System Identification – Practical Assignment 8
Output error identification using the Gauss-Newton method

Logistics

This practical assignment should very preferably be carried out by each student separately. Otherwise, if there are more students than computers, *with the explicit agreement of the lab teacher for each group*, students may team up in groups of 2.

The assignment solution consists of Matlab code. Develop this code in a single Matlab script. If you need to create functions, they can be local to the script, see local functions in scripts.

The overall rules for labs are described on the website. For each particular lab, your attendance will only be registered if you have a working, original solution. The teacher will check that your code **works** during the lab class. Only after this has been done, for the **originality** check, upload your solution here:

`https://www.dropbox.com/request/ApI3XWD1aQmX0c38xYYz`

Upload *only once, a single m-file, named exactly according to the following pattern*:

`L8_ENgh_LastnameFirstname.m`

where `g` is the group, `h` the halfgroup, and the last and first names follow. E.g., `L8_EN32_PopAlex.m`. *If you worked in a group per the above, only upload one file with both student names included*. Any files that are duplicate, nonstandard, inappropriately named, or corresponding to unchecked solutions will not be considered. Files will be automatically tested for plagiarism, and any solution that fails this test will be marked copied; only solutions that pass both the working and the originality test are definitively validated. Therefore, while you are encouraged to discuss ideas and algorithms amongst colleagues, sharing and borrowing pieces of code is forbidden.

Assignment description

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. 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`.

From prior knowledge, it is known that the system is first-order, without time delays, and only affected by measurement noise $e(k)$ at the output. This means that the following Output Error form is appropriate to model it:

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

with the parameters $\theta = [f, b]^\top$. Our objective will be to implement the prediction error method for this particular model structure, using Gauss-Newton optimization. The algorithm is summarized at the end of this description, in a more direct way than in the lectures, so as to help with implementation.

Requirements:

- Compute the recursion formulas required by the algorithm, on paper or at the whiteboard. Hint: follow the line of derivations from the first-order ARMAX case exemplified in the lectures.
- Implement the algorithm, and run it on the identification data starting from *nonzero* initial parameters.

- For the near-optimal values of f and b obtained, create an OE model in the system identification toolbox format, using `idpoly`. Note that the syntax of this function is `idpoly(A,B,C,D,F,0,Ts)` where you need to specify the leading zero in B, the leading 1 in F, and the sampling time can be found e.g. in the identification dataset. The polynomials that you don't use can be set to 1. Use `compare` to see how the model performs on the validation data.
- If the model is not satisfactory, tune α, δ and $\ell_{\max}$ (as well as perhaps θ_0), so as to improve performance.

---

choose stepsize α, initial parameters θ_0, threshold δ, and max iterations $\ell_{\max}$
initialize iteration counter $\ell = 0$
compute recursion formulas for $\varepsilon(k)$, $\frac{d\varepsilon(k)}{d\theta} = [\frac{d\varepsilon(k)}{df}, \frac{d\varepsilon(k)}{db}]^\top$
repeat
  with the current parameters θ_ℓ:
    simulate (apply recursion formulas) to find $\varepsilon(k)$, $\frac{d\varepsilon(k)}{d\theta}$, for $k = 1, \ldots, N$
    compute gradient of the objective function, $\frac{dV}{d\theta} = \frac{2}{N}\sum_{k=1}^{N} \varepsilon(k)\frac{d\varepsilon(k)}{d\theta}$
    compute approximate Hessian of the objective function, $\mathcal{H} = \frac{2}{N}\sum_{k=1}^{N} \frac{d\varepsilon(k)}{d\theta}\left[\frac{d\varepsilon(k)}{d\theta}\right]^\top$
  apply Gauss-Newton update formula: $\theta_{\ell+1} = \theta_\ell - \alpha\mathcal{H}^{-1}\frac{dV}{d\theta}$
  increment counter: $\ell = \ell + 1$
until $\|\theta_\ell - \theta_{\ell-1}\| \leq \delta$, or $\ell_{\max}$ was reached

---