Project Assignment
System Identification 2026-2027

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 (15% for part 1, and 15% for part 2). The assignment is carried out in groups of **three** students. Project class attendance is mandatory, and in addition to that the project should take around 10 hours per person to solve, depending on your experience with MATLAB. Each group will receive their own datasets. To receive them, form groups and send ASAP an e-mail to the project TA. Please, mention the name and email address of each member of the group. You will also receive a unique group index, visible in the online status table.

The assignment consists of two problems. In the first problem, a polynomial approximator is used to model the behavior of an unknown, static function. The second problem concerns nonlinear ARX identification of an unknown dynamical system, using again polynomial approximation. Each part is detailed below. **Crucial rule:** it is strictly forbidden to copy code, text, or results from other students or from online resources such as generative AI. Careful scrutiny as well as automated tools are in place to check this, and there will be absolutely zero tolerance for copying: failing to obey this rule automatically and immediately leads to ineligibility for the exam.

Part 1. Fitting an unknown function

A dataset 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 dataset generated using the same function is provided for validating the developed model. The two datasets 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)`).

We will create a polynomial approximator g of the function f, with a configurable degree m. For example, for the first few values of m, the approximator has the form:

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

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, $x_1 x_2$.

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 (also for the purpose of comparing e.g. different values of m) 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 datasets (true values compared to approximator outputs). *Discuss* the results, including the choice of degree and the quality of the model fit on the two datasets, 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, also because you will need the insight in the second part of the project. For some inspiration you can also look at your solution to lab assignment 2, *Linear regression for function approximation*.

Evaluation of part 1

The complete solution of part 1 consists of both code and a presentation (slides), which will be defended in an interactive session consisting of the oral presentation as well as a question-answering part. The presentations will take place during week 7 of the semester. The exact program (which group presents when) will be communicated a sufficient time in advance. Each group gets 20 minutes, out of which at most 6 are allocated for the presentation itself, and at least 14 minutes for questions. Be available 20 minutes in advance. All three members of the group must attend and answer questions; excepting force majeure, any student who is absent will fail the project and thus be ineligibile for the exam. The grade will consider three aspects: the solution itself (code and results); presentation; and question answers. Questions will be targeted to identify any plagiarism attempts and to differentiate the contribution of each group member, so each student will likely get a different grade.

Your presentation must be written coherently and concisely, and must include at least the following elements:

- A problem statement.
- A very brief description of the approximator structure, and the procedure to find the parameters.
- Any *key* feature(s) of your own individual solution (do not include trivial implementation details).
- Tuning results (at least the MSE as a function of m, either as a graph or a table).
- The representative plots pointed out above, for the optimal value of m.
- The discussion points pointed out above, and an overall conclusion.
- An appendix with listings of your code. This should **not** be discussed during the 6 minutes, but we will ask questions about it.

Since you have 6 minutes for this, as a rule of thumb, each point above should take on average 1 minute and on the order of 1 slide (may be more depending on the density of information on the slide, but definitely not much more).

The deadline for submitting your solution is **November 3rd 2026, 24:00**. In case of delays, each newly entered day of delay results in a 2 point decrease in the maximum grade (for instance, delivering the solution on November 5th at 00:01 AM leads to a maximum grade of 6 since the second day of delay has been entered).

Please **pay attention and follow to the letter** the following rules for delivery. A uniformized, semi-automated processing of solutions is essential for efficient grading, and any deviation from the rules

makes your submission require additional, manual processing time, which may be unavailable and may therefore mean that your solution cannot be graded!

- Your submission must consist of at least two files, named exactly like this: LN1LN2LN3.pdf, and LN1LN2LN3.zip, where LNi is the last (family) name of student "i" in your project group, without diacritics. For example: IonescuFarkasBonta.pdf and IonescuFarkasBonta.zip.
- The first file is the presentation; only the **PDF** format is acceptable, although optionally you might also include a PPT or PPTX presentation if you wish, and it will be used if possible during the session. Please make sure that you send a presentation (slides)! Do **not** send a report instead.
- For identification purposes, the first slide of the presentation should prominently include the names and unique index of your group.
- The second file contains your code, sent as a **ZIP archive** (classical ZIP; no other formats like RAR or 7Z are accepted). Important: there should be no subdirectories in this archive, all the m-files must be top-level.
- These files will be submitted via a DropBox file request. Do not submit multiple versions as these cannot be taken into account; only your first submission will be considered, so make sure it is complete and correct. Submit only one set of files per group; establish in advance who among the students in the group will handle the submission.

Part 2. Nonlinear ARX identification

To work on this part of the project, you need background on *linear* ARX models. The lecture *ARX identification*, together with the corresponding lab, focus 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 dataset measured on the same system is provided for validating the developed model. The two datasets 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:

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

where the vector of delayed outputs and inputs is denoted by $d(k) = [y(k-1),\ldots,y(k-na),u(k-nk),u(k-nk-1),\ldots,u(k-nk-nb+1)]^T$, 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 (**??**), 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 and polynomial degree m; the delay nk can be taken equal to 1 to simplify things. Code also the linear regression procedure to identify the parameters, and the usage of the model on arbitrary input data. Note that the model must be usable in two modes:

- One-step-ahead prediction $\hat{y}$, which uses knowledge of the real delayed outputs of the system; in the example, we would apply (**??**) at step k with variables $y(k-1), u(k-1)$ on the right-hand side.
- Simulation $\tilde{y}$, 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 (**??**) by the previously simulated value $\tilde{y}(k-1)$.

Identify such a nonlinear ARX model using the identification data, and validate it on the validation data. Choose carefully the model orders and the polynomial degree. To reduce the search space you may take $na = nb$. Your presentation should include at least the following elements:

- An problem statement, introductory slide.
- A brief description of the approximator structure, and the procedure to find the parameters.
- Any *key* features of your own individual solution (do not include trivial implementation details).
- Tuning results (at least the MSE as a function of $na = nb$ and of m, either as a table or a graph).
- For the best model obtained above, representative plots with the approximated model output versus the real output, in both simulation and prediction, for the identification and validation datasets (the plots for one dataset can be given in the same graph, in different colors).
- The one-step-ahead prediction error and the simulation error, for both the identification and the validation sets (use the mean squared error).
- A brief discussion of the results, including the quality of the model fit on the two datasets.
- An appendix with listings of your code. This should **not** be discussed during the 6 minutes, but we will ask questions about it.

The task description 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.

Evaluation of part 2

There is no intermediate evaluation for Part 2. A complete solution to Part 2 will be presented orally, following the same timing as for part 1 (max 6 minutes for slides, min 14 minutes for questions, for a total of 20 minutes), in week 14 of the semester. Excepting force majeure, any student who is absent will fail the project and thus be ineligibile for the exam. The deadline for the presentation as well as the associated code is **December 19th 2026, 24:00**. In case of delays, each newly entered day of delay results in a 2 point decrease in the maximum grade, as for part 1 above.

Please follow accurately the following rules for delivery. They closely parallel the rules for Part 1.

- Your submission must consist of at exactly two files, named exactly like this: LN1LN2LN3.pdf, and LN1LN2LN3.zip, where LNi is the last (family) name of student "i" in your project group, without diacritics. For example: IonescuFarkasBonta.pdf and IonescuFarkasBonta.zip. Unlike part 1, PPT or PPTX are no longer acceptable. Please make sure that you send a presentation (slides)! Do not send a report instead.
- The first slide of the presentation should prominently include the names and unique index of your group.
- The second file contains your code itself, sent as a classical **ZIP archive** (no other formats accepted). Important: there should be no subdirectories in this archive, all the m-files must be top-level.
- The files will be submitted via a Teams assignment by one student in the group, without duplication.

Matlab programming and other remarks

If you are less familiar with programming in MATLAB, the following pointers may help (besides of course everything you learn in the labs). 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.

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.