Some Practical Exercises for System Identification

Fall 2018

Introduction

The objective of the computer exercise sessions is to implement different algorithms learned during the lectures and make the student familiar with MATLAB and its system identification toolbox. Twelve computer exercise sessions are planned. The groups of two students should prepare two reports (one report per six sessions) and submit them in the *moodle* of the course in due time. The reports will be graded with 15 points each. The written exam, which will be held on 18th of December, includes some questions about the computer exercises (40 points in total). Only the students are capable to answer theses questions that worked on the computer exercises and understood them.

1 CE-1 : Nonparametric Methods

In order to solve the following exercises, you will require Matlab2014 or later. Make sure your version is up to date. During the first session of the exercises you will create a Simulink model of a fourth order transfer function, which is used for all simulations in CE-1.

1.1 Step response

1. Create a new Simulink model. Create a fourth order `Transfer Function` block with the following parameters :

$$G(s) = \frac{1}{s^4 + 0.4s^3 + 4.3s^2 + 0.85s + 1}$$

2. Define the sample time $T_e = 0.5$ s (Is this a reasonable choice ?).
3. Simulate measurement noise by creating a `Random Number` block with a variance of 0.01 and a sample time of T_e and adding it to the output of the transfer function block.

4. Create an input `Saturation` block with an upper limit of 0.5 and a lower limit of -0.5.
5. Finish the block diagram by creating a `From Workspace` source and a `To Workspace` sink to exchange data with the workspace. Set the sample time of the From Workspace and To Workspace blocks to T_e.

6. Apply a step with a magnitude equal to the upper saturation limit at the input and plot the response of the system. To do this, create an M-file. First we need to generate the input signal. Create a struct `simin` with the following fields :
   - `simin.signals.values` : A column vector representing the input signal (i.e. the step)
   - `simin.time` : The time vector corresponding to the input signal

   The time vector should have a length of 100 s and the step should occur after 1 s. Note that the simulation time should be at least 100 s (in the simulation parameters of Simulink).
7. Use the command `sim` to run the Simulink model with the above input signal and plot the output of the system.
8. In the same way compute the impulse response of the system.

1.2 Auto Correlation of a PRBS signal

1. Download the file `prbs.m` from the Moodle page of the course. The function `prbs(n,p)` generates a PRBS of `p` periods with an `n`-bit shift register.
2. Write a function for Matlab `[R,h] = intcor(u,y)` that computes $R_{uy}(h)$ for the periodic signals.
3. Check your function by computing the autocorrelation of a PRBS signal.

1.3 Impulse response by deconvolution method

The objective is to compute the impulse response of the system in simulink using the numerical deconvolution method. Note that the simulation time should be less than 250 s and the random signal should not exceed the saturation values. For this purpose,

1. Generate a random signal (see `help rand`) with a sampling time of $T_e = 0.5$s as the input to the model.
2. Use `toeplitz` command to construct the input matrix.
3. Generate a time vector for simulink : `t=0:Te:(N-1)*Te`, where N is the length of your input signal.
4. Compute the impulse response using the measured output and the matrix of the input signal.
5. Compare your results with the true impulse response of the discrete-time system obtained by `zoh` option for transformation (see `tf, c2d, impulse`).

1.4 Impulse response by correlation approach

1. Generate an input sequence `Uprbs` using the `prbs` command (with $p = 4$ and $n = 7$) and apply it to your system. Set the sample time to $T_e = 0.5$ s.
2. Using your `intcor` function that you have already developed, compute the impulse response of the discrete-time system $g(k)$ using the correlation approach.
3. Compute the impulse response using `xcorr` function of Matlab.
4. Compare your results (using `xcorr` and `intcor`) with the true impulse response of the discrete-time system.

1.5 Frequency domain Identification (Periodic signal)

Aim : Use the Fourier analysis method to identify the frequency response of a model excited by a PRBS signal.

1. Choose a PRBS signal with a length around $N = 2000$ and a sampling period of $T_e = 0.5$s. Apply the generated PRBS to the Simulink model in 1.1.
2. Compute the Fourier transform of the input and output signal (use `fft`) for each period and use the average.
3. Compute a frequency vector associated to the computed values (Slide 30, Chap. 2).
4. Generate a frequency-domain model in Matlab using the `frd` command.
5. Plot the Bode diagram of the identified model and compare it with the true one.

1.6 Frequency domain Identification (Random signal)

Aim : Use the spectral analysis method to identify the frequency response of a model excited by a random signal.

1. Apply a random signal of length $N = 2000$ to the system. Discuss the characteristics of the random signal (Gaussian or uniform, binary or multi-level) to obtain the highest energy of the excitation signal.
2. Compute the frequency-response of the model using the spectral analysis method.
3. Use a Hann or Hamming window to improve the results.
4. Cut the data to m groups and try to improve the results by averaging.
5. Plot the Bode diagrams of different methods and compare them.