System Identification – Practical Assignment 8
Prediction error methods

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

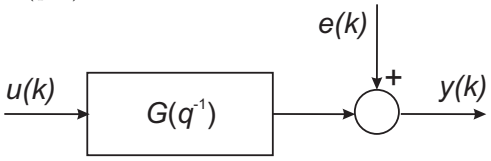
In this assignment we will identify and compare OE, ARMAX, and ARX models, using prediction error methods. See the course material, Part VII: *General Prediction Error Methods*.

Each student is assigned an index number by the lecturer. Then, the student downloads the files that form the basis of the assignment from the course webpage:
`http://busoniu.net/teaching/sysid2017`
There is a datafile, containing the identification data in variable `id`, and separately the validation data in variable `val`. Both of these variables are objects of type `iddata` from the system identification toolbox of Matlab, see `doc iddata`. A function `oeidentify` is also made available.

From prior knowledge, it is known that the system is in the output error (OE) form:

$$y(k) = \frac{B(q^{-1})}{F(q^{-1})}u(k) + e(k)$$

see also the figure, where $G = \frac{B(q^{-1})}{F(q^{-1})}$.



So, the system is only affected by white measurement noise $e(k)$ on its output. It is also known that the system has no time delay. **The order n of the system is given in Matlab variable** `n` in your datafile.

The first part of this lab will consist in computing the prediction error for such an output error model structure. Recall from the lecture that the general formula to compute the prediction error is:

$$\varepsilon(k) = H^{-1}(q^{-1})(y(k) - G(q^{-1})u(k))$$

For OE, $G(q^{-1}) = \frac{B(q^{-1})}{F(q^{-1})}$ and $H(q^{-1}) = 1$. With this in mind, write a function that has the following signature:

```
V = prederr(theta, id, nb, nf)
```

This function must compute the prediction error $\varepsilon(k)$ for $k = 1, \ldots, N$, and return in V the mean squared error, $V = \frac{1}{N}\sum_{k=1}^{N} \varepsilon^2(k)$. This is done for a given parameter vector $\theta = [f_1, \ldots, f_{nf}, b_1, \ldots, n_{nb}]^\top$, on any data set `id`, and for any model orders nb and nf. Hints: write down the formula for the prediction error on paper first; the prediction error will be the result of simulating a dynamical system. You do not have to name the function and variables exactly as above, but the arguments must have these meanings.

Once you have written and tested your function, you can use it with the supplied function:

```
oem = oeidentify('prederr', id, nb, nf)
```

which will find a parameter vector that (approximately) minimizes the mean squared error, and directly creates an OE model `oem` using this vector. Here, replace 'prederr' by the name of your own function.

In the second part, we will use the code above to identify an OE model and compare it to ARMAX and ARX.

- Given the known system structure, take nf and nb appropriately and call the optimizer function with these values to find an OE model. Validate this model on the validation data.
- Repeat the experiment but now with the already available Matlab function `oe`. Compare the two models, keeping in mind that the optimizers used by `oe` and `oeidentify` are different so they may produce different results. In that case, use the better model of the two at the next steps.
- Choose orders na, nb, nc for an ARX and ARMAX model, and call `arx` and `armax` to find the models. Do not use `arxstruc` or other automatic model selection functions, but select the orders manually based on your experience and the prior knowledge about the system.
- Compare the OE, ARMAX, and ARX models. Does the system also obey the ARMAX form? What about ARX? Compare the fit scores of the three models as given by `compare`. If two model scores are similar, consider also the model complexity. What is the best model, when fit quality, model complexity, and system structure are all taken into account?

Relevant functions from the System Identification toolbox: `arx`, `armax`, `oe`, `plot`, `compare`. If you have a model called e.g. `mod`, you can just type `mod` at the command line to investigate its structure, and its component polynomials are accessible as named fields, e.g. `mod.A`, `mod.B`. See also `doc ident` for the full documentation of the toolbox.