System Identification – Practical Assignment 4
Linear Regression for Function Approximaton

Logistics

- Development: This practical assignment should be carried out by each student separately, on the Matlab Grader platform. You should have received an invite to the Grader platform on the email address communicated to the teachers. The assignment solution consists of Matlab code. Develop this code in a single Matlab function. You can pretest your solution as many times as you want, to see if it works.
- Submission: Only after you are satisfied with your solution, it must be submitted in Matlab Grader. Do this only once; in case of mistakes, you have a second chance to submit, but this should be a last resort.
- Verification: The code will be checked both by Grader automatically, and by the teacher, including a plagiarism check against the code of all your colleagues. Your attendance to the lab will only be registered if you have a working, original solution. Validated attendances for all the labs are necessary for eligibility to the exam. Moreover, at most two labs can be recovered at the end of the semester, which means accumulating three or more missing labs leads to ineligibility.
- Other remarks: Discussing ideas amongst the students is encouraged; however, directly sharing and borrowing pieces of code is forbidden, and any violation of this rule will lead to consequences as described in the discipline rules.

Assignment description

In this assignment we will perform function approximation with linear regression and polynomial approximators, see *Linear Regression* in the course material *Part 3 – Mathematical Background.*

A data set of input-output pairs is given, where the outputs are generated by an unknown mathematical function g. The function has one input variable and one output variable, and the output measurements are affected by noise. You will develop an approximator of this function, using a linear model with polynomial terms (basis functions). The parameters of the model will be found using the identification data set. A second data set is provided for validating the developed model. The two data sets are given in a MATLAB data file, containing one structure for each set. The training data set is named `id` and the validation data set `val`. Each of these structures contains a vector X of input samples, and the corresponding output samples in vector Y.

You will develop a function with the exact signature:

```
[index, theta5, mse, yhatstar] = polyreg
```

Each student is assigned an index number in the set 1-16, 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 10, you load file `lab4_10.mat`. All these datafiles are already accessible from your function code, they have been uploaded to the Grader problem (even though they are not visible explicitly).

Requirements follow.

- Load your dataset, and plot the identification and/or validation data to get an idea of the function shape.

- Create a polynomial approximator of degree $n-1$, where n is the number of parameters / basis functions. Here, n should be tunable. Note there is one extra parameter for the constant term, which is why the degree is just $n-1$. For example, when $n=4$, the polynomial has degree 3 and the approximator is:
$$\hat{g}(x) = \theta_1 + x\theta_2 + x^2\theta_3 + x^3\theta_4$$
The order of the polynomial terms is important, please keep them in increasing order of the power of x.

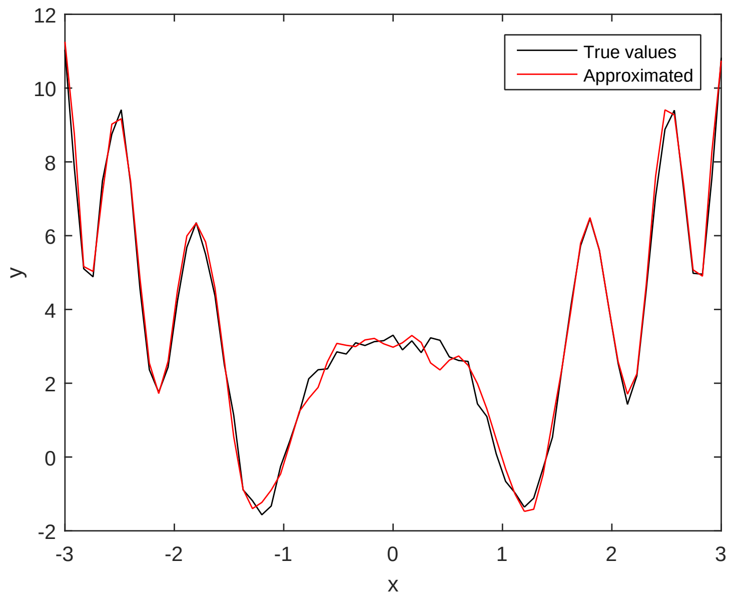
- For any value of n, create a system of linear equations for linear regression, using the identification data. Use the matrix representation explained in the lecture. Solve this system.

- Save the parameters obtained for $n = 5$ in `theta5`, and check that they are correct. This is a sanity check that your code works correctly.

- Validate the model on the different, validation data set: compute the approximated outputs and from those the MSE on the validation data. Make a plot of the approximated function on the validation data set, comparing to the actual outputs. Your plots will look similar to those exemplified in the figure below (except your data and fit quality may be different, of course).

- Tune n for good performance, by trying all the values between 2 and 20. Performance should be evaluated by the MSE on the different, validation data set to avoid overfitting. For each value of n, save the MSE in the vector `mse`, so that `mse(1)` is for $n = 2$, `mse(2)` is for $n = 3$, and so on.

- Produce a plot of the MSE versus n and find the point where the MSE is minimal. Save in `yhatstar` the approximate validation outputs for the optimal value of n.


True values
Approximated
y
x