System Identification – Practical Assignment 6
ARX model identification

Logistics are as before, see previous labs.

Assignment description

In this assignment we will identify ARX models (autoregressive with exogenous input), using least-squares, linear regression. See the course material, Part V: *ARX Identification*.

You will develop a function with the exact signature:

```
[index, PHI, theta, ypred, ysim] = findarx
```

Each student is assigned an index number in the set 1-8, which needs to be saved to variable `index` at the beginning of the function. The index dictates which data file the student should load. For instance, if you have index 3, you load file `lab6_3.mat`. All these datafiles are already accessible from your function code. Each file contains the identification data in variable `id`, and the validation data in variable `val`.

It is known from prior knowledge that the system does not have any time delay, and is of order at most 3.

Requirements:

- Plot and examine the data supplied.
- Implement ARX identification explicitly using linear regression, as described in the lecture. Recall that the regressors are $-y(k-1), \dots, -y(k-na), u(k-1), \dots, u(k-nb)$. Your code should preferably work for any values of na and nb. Use the same order of regressors as above (therefore, the parameters will be in the order $a_1, \dots, a_{na}, b_1, \dots, b_{nb}$).
- To allow for the possibility of disturbances that do not satisfy the ARX model structure, we will take $na = nb = 9$, three times larger than the largest possible order. Identify an ARX model with these orders. Return the matrix of regressors in `PHI`, and the resulting parameter vector in `theta`. (Validating `PHI` will ensure that you are building the system of equations right.) Before returning `theta`, display its value at the console (after issuing `format long`).
- Implement prediction and simulation with the computed model on the validation data. Keep in mind that for simulation, knowledge about the real outputs of the system is not available, so we can only use previous outputs of the model itself; in particular $y(k-i)$ in the model formula must be replaced by its previously simulated value $\hat{y}(k-i)$, for $i = 1, \dots, na$. Plot the predicted and simulated output against the real one, and compute the MSEs in prediction and validation. Return the predicted and simulated outputs in `ypred`, `ysim` respectively.
- Optionally, if you still have time – or if you have bugs and want a known good solution – identify models with the same values of na, nb as above, but this time with the Matlab `arx` function. Compare the results with those that you obtained using your code, and verify that the two results are similar.

Please make sure to include your plots, even if they are not validated explicitly in the tests (we **will** use them to evaluate solutions).

Important: Signals at negative or zero time steps should be taken equal to zero. **Hints**: When you predict and simulate on the validation data, do it iteratively for each time step. To reduce the probability of timeouts, instead of working with objects and structures (e.g. `id.y`), save the data directly into arrays (e.g. `yid`), as Matlab is much faster with arrays.

Relevant functions from the System Identification toolbox: `arx`, `plot`, `compare`. When the `ident` toolbox function has the same name as a function in another toolbox – like in the case of `compare`, which overloads the MPC toolbox implementation – write e.g. `doc ident/compare` to get the documentation of the `ident` variant. See also `doc ident` for the full documentation of the toolbox.