COMM.SYS.300 COMMUNICATION THEORY

Starting to use MATLAB®

Quotation from www.mathworks.com:

"***The Language of Technical Computing***

Millions of engineers and scientists worldwide use MATLAB® to analyze and design the systems and products transforming our world. MATLAB is in automobile active safety systems, interplanetary spacecraft, health monitoring devices, smart power grids, and LTE cellular networks. It is used for machine learning, signal processing, image processing, computer vision, communications, computational finance, control design, robotics, and much more.

Math. Graphics. Programming.

The MATLAB platform is optimized for solving engineering and scientific problems. The matrix-based MATLAB language is the world's most natural way to express computational mathematics. Built-in graphics make it easy to visualize and gain insights from data. A vast library of prebuilt toolboxes lets you get started right away with algorithms essential to your domain. The desktop environment invites experimentation, exploration, and discovery. These MATLAB tools and capabilities are all rigorously tested and designed to work together.

Scale. Integrate. Deploy.

MATLAB helps you take your ideas beyond the desktop. You can run your analyses on larger data sets and scale up to clusters and clouds. MATLAB code can be integrated with other languages, enabling you to deploy algorithms and applications within web, enterprise, and production systems. "

Matlab is available for Windows, Mac and Linux. The latest version of the software (when writing this document) is MATLAB R2016a, but earlier versions (especially since 2014b) are equally good for these exercises. The differences between each version can be compared in more detail in:

http://se.mathworks.com/products/matlab/whatsnew.html

Besides very common mathematical functions and programming tools, Matlab has many functions for specific study fields and applications, organized as toolboxes: DSP (Digital Signal Processing) system, Communications System, Neural network, Statistics and Machine Learning,…

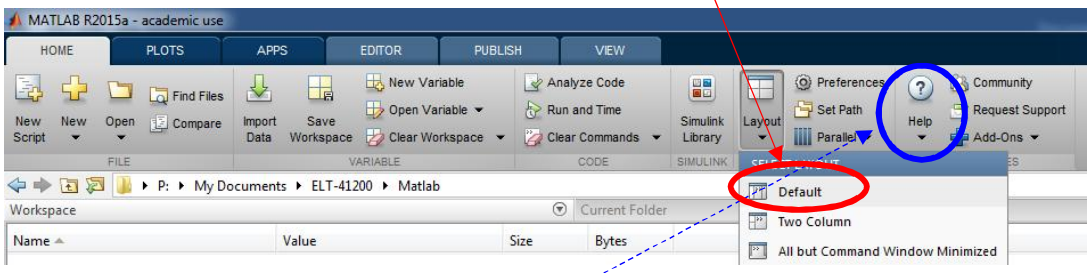
Matlab uses a relatively easy proprietary programming language, called the "M-code" (somewhat similar to C and Fortran), created by Mathworks.

Starting Matlab

Matlab can be started by clicking the Matlab icon (e.g. start menu → all programs → MATLAB →…)

When MATLAB has opened, depending on View settings, you should see some of the following: Command window, Current Directory, Workspace, Command History, Launch Pad, etc.

If you're a beginner with Matlab, the default desktop setup (see figure below) is a good starting point: (HOME tab → Layout → Default)



Using "Help"

You can find information on the Matlab functions, as well as examples and other types of demonstrations, from the Help icon (see the figure above).

Help is also available via the command window:

>> help function_name

>> doc function_name

(Since the Matlab's default command prompt is ">>", this notation is used above and further in the Matlab exercises to describe commands given in the command window. Just remember to press enter after giving the command ☺)

Environment:

Basically all numerical variables in Matlab are M x N -matrixes (a special case is a scalar, i.e. "1 X 1 matrix"). Matlab has been originally designed for matrix operations (MATLAB is actually an abbreviation from MATrix LABoratory), for which reason it can be very fast with matrix operations, Hence, it is generally recommended to use matrix operations as often as possible instead of for-loops etc.

Example constants in Matlab:

pi, complex numbers (imaginary unit i or j, nowadays it is recommended to denote these as 1i or 1j to avoid confusion with possible variables i and j), realmax, realmin, inf, eps etc.

Example functions:

sin, cos, exp, log10, abs, etc.

MATRICES:

Try out commands (preceded with >>) in your Matlab command window (you can skip this part if you already feel comfortable using Matlab):

Creating scalar

```
>> s=2
```

Creating matrix

```
>> a=[1 3;6 9]
```

Creating vector

```
>> v=[1 5 9]
```

Summation

```
>> a+5
```

Multiplication

```
>> b=s*v
```

Element-by-element multiplication

```
>> v.*b
```

Checking length of a vector

```
>> length(v)
```

Checking size of a matrix

```
>> size(a)
```

Accessing certain element of a matrix

```
>> a(1,2)
```

Accessing certain elements of a variable

```
>> v(1:2)
```

Creating vector with elements from 0 to 0.5 with 0.1 step-size (begin:stepsize:end)

```
>> t=0:0.1:0.5
```

OR

```
>> t=linspace(0,0.5,6)
```

M-FILE

M-file is basically an Ascii-file with the extension ".m". It can include a set of Matlab statements/commands (typically line-by-line) similar to any computer program.

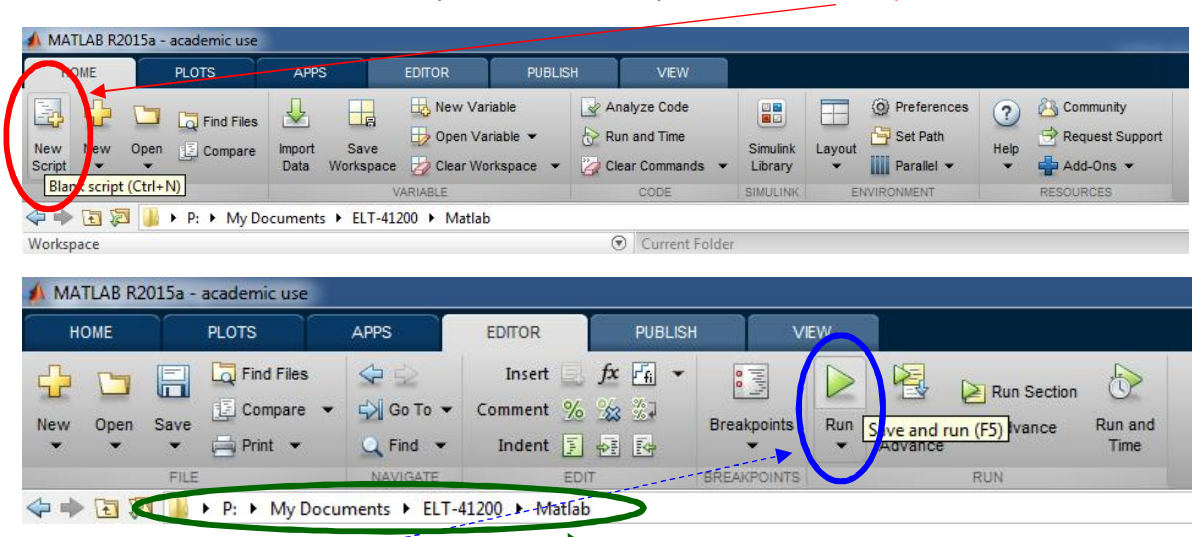
There are different types of functionalities for the M-files in Matlab, but here we consider only the script and the function:

- A script is a series of Matlab statements which are called in the given order, when the script is called (no input/output can be defined)

  >> help script
- A function, similar to script, performs a series of Matlab statements (typically some specific task), but with a listed input(s) and output(s). Also, variables used inside the function are not globally available outside the function and vice versa (local variable workspace)

  >> help function

For all the exercises in this course, you should create your own m-file script:



After you have saved the m-file: e.g. matlab_exercise_1.m, you can run it from the EDITOR-tab by pressing Run or you can run the script directly from the command line by writing the file name of the script without the extension (.m). For example,

>> matlab_exercise_1

(you can try running the example script by taking a few matrix statements from the previous page)

Notice that you should be in the same folder (called the work folder) where the called script (or function) is located, since otherwise Matlab cannot execute the code. However, you can add different folder paths to the matlab memory (help addpath), but this is left outside the scope of these exercises. Thus, make sure that you write all the executable scripts and functions in the same folder, and use that folder as the work folder. If you use the Run-button to run the script from a folder, which is not currently the work folder, Matlab will ask you to change the work folder automatically (so, you can just answer yes).