Introduction to Matlab

Objective: This exercise is an *optional* pre-exercise. You can use the exercise to check that you have the sufficient coding skills, as well as ensuring that you have your integrated development environment (IDE) setup.

The objective is to setup your Matlab IDE. Additionally, the objective is to get you acquainted with Matlab, as well as the exercise format of the course. Upon completing this exercise it is expected that you:

- Have setup your Matlab IDE.
- Can write and execute basic Matlab code.
- Understand how data can be represented as vectors and matrices in Matlab
- Understand basic operations and plotting in Matlab.
- Understand the format of the exercises.

Piazza discussion forum: You can get help by asking questions on Piazza: piazza.com/dtu.dk/fall2019/october2019

0.1 How to do the exercises

The exercises in the course are structured such that you go through an exercise document like this one.

This exercise (Exercise 0) is an optional exercise that you can use to make sure have the sufficient coding skills and have setup the your IDE—that is: the exercise is not introducing any course material, and is meant as a help in preparing for starting with the course and ensuring that you fulfill the course requirements. If you encounter any problems in doing this exercise—in setting up your IDE or installing the 02450 Toolbox—make sure to get in touch with a teaching assistant after the first lecture at the exercises. The first real course exercise (Exercise 1) will be done after the first lecture. Note that if you have difficulties following the scripts, catching up on your progamming skills are to be done as a self-study exercise (see optional material in the end of this document).

The exercises are centered around running and understanding a series of scripts provided in the 02450 Toolbox. The exercise descriptions guide you through the scripts and help you understand what is going on in them. You wont have time to code everything from scratch for the exercises, and so it is important that you get familar with this workflow centered around the existing scripts. The scripts that you go through in the exercises provide the basis for your work the project you will work on, where you will be able to re-use large parts of the code you have worked

with in the exercises. However, for the reports you will have to code more yourself and tailor the scripts to your dataset and problem.

The exercises are structured as smaller numbered sections. When a certain section concerns a particular script, it will be stated and their number will match. For instance, the first script you will run (in a little while) is called `ex0_4_3.m` and corresponds to the section 0.4.3 in this document.

0.2 Getting started with Matlab

The exercises assume that a working installation of MATLAB is already setup. If that is unavailable, MATLAB can be used from the DTU G-bar by logging in via ThinLinc.

0.3 Installing the 02450 Toolbox

The course will make use of several specialized scripts and toolboxes not included with Matlab. These are distributed as a toolbox which need to be installed.

0.3.1 Download and unzip the 02450 Toolbox for Matlab, `02450Toolbox_Matlab.zip` . It will be assumed the toolbox is unpacked to create the directories:

```
<base-dir>/02450Toolbox_Matlab/Tools/     # Misc. tools and packages
<base-dir>/02450Toolbox_Matlab/Data/      # Datasets directory
<base-dir>/02450Toolbox_Matlab/Scripts/   # Scripts for exercises
```

For the exercises, you should work on the example scripts in `<base-dir>/02450Toolbox_Matlab/Scripts/` (notice the scripts are labelled according to exercise number) and not try to write the scripts from the bottom up.

0.3.2 To finalize the installation you need to update your path. To do this run the file `<base-dir>/02450Toolbox_Matlab/setup.m` and ensure you do not get any errors.

0.4 Basic operations in Matlab

0.4.1 *Console* The IDE provides both a console and an editor. The console can be used to execute code fast and interactively, and is often used when debugging your code. You can define and display variables, plot data, and even define functions on-the-fly using interactive console. It is useful when you debug your code or experiment with small chunks of code, mathematical expressions, etc. In MATLAB, find the Command Window (the console) and try typing: `a=9`. The variable `a` should then appear in your Workspace If you write `a` in the console, the value of `a` is displayed.

0.4.2 *Editor, scripts and functions* The editor is used to write longer scripts and functions. Start the editor in MATLAB by typing `edit`. Write the following lines in the editor:

```
a = 3;
b = 4;
disp(a*b);
```

and save the file as `myscript.m`. Run the script in MATLAB by typing `myscript` in the console and observe the results. Notice that adding `;` will supress output from a given line of code. The current file can be run in MATLAB by pressing `F5` (or the green arrow), the current line or selection can be run by pressing `F9`. You can clear all variables by writing `clear` the console. You can interrupt a running script by pressing `CTRL+C`.

Make a new file `myfunction.m` and write the following function

```
function x = myfunction(n)
% myfunction will return an array of values ranging
% from 1 to the input n.
x = 1:n;
```

Note that the percentage sign `%` indicates comments and the rest of the line is not evaluated. Run the function in MATLAB with `y = myfunction(9)` and observe the `y` variable. Write at the prompt `help myfunction` and observe the results. Compare the variables `x` and `y`.

MATLAB help is obtained by typing `helpdesk` at the MATLAB command prompt or by typing `help <function name>` in the command prompt. In practice, the fastest and easiest way to get help in Matlab is often to simply Google your problem. For instance: `"How to add legends to a plot in Matlab"` or the content of an error message. In the later case, it is often helpful to find the *simplest* script or input to script which will raise the error.

0.4.3 *Making sequences* We often need to use a sequence of numbers. Observe the various results obtained when running `ex0_4_3.m`.

0.4.4 *Indexing* We often need to retrieve or assign a value to a certain part of a vector or matrix. Inspect `ex0_4_4.m` to see how to do indexing in Matlab.

0.4.5 *Matrix operations* We will use various matrix operations extensively throughout the course. Inspect `ex0_4_5.m` to see how to do some common ones in Matlab.

0.5 Basic plotting

It is important to visualize the results you obtain. In this part of the exercise, we will make two plots, a simple, and a more elaborate example. Going forwards, try to keep the elements of the figure made in 1.5.2 in mind as a model for minimum required considerations when making a figure.

0.5.1 Inspect `ex0_5_1.m` to see how to do basic plotting in Matlab. Try modifying the script to display a cosine curve for values in the range $[0, \pi]$.

0.5.2 Now, let us consider a slightly more advanced plot. Inspect `ex0_5_2.m`. We simulate measurements from two sensors (sensor 1 and sensor 2) for a period of 10 seconds, and we say that the sensors output measurents in millivolt. Since the two measurements are done at the same time, we want to do plot them in the same figure. Furthermore, we want to make sure that the axis are labelled correctly both with a name and a designation of the unit of measurement. We also need to make sure that it is easy to see which curves comes from which sensor. Lastly, we ensure that the axis are readable (large enough), both when looking at them in the IDE, but also in an exported version that we might use in a report about the sensors.

0.6 OPTIONAL

You can watch all the Matlab video tutorials (about 50 minutes of video) and carry out the operations performed in the tutorials. You can watch the tutorials by clicking on *Help* in the menu and select *Demos*.

References