# **System Identification** – Practical Assignment 3
# Transient Analysis of Impulse Responses

## Logistics

This practical assignment should very preferably be carried out by each student separately. Otherwise, if there are more students than computers, students may team up in groups of 2.

The assignment solution consists of Matlab code. Develop this code in a single Matlab script.

The overall rules for labs are described on the website. For each particular lab, your attendance will only be registered if you have a working, original solution. The teacher will check that your code **works** during the lab class. Only after this has been done, for the **originality** check, upload your solution here:

`https://www.dropbox.com/request/RtpZVYbagHBGmuZg1koI`

Upload *only once, a single m-file, named exactly according to the following pattern*:

`L3_EN_G.H_LastnameFirstname.m`

where G is your group, H your halfgroup, and your last and first names follow. E.g., `L3_EN_3.1_PopAlex.m`. Any files that are duplicate, nonstandard, inappropriately named, or corresponding to unchecked solutions will not be considered. Files will be automatically tested for plagiarism, and any solution that fails this test will be marked copied; only solutions that pass both the working and the originality test are definitively validated. Therefore, while you are encouraged to discuss ideas and algorithms amongst colleagues, sharing and borrowing pieces of code is forbidden.

## Assignment description

Assignment 2 dealt with step response models. In this assignment we will perform transient analysis of *impulse* response models – see the course material, Part II: *Transient Analysis of Step and Impulse Responses*. Just like for the step response, we will do this for both first-order and second-order systems.

Each student is assigned an index number by the teacher. Then, the student downloads the Matlab data files (see function `load`) that form the basis of the assignment from the course webpage. 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 at the console the transfer function as well as relevant intermediate values (e.g. gain $K$ and time constant $T$ for first-order systems, the two areas, the overshoot $M$, the oscillation period $T_0$ for second order systems).
- 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 `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.

Hints: For the numerical computation of the areas for the second-order impulse response, you may look at the example in the lectures. 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`.