System Identification – Practical Assignment 9
Instrumental variable methods

Logistics are as before, see previous labs.

You will develop a function with the exact signature:

```
[index, Z10, phi10, theta] = ividentify
```

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 `lab9_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`. Hint: to increase execution speed and avoid timeouts, save the vectors of inputs and outputs to separate arrays instead of always accessing the `id` object.

From prior knowledge, it is known that the system is of the order given in variable `n` in the data file; and that the disturbance is not white noise, but colored. Thus, for all models we will use $na = nb =$`n`, the value from the datafile.

Your task is to implement the IV algorithm using instruments based on ARX-outputs. To solve the identification problem efficiently in Matlab, it will be useful to rewrite the IV system of equations in a form amenable to matrix left division. To that end, let us take equation (8.3) from the lecture slides and rewrite it as:

$$\left[\frac{1}{N}\sum_{k=1}^{N} Z(k)\varphi^T(k)\right]\theta = \frac{1}{N}\sum_{k=1}^{N} Z(k)y(k)$$

$$\text{or equivalently: } \tilde{\Phi}\theta = \tilde{Y}$$

where the $(na+nb)\times(na+nb)$ matrix $\tilde{\Phi} = \frac{1}{N}\sum_{k=1}^{N} Z(k)\varphi^T(k)$ and the $(na+nb)\times 1$ vector $\tilde{Y} = \frac{1}{N}\sum_{k=1}^{N} Z(k)y(k)$. Note the tildes, which signify that these quantities are variants of the regressors and of the original system outputs, "modified" by the IVs.

In the equation above, the instrument vector is:

$$Z(k) = [-\hat{y}(k-1), \dots, -\hat{y}(k-na), u(k-1), \dots, u(k-nb)]^T$$

where the outputs $\hat{y}$ are **simulated** with the ARX model found earlier. Do not use predicted outputs, as those are correlated with the disturbance and will likely break the IV method!

Requirements:

- Using Matlab function `arx`, identify an ARX model of orders $na = nb =$`n` and inspect its quality on the validation dataset.
- Implement the IV algorithm stated above, to find a model with the same na and nb. As an intermediate verification step, return $Z(10)$ and $\varphi(10)$ in variables `Z10` and `phi10`. Finally, return the resulting parameter vector θ in `theta`.
- Create an IV model in the `idpoly` format using θ, and compare the quality of this IV model with that of the original ARX model, on the validation dataset, using `compare`.

Hints:

- For simplicity, you can fill in the vectors Z directly from the simulated ARX outputs $\hat{y}$, rather than defining polynomials C and D. You can use Matlab functions to compute $\hat{y}$.
- Always ensure that you are working with the right-sized vectors and matrices. In particular, Z, φ, and $\tilde{Y}$ must be $(na+nb)\times 1$ vectors; whereas $\tilde{\Phi}$ and each of its inner summation terms $Z(k)\varphi^T(k)$ must be $(na+nb)\times(na+nb)$ matrices.
- Solve the matrix equation using $\backslash$.
- 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 IV model, where `Ts` is the sampling period. Do not forget that all vectors of polynomial coefficients must always contain the leading constant coefficients (i.e. for power 0 of the argument q^{-1}), which must be 1 in A, and 0 in B.
- You can compare the quality of several models at once, using the syntax: `compare(val, model1, model2, ...)`.