Project Assignment
System Identification 2018

Logistics

This MATLAB-based project assignment is a compulsory part of the System Identification course in the Control Engineering B.Sc. program of the Technical University of Cluj-Napoca. It will be graded and the mark counts for 30% in the final grade of the course. The assignment is carried out in groups of two students, and should take around 20 hours per person to solve, depending on your experience with MATLAB. Each group will receive their own data sets. To receive them, form groups and send as soon as possible an e-mail to the lecturer Lucian Buşoniu at `lucian@busoniu.net`. Please, mention the names of the two members of your group and their e-mail addresses. The assignment must be worked out in the form of a short written report (in English, one report per group).

Please email the report **in PDF format** to the lecturer at the same address. Do not forget to include your names on the title page of the report. It is required to include in the report complete listings of the MATLAB code (functions and scripts) that you developed for solving the assignment problems. In addition to that, your code must also be sent as a ZIP file, with the same email in which you send the report.

The deadline for the full report and code is **January 13th 2019, 24:00 (midnight) at the latest**. In addition, an intermediary version of the report and code with the first problem solution must be sent by **November 18th 2018, 24:00**. In case of delays, each newly entered day of delay results in a 2 point decrease in the maximum grade for the assignment (for instance, delivering the full report on December January 15th at 00:10 AM leads to a maximum grade of 6 since the second day of delay has been entered). **Crucial rule:** it is strictly forbidden to copy code, text, or results from other students or from online resources. There will be absolutely zero tolerance for copying, and failing to obey this rule automatically and immediately leads to ineligibility for the exam. So, be extremely careful!

The assignment consists of two problems. In the first problem, the polynomial is used to model the behavior of an unknown function. The second problem concerns nonlinear ARX identification of an unknown dynamical system.

Part 1. Fitting an unknown function

A data set of input-output pairs is given, where the outputs are generated by an unknown, nonlinear but static function f. The outputs are corrupted by noise, which is assumed to be additive and zero-mean Gaussian. The function has two input variables and one output variable. You will have to develop a model for this function, using a polynomial approximator. A second data set generated using the same function is provided for validating the developed model. The two data sets will be given as a MATLAB data file, containing one structure for each set. The training data structure is named `id` and the validation data structure `val`. Each of these structures contains the following fields:

- A set of grid coordinates `X` for the inputs, where `X` is a cell array of two vectors, each vector `X{1}(i),X{2}` containing n grid points for input dimension `dim`.
- A set of corresponding outputs `Y`, a matrix of size $n \times n$, where `Y(i,j)` is equal to the value of f at point `(X{1}(i),X{2}(j))`.

Your polynomial approximator g should have a configurable degree m. For example, for the first few values of m, the approximator has the form:

$$m = 1, \quad \hat{g}(x) = [1, x_1, x_2] \cdot \theta = \theta_1 + \theta_2 x_1 + \theta_3 x_2$$
$$m = 2, \quad \hat{g}(x) = [1, x_1, x_2, x_1^2, x_2^2, x_1 x_2] \cdot \theta = \theta_1 + \theta_2 x_1 + \theta_3 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \theta_6 x_1 x_2$$
$$m = 3, \quad \hat{g}(x) = [1, x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3, x_1 x_2, x_1^2 x_2, x_1 x_2^2] \cdot \theta$$

where the first two polynomials have been made explicit for clarity.

Once the degree m has been chosen, model fitting consists of finding the optimal parameter vector θ so that $g(x)$ best matches $f(x)$ on the identification dataset, in a least-squares sense. This can be done with linear regression, keeping in mind that g is linear in the parameters (even though it is nonlinear in the variables x). Details can be found in the lectures, Part 3: *Mathematical Background*, see the linear regression sections. The regressors for the polynomial case here are the powers of x_1 and x_2, e.g. for $m = 2$ they are $1, x_1, x_2, x_1^2, x_2^2, x_1x_2$.

The **requirements** are given next. Program such an approximator of configurable degree m. Try to fit approximators of varying degrees, so as to obtain the most accurate one. Validation should always be performed on the different, validation dataset. Report the mean squared errors for both sets and show a representative plot for the fit on the training and the validation data sets (true values compared to approximator outputs). Discuss the results, including the choice of degree and the quality of the model fit on the two data sets, relating them to the discussion during lectures on model choice and overfitting in regression.

While MATLAB can automate polynomial regression, you are required to code the approximator and regression procedure on your own. For some inspiration you can look at your solution to practical assignment 4, *Linear regression for function approximation*.

Part 2. Nonlinear ARX identification

Before working on this part of the project, you will need background on *linear* ARX models. Part 5 of the lectures, *ARX identification*, focuses on this.

A dataset is given, measured on an unknown **dynamic system** with one input and one output. The order of the dynamics is not larger than three, and the dynamics may be nonlinear while the output may be affected by noise. Your task is to develop a black-box model for this system, using a polynomial, nonlinear ARX model. A second data set measured on the same system is provided for validating the developed model. The two data sets will be given in a MATLAB data file, with variables `id` and `val` containing the two sets as objects of type `iddata` from the System Identification toolbox. Recall that the input, output, and sampling time are available on fields `u, y, Ts` respectively. As a backup in case the system identification toolbox is not installed on the computer, `id_array` and `val_array` contain the same two datasets but now in an array format, with the structure: time values on the first column, input on the second column, and output on the last column.

Consider model orders na, nb, and delay nk, following the convention of the `arx` MATLAB function. Then, the nonlinear ARX model is:

$$\begin{aligned}\hat{y}(k) &= p(y(k-1), \dots, y(k-na), u(k-nk), u(k-nk-1), \dots, u(k-nk-nb+1)) \\ &= p(d(k))\end{aligned} \tag{1}$$

where the vector of delayed outputs and inputs is denoted by $d(k) = [y(k-1), \dots, y(k-na), u(k-nk), u(k-nk-1), \dots, u(k-nk-nb+1)]^T$, and p is a polynomial of degree m in these variables.

For instance, if $na = nb = nk = 1$, then $d = [y(k-1), u(k-1)]^T$, and if we take degree $m = 2$, we can write the polynomial explicitly as:

$$y(k) = ay(k-1) + bu(k-1) + cy(k-1)^2 + vu(k-1)^2 + wu(k-1)y(k-1) + z \tag{2}$$

where a, b, c, v, w, z are real coefficients, and the parameters of the model. Note that the model is nonlinear, since it contains squares and products of delayed variables (as opposed to the ARX model which would only contain the terms linear in $y(k-1)$ and $u(k-1)$). Crucially however, the model is still linear in the parameters so linear regression can still be used to identify these parameters.

Note that the linear ARX form is a special case of the general form (1), obtained by taking degree $m = 1$, which leads to:

$$\hat{y}(k) = ay(k-1) + bu(k-1) + c$$

and further imposing that the free term $c = 0$ (without this condition, the model would be called affine).

The **requirements** follow. Code a function that generates such an ARX model, for configurable model orders na, nb, delay nk, and polynomial degree m. Code also the linear regression procedure to identify the parameters, and the usage of the model on arbitrary input data. Note that the model should be usable in two modes:

- One-step-ahead prediction, which uses knowledge of the real delayed outputs of the system; in the example, we would apply (2) at step k with variables $y(k-1), u(k-1)$ on the right-hand side.
- Simulation, in which knowledge about the real outputs is not available, so we can only use previous outputs of the model itself; in the example we would replace $y(k-1)$ on the right-hand side of (2) by the previously simulated value $\hat{y}(k-1)$.

Identify such a nonlinear ARX model using the identification data, and validate it on the validation data. Choose carefully the model orders and the delay, as well as the polynomial degree. To reduce the search space you may take $na = nb$. Report the one-step-ahead prediction error, and the simulation error for both the identification and the validation sets (use the mean squared error). Show a representative plot for the fit on the training and the validation data sets, for both simulation and prediction. Discuss the results, including the quality of the model fit on the two data sets.

The presentation above is self-contained, but you may e.g. look at the following papers for additional technical insight:

1. H. Peng et al., *RBF-ARX model-based nonlinear system modeling and predictive control with application to a NOx decomposition process*, Control Engineering Practice 12, pages 191–203, 2007. Here the model is explained in Sections 2.1-2.2, and uses tunable radial basis functions instead of polynomials.

2. L. Ljung, *System Identification*, Wiley Encyclopedia of Electrical and Electronics Engineering, 2007. Available as technical report LiTH-ISY-R-2809. See Section 4 for nonlinear models, again mainly using basis functions.

Matlab programming and other remarks

Strive for a compact and elegant MATLAB code, avoid the use of loops (`for`, `while`, etc.) and `if-then-else` constructs where vector operations would be easier and more readable. Search for "vectorization" in the MATLAB help system for helpful tips on the proper MATLAB programming style. However, do not exaggerate with applying vectorization: if the code is clearer with loops or if statements, use them.

If you are less familiar with programming in MATLAB, the following pointers may help. Type `doc` at the command line to access the documentation. A good initial read is the *Getting Started with Matlab* node of the documentation. *Matrices and Arrays*, *Programming Basics*, and *Plotting Basics* are also useful.

It is preferable that each group brings their own laptop to the project classes, and always works on the same laptop for the project. This is to prevent incompatibilities between the MATLAB versions installed on your computers and the University computers. If this is not possible for you, please discuss with the lecturer.