

System Identification – Practical Assignment 8

Identifying output error models using the Gauss-Newton method

Logistics

Please reread the logistics part of lab 2, the same rules will apply to this lab. The only thing that changes is the dropbox link, which for this lab is:

<https://www.dropbox.com/request/QiI4tZw6Z1klw0GDUKqs>

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 $\theta = [f, b]^T$. 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 ℓ_{\max} (as well as perhaps θ_0), so as to improve performance.

choose stepsize α , initial parameters θ_0 , threshold δ , and max iterations ℓ_{\max}

initialize iteration counter $\ell = 0$

compute recursion formulas for $\varepsilon(k)$, $\frac{d\varepsilon(k)}{d\theta} = \left[\frac{d\varepsilon(k)}{df}, \frac{d\varepsilon(k)}{db} \right]^\top$

repeat

with the current parameters θ_ℓ :

simulate (apply recursion formulas) to find $\varepsilon(k)$, $\frac{d\varepsilon(k)}{d\theta}$, for $k = 1, \dots, N$,

starting from $\varepsilon(0) = 0$, $\frac{d\varepsilon(0)}{d\theta} = [0, 0]^\top$

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 ℓ_{\max} was reached
