2. Matlab – graphics

Hande Bozkurt, Daria Semenova & Krist V. Gernaey

February 11th, 2015

Course 28864: Introduction to Matlab programming

In the previous exercise you learned the difference between a Matlab script and a Matlab function. The purpose of this 2nd exercise is to learn to work efficiently with Matlab graphics. That means that you will learn how to use scripts and functions to make plots of your own data sets.

After you have gone through the course slides (test the commands and statements given there!) of the first and the second teaching session, you will write one Matlab script and one Matlab function in the frame of this assignment. **That means that the solution of this exercise that you will submit will consist of only 2 m-files!** Note that we know very well that one can also solve the problem below without using a function, but that is not what we would like you to do today.

You will work with two data sets that are stored as mat-files. The data sets are named Data_measurement.mat and Data_simulation.mat. The given data represents the yearly influent flow rate variations of the Bromma wastewater treatment plant (WWTP) in Sweden for the years 2009 and 2010. Each data set comprises several variables namely: cal_12, cal_365, val_12, val_365, t and tt. The first four variables represent the influent flow rate variations that were used for calibration (cal) and validation (val) of the model, with a frequency of monthly averages (12) and daily average (365) values. The remaining two variables represent the time vector with a length of 12 (t) and 365 (tt) data points. Both data sets have the same variables stored, where Data_measurement.mat file contains measured data and Data_simulation.mat file contains results of model simulations. The details of the study can be found elsewhere (Flores-Alsina et al., 2014).

You will often experience that various data sets of the same process have to be plotted in the same way in order to allow comparison. One way of organizing this is that you write a script in which you load the data and then send the data to a function which creates your plot. To keep the function applicable for multiple purposes, the plot titles and the axis labels will be defined in the script. In this case we will plot the three data sets together in one plot in order to visually compare them.

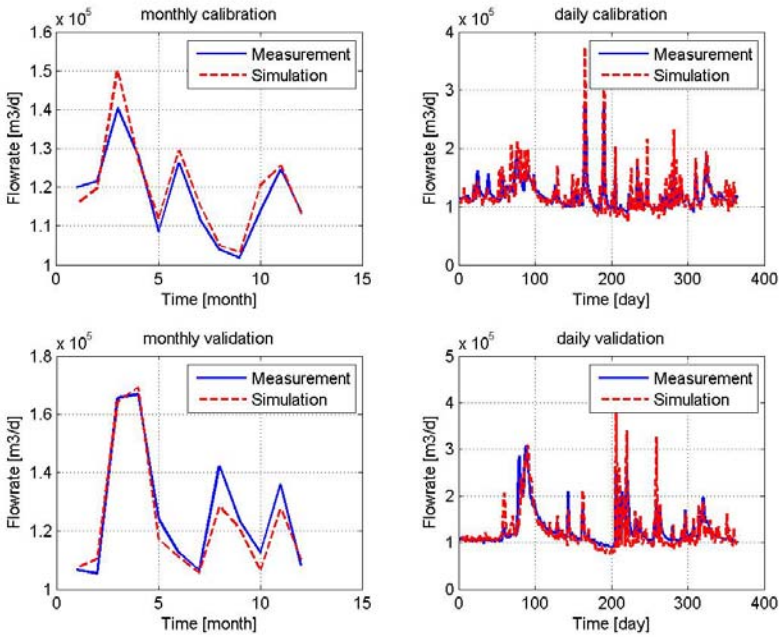
In the script:
You will start with loading the first file containing the data set to the workspace (use the command `load`).
You will assign names to the variables that you will need later on (time and flow rates), before you load the second file with data, after which you also assign names to the variables from this data set.
You will plot the measured and simulated flow rates with respect to time. In order to make this plot, you are required to send these variables as inputs to a function for generating XY-plots. You will also write that function. Other information that has to be provided in the script are the X-labels, the Y-labels, the legends and the plot titles, and all this should be sent to the function as inputs.

The function will then generate the graph and return the figure handle (See below, for a short explanation on what a figure handle is).
The script should then take this figure handle and print the figure into a file (.jpg format). To that purpose you should use the command `saveas`.

In the function:
The function receives the input data from the script and creates one figure with four plots in it (use the function "subplot" for this). It then returns the figure handle to the script. In that respect, it is important to know what a figure handle is: It is the number identifying the current figure window that is open in Matlab. When you define a figure, you typically give it a number. For example: `figure(1)`. You can obtain the value of the current figure handle using the command `gcf`.
In the figure that you will create, the top left plot should be the calibration values with monthly measurements, the top right plot should be the calibration values with daily measurements. In the third plot (bottom left) you will plot the validation values with monthly measurements and finally in the last plot (bottom right) validation values with daily measurements will be plotted. Plot the data for measurements as a blue line, and the simulation data as a red dotted line such that it is easy to distinguish between the two time series in each plot. Remember to assign the correct axis labels and figure titles.
You will need to use the commands `plot`, `xlabel`, `ylabel`, `legend`, `title`.
Furthermore, you must add some formatting to your plots. Hence the following plot formatting commands that might be useful are: (use `help plot` to find out which color codes you can use)

- *Line width:* An option of the plot command, indicating the width of the line that will be plotted in the XY plot. In this case, use LW = 1.5 for all plots
- *xlabel:* The command to put a label on the X-axis. Can be used with a text string containing the label of the X axis, and this label can be defined as an input to the function (e.g. XL = 'Time [hour]' is defined in the script and sent to the function as an input; assuming that the variable name XL is used in the function as well, one can then write the command 'xlabel(XL)') in the function to add the label to the X-axis.
- *ylabel:* The command to put a label on the Y-axis. Can be used with a text string containing the label of the Y axis, similar to the xlabel command (e.g. YL = 'Concentration [g/L]')
- *Legend:* The command to put a legend on the figure. Can be used with a text string containing the legend of the plotted data in the figure, where the legend is defined as an input to the function (e.g. Leg1='Measurement' and Leg2='Simulation' are defined in the script, and in the function one can write the command 'legend(Leg1,Leg2)', assuming in this case that we have a plot with only two time series in it)

Your result:
If you did everything correctly, then the figure should look as follows:



Handing in:
The two m-files, a script and a function, that are the solution of this exercise will be sent as e-mail attachments to kvg@kt.dtu.dk before Wednesday February 18th 2015 at 4 pm! You should not send any figures as part of the solution. If your code works correctly, your code will generate the figures!
In the email, you clearly list the names and student numbers of the members of your group (maximum 2 persons per group).
To allow automatic sorting of the emails containing your solutions, the subject line of the email containing your solution should be: 28864-F2015-E2
For help or questions: dsem@kvg.dtu.dk or kvg@kt.dtu.dk

Reference:
Flores-Alsina X., Saagi R., Lindblom E., Thirsing C., Thornberg D., Gernaey K.V. and Jeppsson U. (2014) Calibration and validation of a phenomenological influent pollutant disturbance scenario generator using full-scale data. Water Research, 51, 172-185.