System Identification – Practical Assignment 3
Transient Analysis of Impulse Responses

Logistics

- This practical assignment is a compulsory part of the course "System identification". It should be carried out by each student separately.
- The assignment solution consists of Matlab code. Develop this code in a single Matlab script. This code will be checked and run by the teacher in order to validate your attendance to the lab; the teacher will strive to do this as far as possible during the lab, together with you. Nevertheless, please write your code in a self-explanatory fashion (adding comments where necessary), so as to make it understandable on its own. At the end of the lab, please email the code as an m-file or ZIP file to the teacher (Zoltán Nagy at `zoltan.nagy@aut.utcluj.ro`, or Marius Costandin at `marius.costandin@aut.utcluj.ro`), using the following filename template:
`sysid_labN_indexINDEX_NAME`
where N is the lab number, INDEX stands for your dataset index, see below; and NAME is your last (family) name. Please *include this file name also in the subject line of your email*.
- Discussing ideas amongst the students is encouraged; however, directly sharing and borrowing pieces of code is forbidden, and any violation of this rule will lead to disqualification of the solution.

Assignment description

The last assignment dealt with step response models. In this assignment we will perform transient analysis based on nonparametric, *impulse* response models – see the course material, Part II: *Step and Impulse Response Graphical Models*. As before, we will do this for both first-order and second-order systems.

Each student is assigned an index number by the lecturer. Then, the student downloads the Matlab data files (see function `load`) that form the basis of the assignment from the course webpage:
`http://busoniu.net/teaching/sysid2017`

There are two files: the first contains several impulse inputs signals and the response of a first-order system, and the second contains similar data for a second-order system. The data is provided as an object called `data` of type `iddata` from the system identification toolbox, see `help iddata`. For convenience, a separate variable `t` holds the time vector of the experiment. Each experiment begins with 30 initial steps where the system is in its initial, steady-state condition, after which three consecutive impulse-response experiments are performed, each corresponding to 100 time steps (see the figure). Keep in mind that the initial conditions are nonzero.

Requirements (apply this procedure first for the first-order system, and then for the second-order one):

- Develop a transfer function model of the system using the method described in Lecture 3, using the *first* impulse signal and response from the data. Include instructions that print out the transfer function, as well as relevant intermediate values (e.g. overshoot M, oscillation period T_0 in the case of second order systems) at the console when your script is run.

- Validate your model using the data for the second and third impulse responses (this is the validation data). Simulate the system from the correct non-zero initial condition; to this end, create a state space model using Matlab function `ss`. The validation should consist of: (a) a plot where the system output is compared with the model output on the same graph; (b) and the computation of the MSE. Both of these results should be automatically produced by the Matlab code you provide. See function `lsim` to simulate the system response to the validation input, and investigate how you can provide the initial condition to this function.

A programmatic, code-based solution for the identification of the intermediate values (e.g. M, K, T_0) is more general and elegant. However, it is not strictly necessary, and you can also perform this step "by eye" on the graph. Especially if you see that you are running out of time, it is recommended that you apply this second, simpler solution.

One step that has to be performed programmatically in any case is the numerical computation of the areas for the second-order impulse response; you need to look a the example in the lectures to find out how to do this. Hints: always keep in mind the difference between continuous time and the corresponding indices of discrete-time steps; and watch the signs of the integrals!

Some relevant Matlab functions: `ss, lsim, find, sum`.