Some Practical Exercises for System Identification

Spring 2025

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 18 points each.

1 CE-1 : Nonparametric Methods

During the first session of the exercises you will create a Simulink model of a third order transfer function, which is used for all simulations in CE-1.

1.1 Step response

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

$$G(s) = \frac{1.2}{s^3 + 2s^2 + 1.35s + 1.2}$$

2. Define the sample time $T_s = 0.25$ 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 0.7 and a lower limit of -0.7.
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 50 s and the step should occur after 1 s. Note that the simulation time should be at least 50 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 unit step response of the system $G(s)$.
8. In the same way compute the impulse response of the system $G(s)$.

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 100 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 $T_s = 0.25$ 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 $p = 2$ and $n = 8$) 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(.,.,'unbiased')` 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 $T_s = 0.25$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, we will use a proportional controller to stabilize the system. Thus, the objective is to identify the whole closed-loop system. The excitation signal u is a random binary signal and the axis position y is measured with a sampling period of 0.01s. The input and output must be obtained from the real system using the provided user interface as follows :

1. To connect to one of the experimental setups please go to https://courseware.epfl.ch/courses/course-v1:EPFL+controlsys+2017_T1/aboutand enroll by clicking on the "`Enroll Now`" button and following the instructions. Once you are enrolled follow : *View Corse* $\rightarrow$ *Module* 8 $\rightarrow$ *Implementation* and click on the QUBE access link.

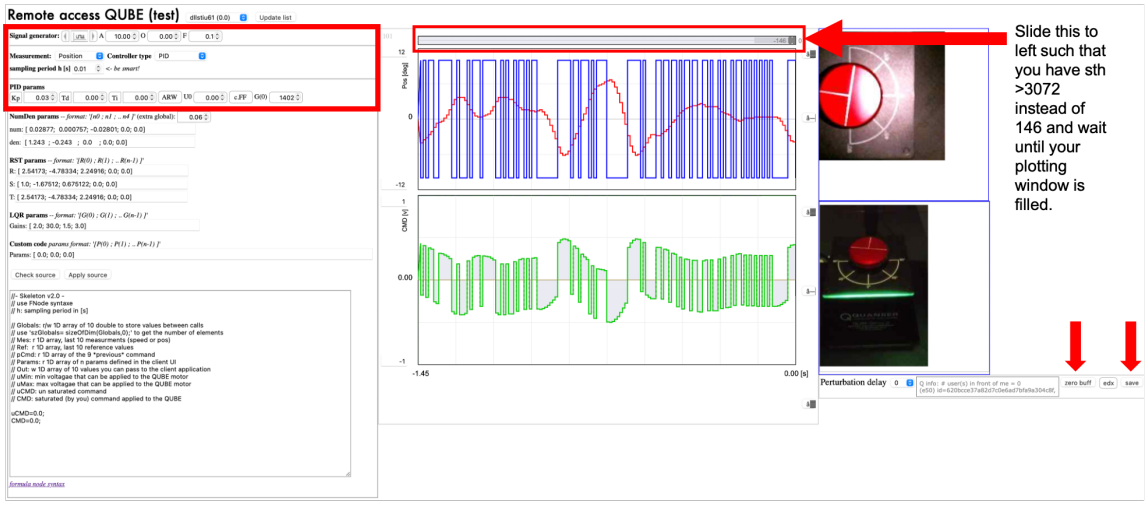

Figure 2. GUI for data collection with remote access.

2. On the GUI set :
   — **Signal Generator** → random binary signal with amplitude $A = 10$. (this is the input signal that you will use for data collection)
   — **Measurement** → position
   — **Controller type** → PID (this is your initial stabilizing controller)
   — **Sampling period h [s]** → 0.01
   — **PID params** → $K_p = 0.03, T_d = 0, T_i = 0$
   The GUI with the correct settings will be as presented in Figure 2 where also the input output signals and a live video of your setup are being displayed.
3. Once the the settings are done click on the "`zero buff`" button to clear the previous data. Wait until you have at least 3072 new samples and then click on the "`save`" button to download the data into your PC. (You can check how many samples you have after the zero buff by using the slider on top of the plots, see Figure 2).
4. Rename the downloaded data file. Then, by running the `GetExperimentData.m` file provided in the Moodle page of our course load the data to your Matlab workspace (You need to provide the path of your data file to `GetExperimentData.m` as input). This should provide you the input output data of 3 periods with a period of 1024 samples.

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

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 $J(\hat{\theta})$ (loss function is equal to the sum of squares of the prediction error).
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 two norm 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 two norm of the error.

2.2 Parametric Identification of an Electromechanical System

The objective of this part is to identify and validate a parametric black-box model of a DC motor with a flexible element attached on top, resulting in large resonant modes at high frequencies. Different weights can be attached to the flexible element at different positions, resulting in different loadings (see Fig. 3). In the first part the order of the system is identified using the data and in the second part different parametric models are identified and validated.

A model between the DC motor input voltage u and the axis position y should be identified. For data collection the system is excited with a PRBS signal of amplitude 5 and shift register 13, the axis position y is measured with a sampling frequency of 500 Hz. As in Part 2.1, the open loop system from input to the axis position contains an integrator. This time instead of using an initial controller to stabilize the system, we will first identify a model from the input to the angular velocity of the axis and add an integrator to our model afterwards such that we end up with a model from the input to the axis position.

Download the file `data_position1.mat` that contains the experimental data of the flexible link system containing the input signal u and the sampled output signal y. Compute the derivative of the output measurements to obtain angular velocity data of the axis by the command :

```
y_derivative = lsim(1 - tf('z', Ts)^-1, y);
```

Use the command `iddata` to generate a data object that can be used for the System Identification Toolbox and remove the mean value of the data using the command `detrend` :

```
Z = detrend(iddata(y_derivative, u, Ts, "Period", 8191));
```

Now you can perform system identification on `Z` and you will add back the integrator at the very last step of model validation.

Figure 3 : Quanser Servo-Qube with weights attached on top

Remark : Matlab uses the following notation :

$$A(q^{-1})y(t) = \frac{B(q^{-1})}{F(q^{-1})}u(t - n_k) + \frac{C(q^{-1})}{D(q^{-1})}e(t)$$

where :

$$A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a} \quad B(q^{-1}) = b_1 + b_2 q^{-1} + \cdots + b_{n_b} q^{-n_b+1}$$
$$C(q^{-1}) = 1 + c_1 q^{-1} + \cdots + c_{n_c} q^{-n_c} \quad D(q^{-1}) = 1 + d_1 q^{-1} + \cdots + d_{n_d} q^{-n_d}$$
$$F(q^{-1}) = 1 + f_1 q^{-1} + \cdots + f_{n_f} q^{-n_f}$$

For ARX structure we have $n_c = n_d = n_f = 0$, for ARMAX $n_d = n_f = 0$, for OE : $n_a = n_c = n_d = 0$ and for BJ : $n_a = 0$.

2.2.1 Order estimation

1. Using the `arx` command, identify 15 models of orders from 1 to 15 ($n_k = 1, n_a = n_b = \delta, \delta = 1, \ldots, 15$) and plot the "loss function" of the models versus the model order (loss function is available for the model `M` in the following variable : `M.EstimationInfo.LossFcn`). Estimate the order of the system from this curve.
2. Validate the chosen order by checking the zero/pole cancellation approach using several models identified by the ARMAX structure ($n_k = 1, n_a = n_b = n_c = \delta, n = \delta_{\min}, \ldots, \delta_{\max}$). Use `h=iopzplot(M)` to plot the zeros and poles of M and `showConfidence(h,2)` to plot their confidence intervals ($\pm 2\sigma$).
3. Estimate the delay n_k by inspecting the coefficients of $B(q^{-1})$ and their standard deviations (use `M.b` and `M.db`. Compute the number of parameters in the numerator ($1 \leq n_b \leq \delta - n_k + 1$).
4. Compare your results with those proposed by `struc, arxstruc, selstruc`.

2.2.2 Parametric identification

1. Divide the data into two parts and use one part for identification and the other part for validation.
2. Using the estimated values of n_a, n_b and n_k, identify different parametric models by the following methods : `arx, iv4, armax, oe, bj, n4sid`. For the structures with noise model take $n_c = n_d = n_a$. For the state-space model choose number of states equal to the global order δ.

2.2.3 Model validation

1. Compare the output of the identified models with the derivative of the measured output using the command `compare`. What is the best model ?
2. Compare the frequency response of the models with a nonparametric frequency-domain identified model from `u` to `y_derivative`. What is the best model ?
3. Validate the identified models by the whiteness test of the residuals as well as the cross-correlation of the residuals and past inputs using the command `resid`. Which models are validated ?
4. Add an integrator to all of the parametric models that you have obtained such that :

```
G = G_derivative / (1 - z^-1);
```

Generate a new validation dataset using the second half of the position data `y`. Compare the output of the identified models with the measured output using the command `compare`. Compare the frequency response of the models with a nonparametric frequency-domain identified model from `u` to `y`. What is the best model?