Some Practical Exercises for System Identification

Spring 2024

Introduction

The objective of the computer exercise sessions is to implement the various algorithms that students have learned during the lectures and to familiarize them with MATLAB and its system identification toolbox. There are twelve computer exercise sessions planned, and each group of two students should prepare two reports (one report for every six sessions) and submit them on the course's Moodle platform by the specified due date. The reports will be graded out of 15 points each.

1 CE-1 : Nonparametric Methods

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 $T_s = 0.5$ 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_s 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_s.

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 $T_s = 0.5$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:Ts:(N-1)*Ts`, where N is the length of your input signal.
4. Compute the finite impulse response using the measured output and the matrix of the input signal. Make an assumption on the length of the impulse response.
5. Compute the impulse response by regularisation (trade-off between bias and variance).
6. 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.
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 $T_s = 0.5$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 33, 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.

2 CE-2 : Parametric Identification Methods

The objective of this exercise is to practice the parametric identification methods. The real data of two different systems are considered. In the first part, input-output data is collected from a real DC servomotor using the provided user interface. Then, based on the collected data, a simple model is obtained using different identification methods. In the second part the complete identification (order estimation, parametric identification and model validation) of a mechatronic system is considered.

2.1 Identification of a DC servomotor

The Quanser QUBE-Servo, pictured in Fig. 1, is a compact rotary servo system that can be used to perform a variety of classic servo control and inverted pendulum based experiments.

Figure 1. DC servomotor

Since the open-loop system contains an integrator, a proportional controller is used to stabilize the system. The objective is to identify the whole closed-loop system. The excitation signal u is a PRBS and the axis position y is measured with a sampling frequency of 1 kHz. The input and output must be obtained from the real system using the provided user interface (installed on the PCs in the lab) as follows :

1. Open the file named `Main.vi` located inside the `SYSID` folder on your desktop. The GUI in Fig. 2 will appear allowing one to apply a PRBS input to the system and record the output response.

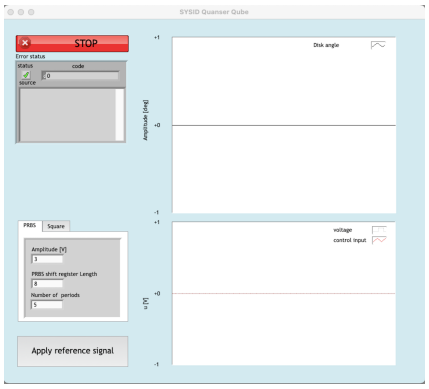

Figure 2. GUI for data collection

2. On the GUI, set the PRBS input parameters as
   — Amplitude = 1
   — PRBS shift register length = 8
   — Number of periods = 5
   and apply the input by pressing the "**Apply reference signal**" button at the bottom. The input and output signals will be displayed on the GUI as in Fig. 3.

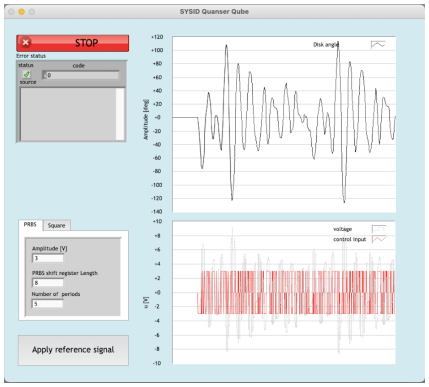

Figure 3. GUI during the application of a PRBS input

3. Once the application of PRBS input is completed, such that no input is being applied to the motor anymore (can be observed from the GUI), press the red "**STOP**" button on top of the window. This saves the input-output data obtained during the application of the PRBS input into a file named `logs.bin`, located at the same directory with `Main.vi`.

4. Retrieve the data from the `logs.bin` file in MATLAB, using the `GetExperimentData.m` file also provided in the `SYSID` folder, by the command `[y,u,Te] = GetExperimentData('logs.bin')`.

Then, perform system identification with different methods as requested in the following sections.

Remark : Once the data is collected you can send the `logs.bin` and `GetExperimentData.m` files to yourselves and work on your own PCs as well.

2.1.1 FIR model identification

Assume an FIR model for the data with $m = 200$ parameters ($d = 1$) :

$$\hat{y}(k, \theta) = b_1 u(k-1) + \cdots + b_m u(k-m),\ \theta^\top := [b_1\ \cdots\ b_m]$$

1. Estimate the vector of parameters θ as $\hat{\theta}$ using the least squares algorithm. Note that the matrix Φ is a Toeplitz matrix.
2. Compute the predicted output of the identified model, $\hat{y}(k, \hat{\theta})$. Plot the measured output $y(k)$ and the predicted output in the same figure and compute the loss function $J(\hat{\theta})$.
3. Assuming that the noise is white, estimate the noise variance and compute the covariance of the estimated parameters. Plot the finite impulse response of the system (the vector $\hat{\theta}$) together with $\pm 2\sigma$ confidence interval (use `errorbar`).

2.1.2 ARX model identification

Assume a second order ARX model for the data with the following predictor :

$$\hat{y}(k, \theta) = -a_1 y(k-1) - a_2 y(k-2) + b_1 u(k-1) + b_2 u(k-2),\ \theta^\top = [a_1\ ,\ a_2\ ,\ b_1\ ,\ b_2]$$

1. Estimate the vector of the parameters θ as $\hat{\theta}$ using the least squares algorithm.
2. Compute the predicted output of the identified model, $\hat{y}(k, \hat{\theta})$. Plot the measured output $y(k)$ and the predicted output in the same figure and compute the loss function $J(\hat{\theta})$.
3. Compute the output of the identified model, $y_m(k)$, for the input signal, $u(k)$, using `lsim`. Plot the measured output $y(k)$ and $y_m(k)$ in the same figure and compute the sum of squares of the error.
4. Try to improve the results using the Instrumental Variable method (note that $y_m(k)$ is a noiseless estimate of $y(k)$). Compare the result with that of ARX model.

2.1.3 State-space model identification

The objective of this part is to identify a sate-space model for the Quanser Qube-servo system using the subspace method based on the extended observability matrix.

1. Construct the matrix $\boldsymbol{Y}$ and $\boldsymbol{U}$ based on the measured data (see page 83 of the course-notes) and compute $Q = \boldsymbol{Y}\boldsymbol{U}^\perp$.
2. Compute the singular values of Q and conclude the number of states n (use `[UU,S,V] = svd(Q)`). Compute the extended observability matrix O_r (the first n columns of `UU`).
3. Compute the matrix A and C from the extended observability matrix.
4. Assume that $D = 0$ and estimate B using the least squares algorithm.
5. Compute the output of the identified model, $y_m(k)$ for the input signal $u(k)$ using `lsim`. Plot the measured output $y(k)$ and $y_m(k)$ in the same figure and compute the the sum of squares of the error.