System Identification – Practical Assignment 10: Recursive identification

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. 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 at `zoltan.nagy@aut.utcluj.ro`, or Marius 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

In this assignment we will study the recursive variant of the ARX method, see Part 9 of the lectures: *Recursive identification methods*.

Each student is assigned an index number by the lecturer. Then, the student downloads the data file that forms the basis of the assignment from the course webpage:
`http://busoniu.net/teaching/sysid2017`

The file contains the identification data in variable `id`, and separately the validation data in variable `val`. From prior knowledge, it is known that the system has order n, given in variable `n` in the data file; that it is of the output error (OE) type; and that it has no time delay.

In the first part of this lab, your task is to implement the general recursive ARX algorithm in a Matlab function. This function should take at the input the identification dataset, the model orders na and nb, the initial parameter vector $\theta(0)$, and the initial inverse matrix $P^{-1}(0)$. The function should produce at the output a matrix $\Theta \in \mathbb{R}^{N\times(na+nb)}$ containing on each row k the parameter vector $\theta(k)$: first the coefficients $a_1, \ldots, a_{na}$ of A, and then the coefficients $b_1, \ldots, b_{nb}$ of B. (This is compatible with the output of the Matlab function, so it will be easier to compare later).

Since the system is of OE type, to account for the model mismatch we will take larger orders for for all the ARX models to be found. The recommendation is to take $na = nb = 3 \cdot n$. Using these orders, for the second part of the lab:

- Run recursive ARX identification with the function you implemented, 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 on the validation

data 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.

- 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?

- Repeat the initial experiment, but now with the already available Matlab function `rarx`. Compare the results it gives (e.g., the 10% and 100% models) with those given by your own function, to verify whether they are the same or similar.

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

- Once you have your polynomials A and B as vectors of coefficients in increasing powers of q^{-1}, use `idpoly(A, B, [], [], [], 0, Ts)` to generate the ARX model, where `Ts` is the sampling period. Do not forget that all vectors of polynomial coefficients must always contain the leading constant coefficients (power 0 of the argument q^{-1}), which must be 1 in A and 0 in B. Keep in mind that the matrix of parameters returned by the algorithm does *not* contain these leading coefficients.

- The quantity denoted `P` by the `rarx` function documentation is actually the *inverse* matrix P^{-1} from the lecture, so be careful when setting it. Don't forget to also configure the algorithm of `rarx` by using the `'ff', 1` arguments.

- `rarx` returns a matrix of parameters with exactly the format asked above for your function.