Parameter estimation – Homework 1

Deadline: 19rd of March, 23:59:59

Solutions should be submitted via Moodle

General rules (for all homeworks)

- All exercises must be solved using Matlab. University license is provided, for installation instructions see https://info.itk.ppke.hu/it/u/licenc/matlab. You can also use Matlab online, however it can be slow and cumbersome.
- In order to effectively solve some problems, you might need different toolboxes in Matlab (like Statistics and Machine Learning Toolbox, Curve fitting toolbox, etc). These are all included in the university license, and can be added easily at any time using the Add-on browser. When you try to use a function that needs a specific toolbox, Matlab will throw an error containing the toolbox name.
- If someone is really afraid of Matlab, as a last resort we also accept Python3 scripts.
- All exercises must be solved individually. Solutions copied from other students of the course, former students, etc. will receive score 0.
- Try to write easily readable, understandable code (meaningful variable names, explanation in comment where necessary, etc). We do not have the resources to debug crashing codes, syntax errors, etc. Points are given for partial, incomplete solutions as well, but please clearly indicate (or comment out) which parts are not working properly.
- Naming convention for scripts: `<username>_hw<number>_task<number>.m` (eg. for task 2 I should submit a file named `csuba_hw1_task2.m`).
- Should you have any question, problem or remark, contact Balazs Csutak at the weekly practice / consultation session or in email (`csutak.balazs@itk.ppke.hu`).

Task 1 - ARMAX processes and data transformations

In this exercise we will simulate an ARMAX process (autoregressive moving average with exogenous noise) with a given input, followed by performing some transformations on its output.

Let us have the following process:

$$y(k) = 1.5y(k-1) - 0.7y(k-2) - 0.25u(k-1) + 1.64u(k-2) - 1.6\epsilon(k-1) + 0.75\epsilon(k-2),$$

where u is the input, and ϵ is normally distributed noise with expected value 0 and $\sigma = 1.2$.

Download the following file, containing the time series u(k):
https://users.itk.ppke.hu/ csuba/parameterestimation/paramest_hw1_task1.mat, then write a Matlab script that performs the following tasks:

1. Load the downloaded file into the workspace.
2. Calculate the output $y(k)$ of the system for $k = 1, 2, ..., N$, $N = 100$. (Hint: for ϵ use `randn`). You can assume $y(k) = u(k) = \epsilon(k) = 0$ for $k \leq 0$.
3. Create a figure showing the output of the system (the X axis should be the time). Turn on the grid on the plot.
4. As it typically occurs with real-world systems, the output is quite noisy. Apply a moving average transformation with a sliding window size $K = 5$ to y. (You can implement it using a for loop, or you can look for the built in Matlab function and use that).
5. Plot the transformed data on the same figure along y (using a different color). (Hint: use `hold on`).
6. Approximate the transformed data using a cubic spline. (Cubic splines are two times continuously differentiable functions / curves, commonly used for interpolation when we want to ensure a smooth result). Evaluate the spline with sampling time $T_s = 0.05$, ie. at times $t = 1, 1.05, 1.1, 1.15, ..., N$. Plot the approximation on the previously created figure (use a third color).
7. Calculate the first, second and third derivative of the spline using `diff`. Create a second figure, turn on the grid. Plot the spline and the calculated derivatives using different colors. (Note: `diff(·)` returns a vector which is 1 element shorter than ·).

Task 2 - probability recap

Download the file https://users.itk.ppke.hu/ csuba/parameterestimation/paramest_hw1_task2.mat and write a Matlab script that performs the following tasks:

1. Load the downloaded file into the workspace. It contains a vector named X, containing the coordinates of N=1000 two dimensional points.
2. Create a figure, plot the points.
3. Assuming the points are sampled from a two dimensional normal distribution, calculate the mean (expected) value and covariance matrix of the data. Plot the mean value on the figure along the points (choose a different color). (Hint: `mean`, `cov`).
4. Plot the multivariate probability density function having the previously calculated mean and covariance. This will be a 3D plot, use sampling distance 0.1 on the X and Y axis. (Hint: `meshgrid`, `mvnpdf`, `surf`/`mesh`.). Find a viewpoint (rotate the 3D plot) from where we can see the points and the surface properly (it might also help if you set the opacity of the surface using `FaceAlpha`).