Matlab Laboratory 1. Introduction to MATLAB

Aim of the laboratory

- Familiarize with the Matlab environment
- Creating periodical function plots

Necessary equipment

- Workstations with MATLAB

Theoretical Approach

Matlab is a numerical computing environment dedicated to numerical calculus and graphical representations in the fields of science and engineering [1]. Matlab is a very powerful tool in academia in industry to solve problems in the fields of signal processing, system identification, statistic control, etc. [1]

In contrast to traditional programming languages, Matlab resembles a natural, easy-to-learn and easy-to-use syntax, which enables a straightforward expression of the problems to be solved. In this respect, Matlab consists of high-performance software packages dedicated to numerical analysis, matrix calculus and signal processing. Accordingly, Matlab enables a variety of applications, such as:

- numerical calculus and evaluation
- algorithm development,
- data acquisition,
- modelling and simulation,
- data analysis, exploration and visualization,
- graphical environment,
- applications with graphical user interface (GUI).

The elementary data structure in Matlab is the matrix. Computation in Matlab is then performed around matrices, enabling easy matrix manipulation, algorithm implementation, as well as plotting of data structures and functions. Further on, Matlab allows easy creation of user interfaces and interfacing with routines written in other programming languages.

The Matlab environment consists of five subsystems:

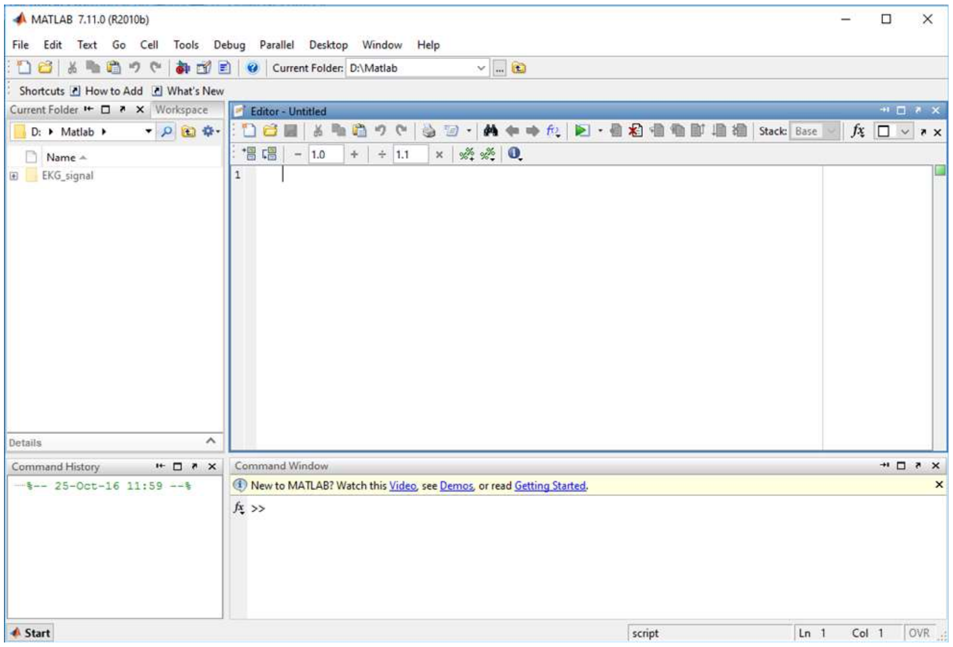
- an easy-to-work-with environment
- mathematical functions libraries,
- computation resources,
- graphical environment,
- an Application Programming Interface (API) which enables writing programs in other programming languages such as C or FORTRAN, making routines easier.

For familiarization with the Matlab environment, a tutorial for plotting a sine wave is presented further on in Exercise 1.

Exercise 1.

1. Launch the Matlab environment.

Matlab is launched by double clicking the shortcut on the desktop. The Matlab workspace is illustrated in figure 1.



```
%-- 25-Oct-16 11:59 --%
```

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
>>
```

Figure 1.

The Matlab environment illustrated in figure 1 exhibits four regions:

- the command line,
- the file editor,
- the folder managed/workspace,
- the command history.

Nevertheless, the Matlab environment can be customized to the specific needs of the developer, and can also include a debugger, the help browser, etc. Additionally, each region can be either docked in the Matlab main window, as shown in figure 1, or can be unlocked into a dedicated window.

2. Create a new folder / Open an existing folder

Every new Matlab project should be developed in a dedicated folder. To create a new folder one should open the folder browser, select the desired folder location, type the name of the new folder and click on the "Make New Folder" button, as illustrated in figure 2.

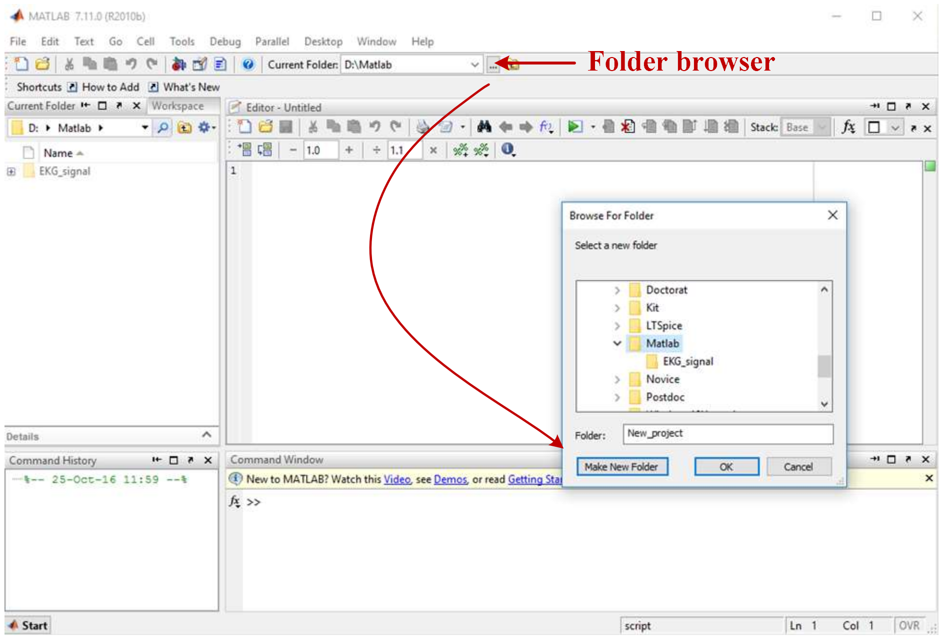


Figure 2.

To continue working on an existing project, one should open the project folder by going to the folder browser and selecting the desired folder.

3. Editing the source code

The source code can be written and run either directly in the command window, or in dedicated source files.

For exemplification, type the following Matlab code in the command window, one line at a time.

```
clear all;
close all;
f=50;                   %frequency
t=(0:0.001:0.04);       %time with the values ranging from 0 to 0.04s
                        %with a step size of 0.001s
y=sin(2*pi*f*t);        %function y
plot(t,y);              %the function that plots the graphical representation
```

Every function used in editing Matlab source codes can be found in the Matlab help. For the Matlab product help go to ***Help*** → ***Product help***, or for the function browser go to ***Help*** → ***Function browser***. Information regarding a specific function can be aquired by typing ***help*** into the command line, followed by the function to be inquired, for example:

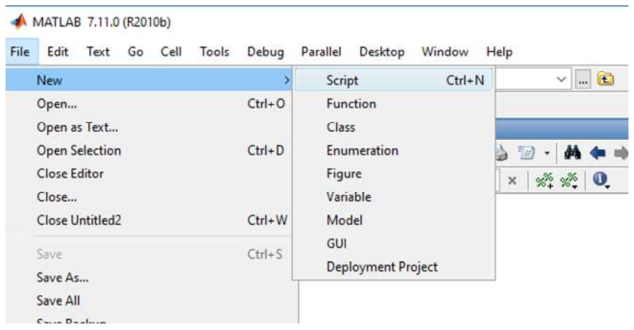
```
help [function];
```

At this stage, the preferable approach is consulting the Matlab product help, as it provides a thorough presentation of the inquired matlab function with consistent explanatory text and coding examples. Also, the Matlab product help provides related functions which can be helpful in the development process.

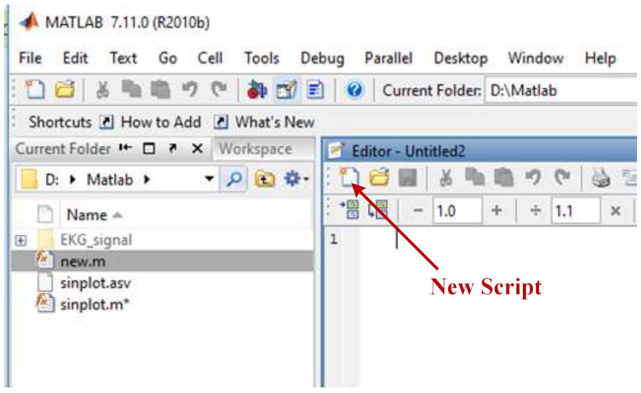
In the source code above, functions ***close*** and ***clear*** close the windows opened and delete the values of all variables declared at a given moment respectively.

The same code can be written into a source file. Basically, Matlab exhibits two types of source files: functions and scripts.

A script is a source file that contains stand-alone code which is run by typing the filename in the command window. To edit a Matlab script, create a new folder by going ***File*** → ***New Script***, or by clicking on the ***New*** button in the file editor, as illustrated in figure 3.



(a)



(b)

Figure 3

Type the same Matlab code into a source file, save it into the current folder and the run the script by typing the filename into the command window.

A function is a source file which contains arguments. These are later used in evaluation the instructions written in the function code. For exemplification, create a new file and type the following function code into a new file. When saving the function, the filename should be the same with the function name.

```
function f = sinplot(f,t)
```

```
y = sin(2*pi*f*t);       %function y
f = plot(t,y);           %the function that plots the graphical representation
```

When running the function from the command line, the argument values should also be specified. Accordingly, type the following instructions into the command window, one line at a time:

```
clear all;
close all;
f=50;
t=(0:0.001:0.04);
sinplot(f,t)
```

The result of all three approaches to plot the sine wave is the same, as illustrated in figure 4.

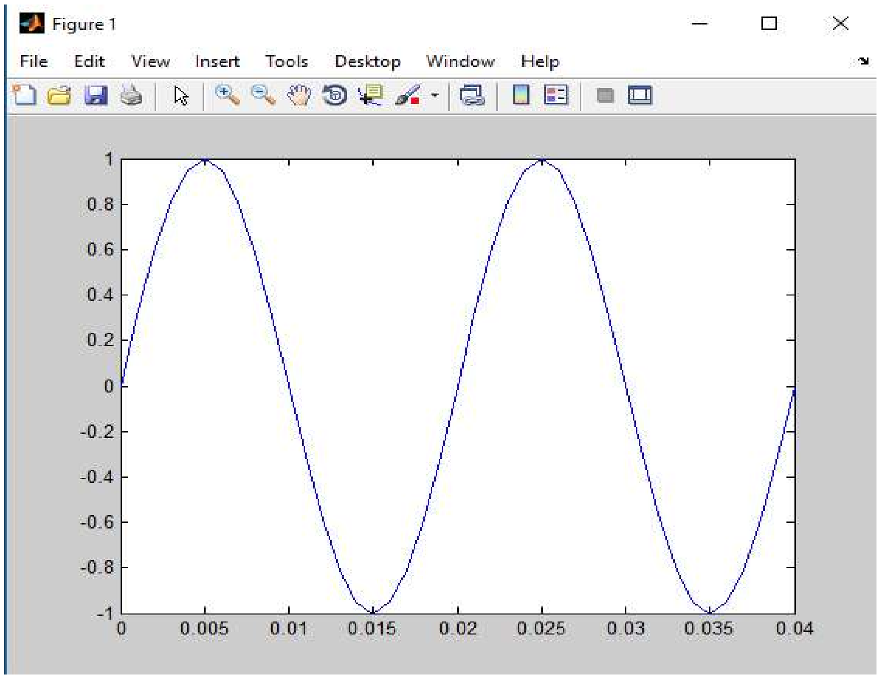


Figure 4

For editing the graphical representation of the sine wave, look up the *plot* function if the Matlab product help by going ***Help* → *Product help***. Information can also be obtained by typing into the command line.

```
help plot;
```

For exemplification, readability of the sine wave can be enhanced by introducing gridlines into the function plot. For this purpose, type

```
grid on;
```

into the Matlab script/function. Accordingly, the function plot changes as illustrated in figure 5.

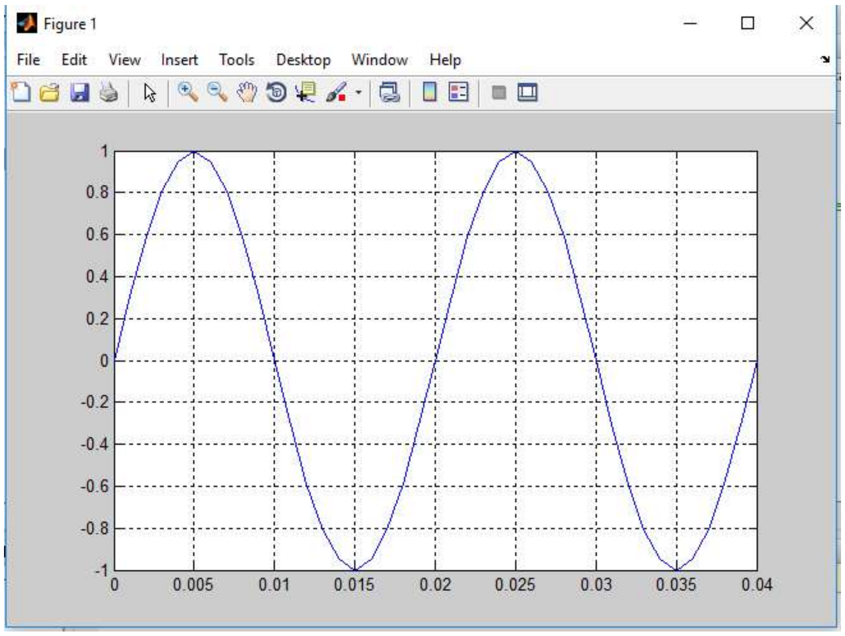


Figure 5.

The signal plotted in figure 5 has a better representation. Indeed, one can observe the period of the signal reported to the frequency introduced in the Matlab script, or transmitted as a function argument. For even more customization several other MATLAB functions can be used, as will be illustrated in the next exercise.

Exercise 2

This exercise plots the sine wave expressed as

$$y(x) = A \cdot \sin(2\pi f t)$$

with the possibility to control the signal amplitude *A*. One can easily observe that this function is similar to the one from the previous exercise. Accordingly, plotting the function follows the same steps listed above.

The starting point is setting the frequency. Let the frequency value be 50 Hz. The signal period is then computed as

$$T = \frac{1}{f} = \frac{1}{50} = 20\ ms$$

which states that the representation length should be at least *20 ms* in order to have at least one signal period represented. To represent one signal period, the time domain should be exactly *20 ms* long. A rule of thumb states that a proper resolution to plot periodic signals is about 100 points per period. Accordingly, the time domain is defined as:

```
p = 1/f;               %period
t = (0:p/100:p);    %time with the values ranging from 0 to 0.2s
```

Create a Matlab function, *sinplot.m*, which takes the signal amplitude and frequency as function arguments, and plots the corresponding sine wave. In contrast to the previous exercise, define the time domain within the function code.

Next, create a Matlab script to call the previously defined function. First, clear all data and close all opened windows. Next, set the frequency value. Finally, introduce control elements into the function plot to control the signal amplitude.

To modify values in MATLAB one can use the ***uicontrol*** function to create user interface control elements. Look up the ***uicontrol*** function in the Matlab product help and study the control elements available in a Matlab GUI.

In this exercise, two types of control elements must be defined: one text and one edit element respectively. The parameters of the control elements are determined using the

```
get(uicontrol);
```

command. However, not all parameters are used in the definition of a control element, some may be left with their implicit values. The parameters of interest on the other hand are modified by:

```
uicontrol('Style','text', ...
    'Units','normalized', ...
    'Position',[0.91 0.95 0.1 0.04], ...
    'String','A');

uicontrol('Style','edit',...
    'Units','normalized',...
    'Position',[0.91 0.9 0.1 0.04],...
    'String','A',...
    'Callback','A=str2num(get(gco,''string''));sinplot(A,f,t);');
```

To be noticed is that the callback of the '*edit*' control element calls the plotting function defined beforehand.

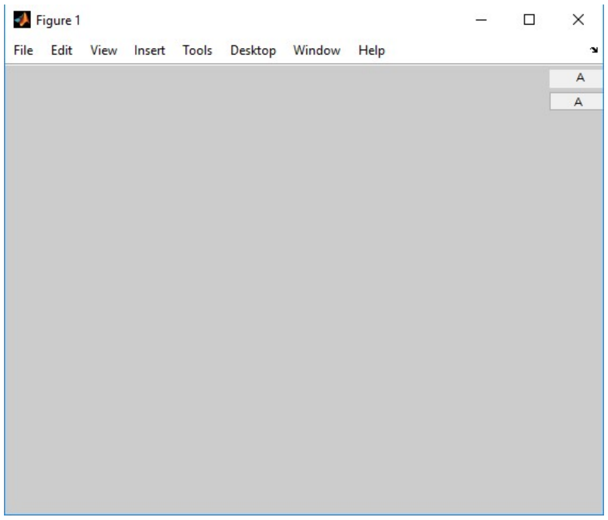


Figure 6.

After running the Matlab script, the window which appears is illustrated in figure 6. The two kinds of control elements defined using ***uicontrol*** can be observed. On is the text element is used to display the name of the parameter that needs to be modified and the other is the edit element to introduce a specific value. After introducing an amplitude value, for instance A=2, and hitting *enter*, Matlab will compute the sinusoidal plot using this value, as illustrated in figure 7.

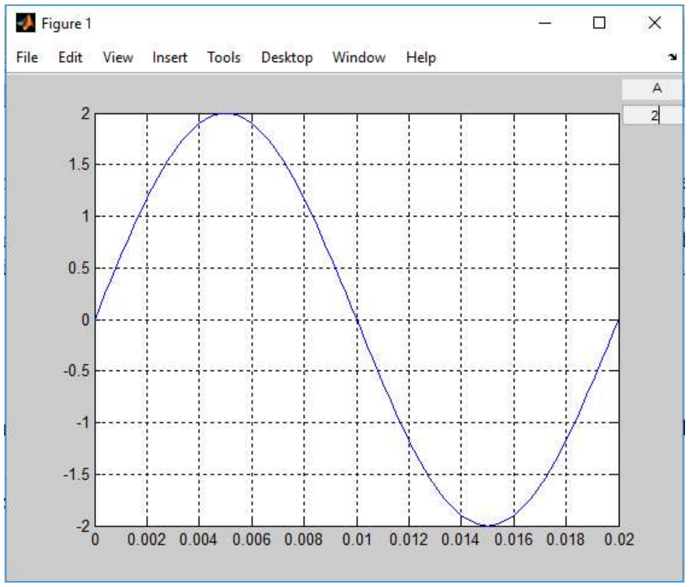


Figure 7.

The program further allows to change the amplitude value and redraw the function plot.

As an additional task, consider introducing a control for the signal frequency and the number of periods to be displayed.

To do…

1. Study the functions: str2num, get, gco, plot, subplot using MATLAB help,
2. Write a Matlab script to plot the graph of the function:

$$y = 2 \cdot \sin(\omega t) + 5 \cdot \cos(5\omega t)$$

Remember that

$$\omega = 2\pi f$$

3. Introduce control elements to set the value of the fundamental frequency and the number of periods to be displayed.