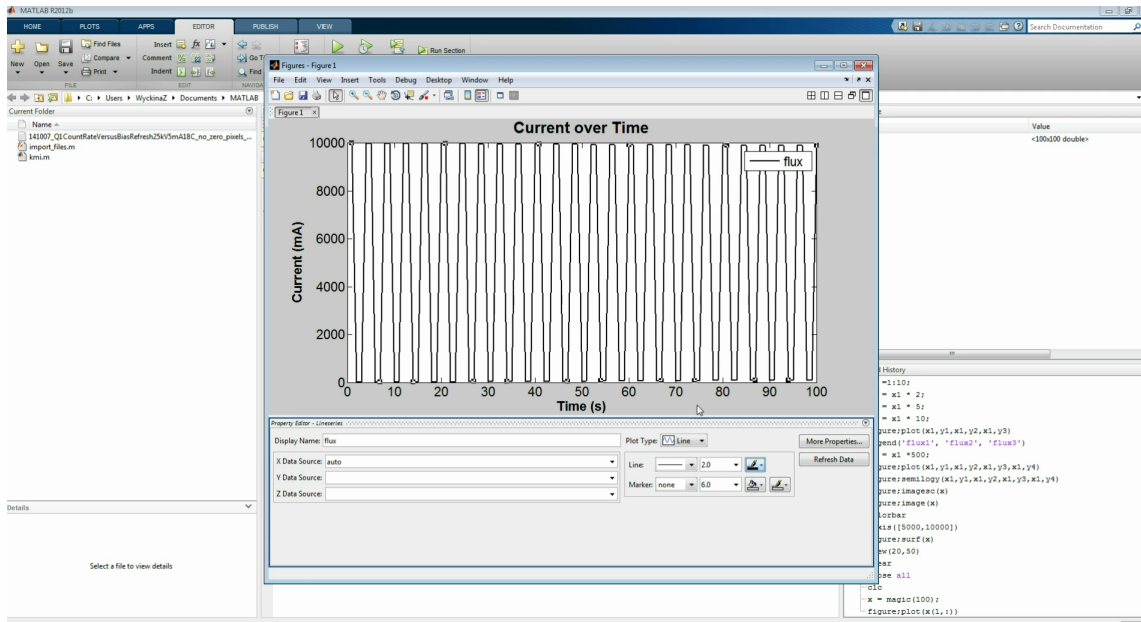
In order to change the font or **font size of your labels**, double click on the label. A different sub-window will change on the bottom of the “Axes Properties“ window. Change the font size to be “16“. I find this to be used as a rule of thumb in most cases. Also, change the font to bold. Moreover, perform the same actions on the Y axis and the title.

In order to change the **font size of the numbers on your axes**, select your graph again and go to “Font“ tab. Here, change the font size to 16 to match the labels.

**Legend** is added by clicking the “Insert Legend“ button in the toolbar and a legend should appear. You can move the position of the legend on the figure as well as change the string explaining the line. In order to change the text, double click on the legend and type

“flux” in this case.

Finally, you can alter the way the line itself is represented. Click on the line of your graph and you can change the style, color and line width on the right hand side of the of the sub-window. As an example, change the line width to “2”. I usually keep it “2” as a rule of thumb.



## 2 Manipulating a plot using command line

These are the commands to use in the command line to perform the same figure manipulations as using the GUI:

`xlabel('String')` assigns text to your x axis. If you want to change the font size, then add a property after the string i.e. `xlabel('String','fontsize', Font size value)`.

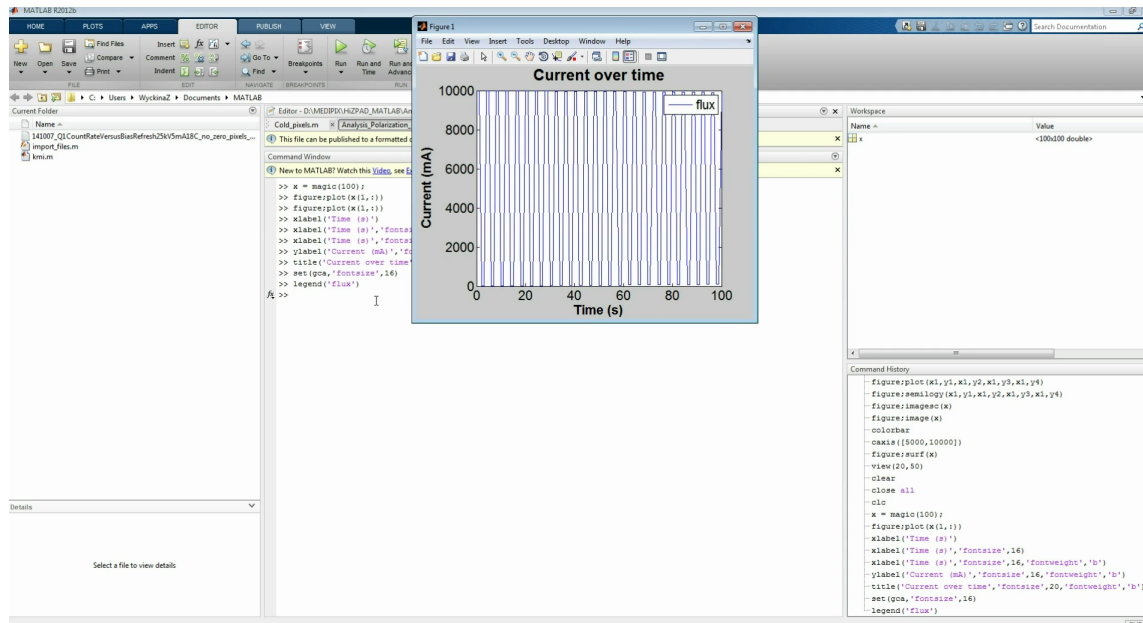
The same goes for y axis i.e. `ylabel('String','fontsize',Font size value')`

Title is added using `title('String','fontsize',value')`

Changing the font size of the numbers on the axes is a bit different. You have to have the figure you want to alter opened. Afterwards enter `set(gca,'fontsize', value)`. This sets

the font size of the current figure to the value you want.

The legend is added using `legend('String')`.

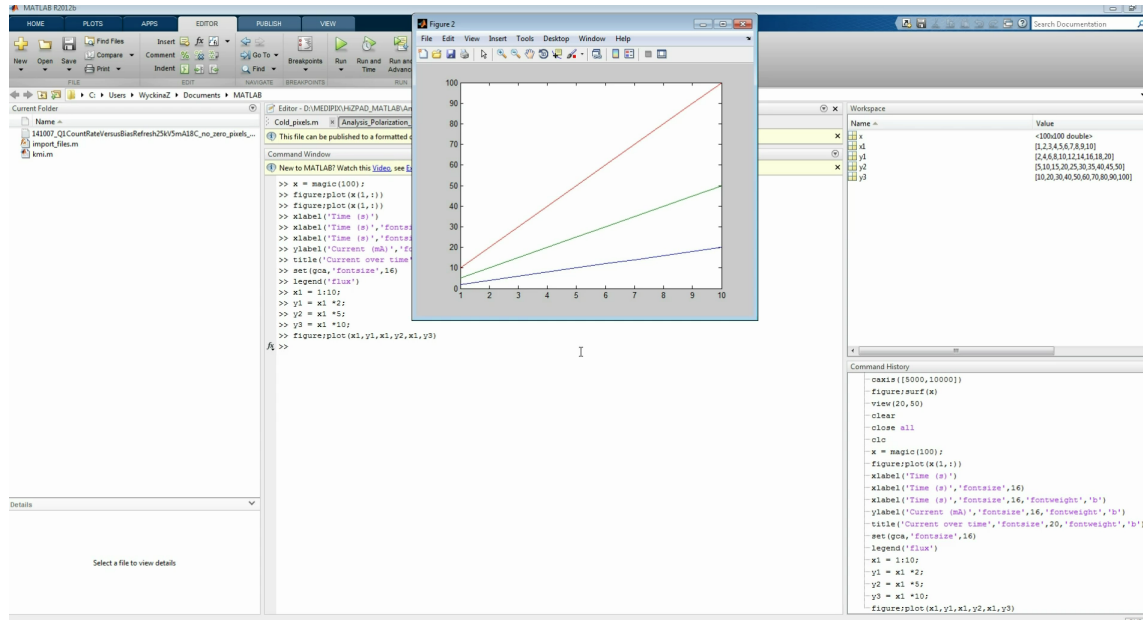


### 3 Adding multiple lines to the same plot

Firstly, define the different variables you want to plot. As an example, define  $x1 = 1:10$ ;  $y1 = x1 * 2$ ;  $y2 = x1 * 5$ ;  $y3 = x1 * 10$ ; This provides us with three different variables to plot. Now, if you want to put them on the same figure:

```
figure;plot(x1,y1,x1,y2,x3,y3);
```

You can put as many lines as you want as long as you define it x axis variable followed by an y axis variable.

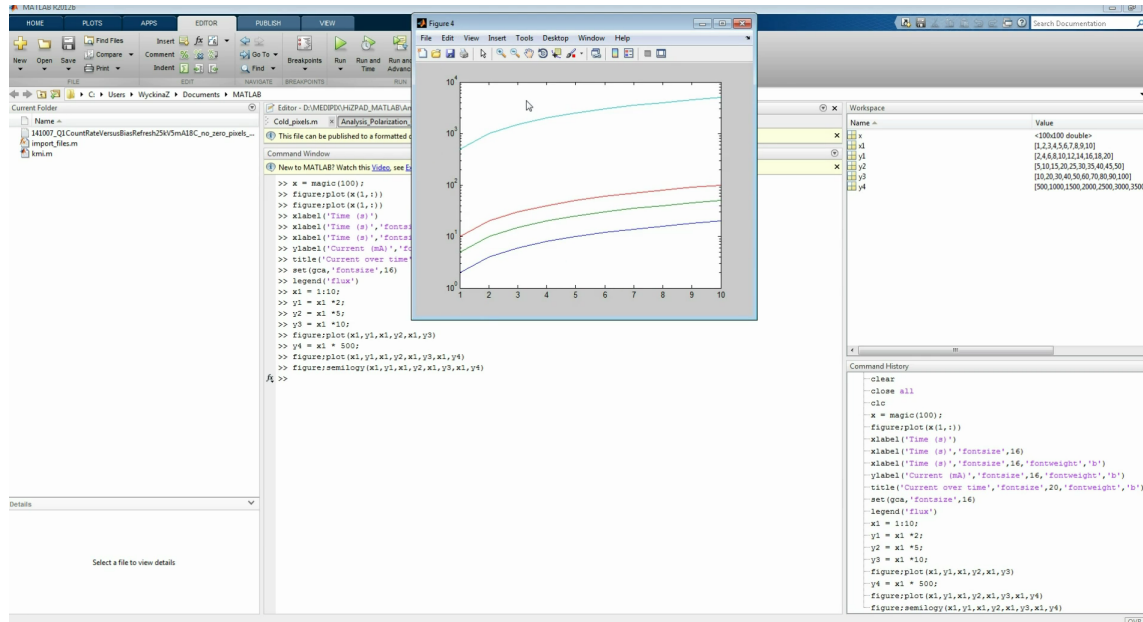


## 4 Logarithmic axes

I have showed how to change the type of axes from linear to logarithmic in the first section. If you want to perform the same action using the command line you have to use **semilogx** or **semilogy** functions which change x and y axes from linear to logarithmic, respectively.

As an example, define  $y_4 = x_1 * 500$ ; . This way using plot function your  $y_1$ ,  $y_2$  and  $y_3$  lines would not be visible. However, plotting all four lines using semilogy allows you to represent all the data using the same figure.

**figure;semilogy(x1,y1,x1,y2,x1,y3,x1,y4);**

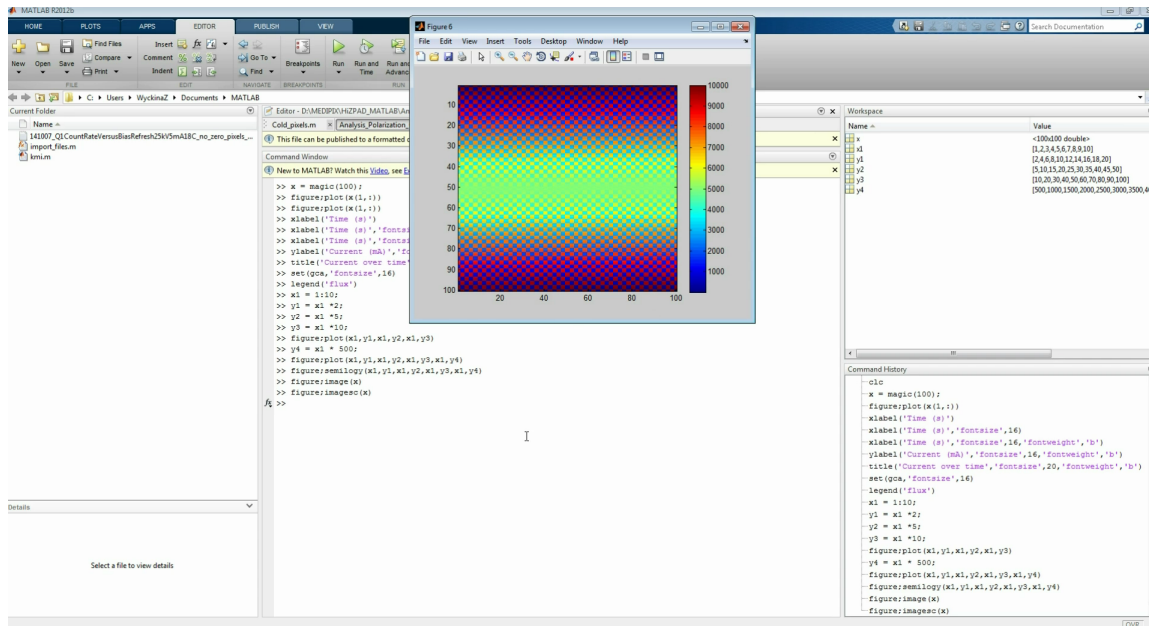


## 5 Plotting matrices

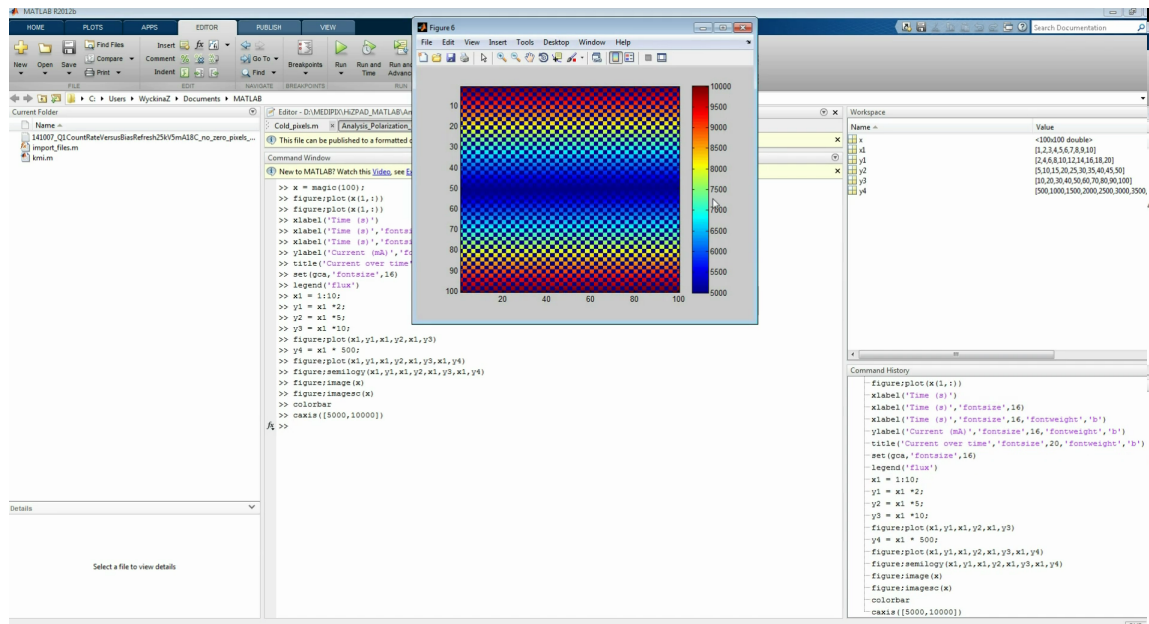
The most common function for plotting matrices is **imagesc**. *Imagesc* creates a two dimensional view of your matrix with rows and columns shown in x and y axes, respectively. The value of your matrix is going to be shown on the colorscale. *Imagesc* has an advantage over *image* as it automatically adjusts the colorbar axes to accommodate the data.

In order to visualise this run **figure;imagesc(x)** and **figure;image(x)** to compare the results. I would recommend using *imagesc*.





In order to adjust the colorbar scale on your *imagesc* graph you have to use two functions. Firstly, add the colorbar using `colorbar`. Secondly, use `caxis([min_value max_value])`. *Caxis* has to be defined as a vector. As an example, use `caxis([5000 10000])` on the figure you have plotted previously.



## 6 Plotting 3D surfaces

**surf** is used to plot 3D surfaces. The syntax for surf is **surf(x,y,z,C)** in order to define the dimensions of each of the three dimensions as well as the colorscaling using the C vector. In our example, you can plot **surf(x)** just as happily. In this case it will assign the matrices rows and columns to be x and y axes. Z axis is going to be defined as the minimum and maximum of your values in the matrix. Colorscale is going to be the same as the Z axis.

In order to change the viewing angle you can use **view(AZ,EL)** function where AZ is the azimuthal or the horizontal angle and EL is the vertical elevation.

For our example, use **figure;surf(x);view(10,80)**

