System Identification – Practical Assignment 10: Recursive identification. Correlation tests for model validation.

Logistics

- This practical assignment is a compulsory part of the course "System identification".
- The assignment is preferably carried out in groups of two, but can also be done alone if necessary. Note that groups of three or more students are strongly discouraged.
- 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 (either Lucian Busoniu at `lucian@busoniu.net` or Zoltán Nagy at `zoltan.nagy@aut.utcluj.ro`), using the following filename template:
`sysid_labN_indexINDEX_STUDENTNAME1_STUDENTNAME2`
where N is the lab number, INDEX stands for your dataset index, see below; STUDENTNAME1 and 2 stand for the last names of the two students in the group. Please *include this file name also in the subject line of your email* (for automatic email filing purposes).
- 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

In this assignment we will study a recursive system identification technique, as well as model validation. See Parts VIII and IX of the lectures: *Recursive identification methods*; *Model validation and structure selection*. Each student group is assigned an index number by the lecturer. Then, the group downloads the data file that forms the basis of the assignment from the course webpage:
`http://busoniu.net/teaching/sysid2016`
The data file contains the identification and validation data in variables `id` and `val`, as well as the **true order of the system** in variable `n`. From prior knowledge it is also known that the system is output-error type.

Part 1: Recursive identification

We will restrict ourselves to ARX models for this part. Since the system is of OE type, we will identify larger-order ARX models to account for the model type mismatch. The recommendation is to take $na = nb = 3 \cdot n$.

Requirements:

- Run recursive ARX identification (`rarx`) on the identification data, using an initial matrix $P^{-1}(0) = \frac{1}{\delta} I_{na+nb} = 100 I_{na+nb}$ (so $\delta = 0.01$). Compare the quality of two models: one with the final parameters found after processing the whole dataset; and another after only 10% of the data. Explain the differences. (Use `idpoly` to create ARX models with the correctly extracted parameters from the output of `rarx`, see also the hints below.)

- Repeat the experiment with $P^{-1}(0) = \frac{1}{\delta} I_{na+nb} = 0.01 I_{na+nb}$ (so $\delta = 100$). Consider the results. For which value of δ is the early model worse, and why?

Part 2: Model validation

The setting is different from Part 1, as follows. We will use batch, offline identification, and we will take the orders of the polynomials equal to the known system order n, accepting the risk of incorrect models – in fact, the goal of this part is exactly to learn how to check their correctness using residual tests.

Requirements:

- Identify an ARX model with the orders $na = nb = n$ equal to the system order available from prior knowledge. Use `resid` to perform correlation tests and determine whether the model is good. Also validate the model in simulation to confirm the results.

- Identify an OE model with orders $nb = nf = n$. Use `resid` to perform correlation tests and determine whether the model is good. Also validate the model in simulation to confirm the results. Discuss the results with OE in comparison to the ARX model.

Relevant functions from the System Identification toolbox: `rarx`, `idpoly`, `resid`, `plot`, `compare`.
Hints:

- The quantity that the help text of the `rarx` function denotes `P` is actually the *inverse* matrix P^{-1} from the lecture, so be careful when setting it.

- `rarx` returns a matrix of parameters, each row k containing the parameters obtained by running recursive ARX on the data up to and including k. The row contains first the coefficients $a_1, \ldots, a_{na}$ of A, and then the coefficients $b_1, \ldots, b_{nb}$ of B, *not including* the leading coefficients 1 and 0, see below.

- `idpoly` requires explicitly setting the leading 1 coefficient of the polynomial A, and the leading 0 coefficient of the polynomial B. To avoid warnings, it is also recommended to explicitly set the sampling time of the model to that of the data.