Data-driven learning of dynamical systems

Hugues Garnier

February 2022

Contents

Tutorials 1

Basic process model identification methods

Downloading of the data needed for the tutorial

- Download the zipped file **Tutorial1_SYSID.zip** from the course website and save it in your Matlab working directory.
- Start Matlab.
- By clicking on the browse for folder icon , **change the current folder of Matlab so that it becomes your Tutorial1_SYSID folder** that contains the files needed for this lab.
- When necessary, it is recommended to use the Matlab Live Editor. In the Live Editor, you can create live scripts that show output together with the code that produced it.

Exercise 1.1 - Identification of a heat exchanger model from step response data
The experimental step response of a heat exchanger is stored in the file `heatexchanger_data.mat` and is plotted in Figure 1.1.

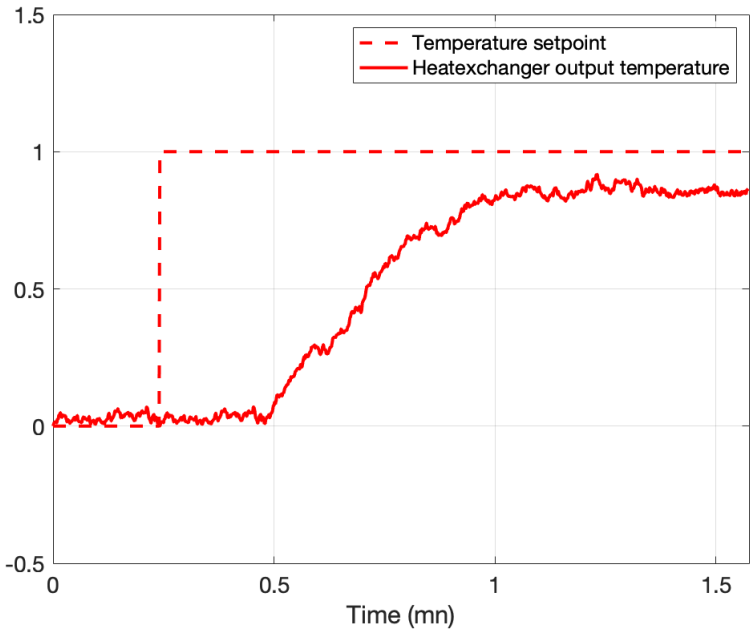

Figure 1.1: Step response of the heat exchanger (experimental data)

From the step response shown in Figure 1.1:

1. Propose a transfer function model form $G(s)$ for the heat exchanger. Explain your choice.
2. Determine the different parameters of your chosen transfer function model $G(s)$.
3. Use the `test_your_model.m` file to compute the step response of your model and compare it to the measured one.
4. Use the CONTSID toolbox App to estimate the parameters of your chosen transfer function model.

Exercise 1.2 - Identification of a mechanical suspension system from step response test

Consider the position response $y(t)$ of a mechanical suspension system to a unit step input ($u(t) = \Gamma(t)$) plotted in Figure 1.2. The input/output data are stored in the file `suspension_data.mat`.

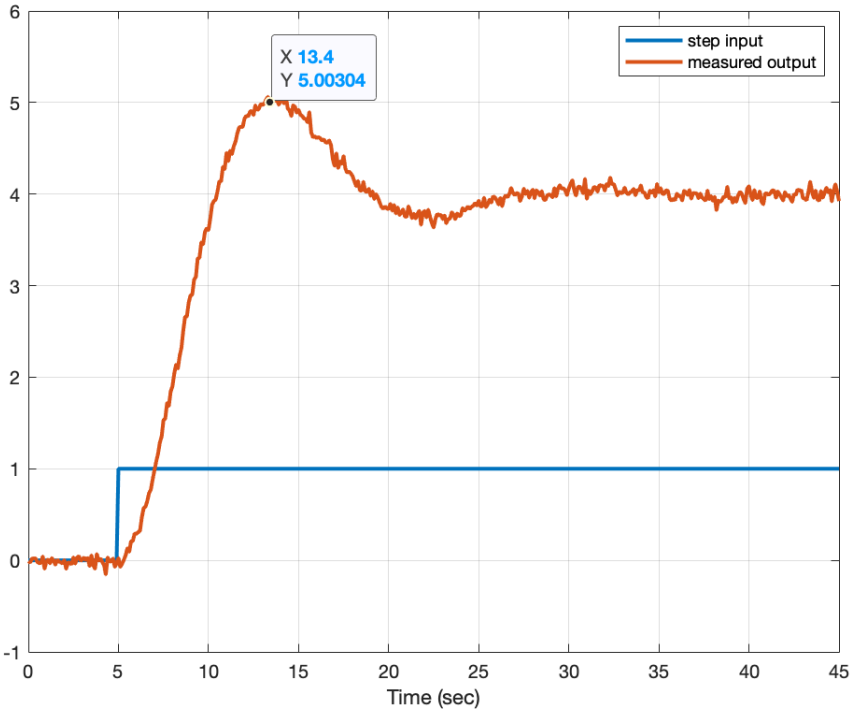

Figure 1.2: Response of a mechanical system to a unit step input

1. Propose a transfer function model form $G(s)$ for the system. Explain your choice.
2. Determine the numerical values of the different parameters of your chosen $G(s)$.
3. Use the `test_your_model.m` file to compute the step response of your model and compare it to the measured one.
4. Use the CONTSID toolbox App to estimate the parameters of your chosen transfer function model.

Model identification from step responses

Identification of a first-order model

$$G(s) = \frac{K}{1 + Ts}$$

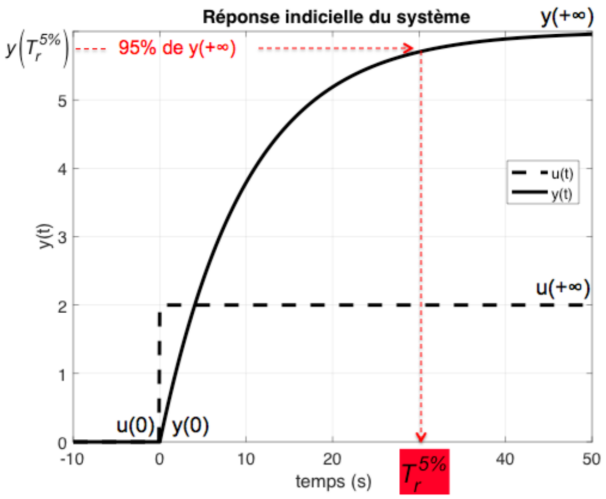

Figure 1.3: Step response of a first-order system

From the step response displayed in Figure 1.3, it is necessary to determine the steady-state gain K and the time constant T. The procedure is as follows:

1. Find the final and initial values of the response and of the step. Deduce K from:

$$K = \frac{y(+\infty) - y(0)}{u(+\infty) - u(0)}$$

2. Find $y(T_r^{5\%})$, deduce from it $T_r^{5\%}$ then T:

$$T = \frac{T_r^{5\%}}{3}$$

Identification of a second-order underdamped model

$$G(s) = \frac{K}{\frac{s^2}{\omega_0^2} + 2\frac{z}{\omega_0}s + 1} = \frac{K\omega_0^2}{s^2 + 2z\omega_0 s + \omega_0^2}$$

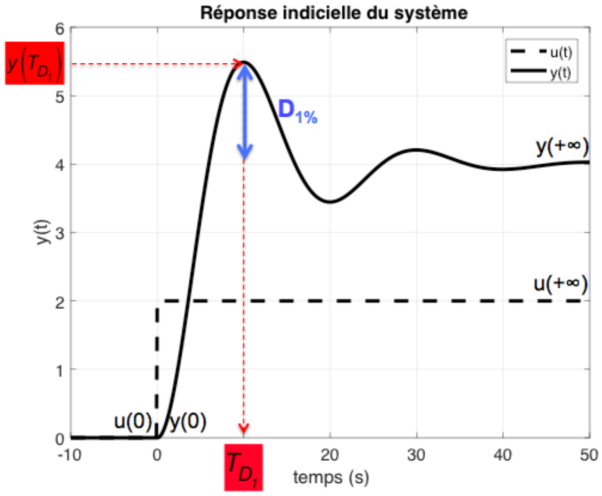

Figure 1.4: Step response of a second-order system

From the step response displayed in Figure 1.4, it is necessary to determine the steady-state gain K, the damping ratio z and the undamped natural frequency ω_0. The procedure is as follows:

1. Find the final and initial values of the response and of the step. Deduce K from:
$$K = \frac{y(+\infty) - y(0)}{u(+\infty) - u(0)}$$

2. Find the final and initial values of the response and that of the first overshoot $y(t_{D_1})$. Deduce from it D_1, then z:
$$D_1 = \frac{y(T_{D_1}) - y(+\infty)}{y(+\infty) - y(0)}$$
$$z = \sqrt{\frac{(\ln(D_1))^2}{(\ln(D_1))^2 + \pi^2}}$$

3. Find the time-instant of the first overshoot T_{D_1}. Deduce from it ω_0:
$$\omega_0 = \frac{\pi}{T_{D_1}\sqrt{1 - z^2}}$$

Tutorials 2

Simple least squares for estimating the parameters of static and dynamic linear models

Downloading of the data needed for the tutorial

- Download the zipped file **Tutorial2_SYSID.zip** from the course website and save it in your Matlab working directory.

- Start Matlab.

- By clicking on the browse for folder icon  , **change the current folder of Matlab so that it becomes your Tutorial2_SYSID folder** that contains the files needed for this lab.

- It is recommended to use the Matlab Live Editor. In the Live Editor, you can create live scripts that show output together with the code that produced it.

Exercise 2.1 - Use of least squares to model and forecast the winning men's 100 m time at the Summer Olympics in Paris in 2024

The file `winning_time.mat` includes the winning men's 100 m times and the olympic years since 1896.

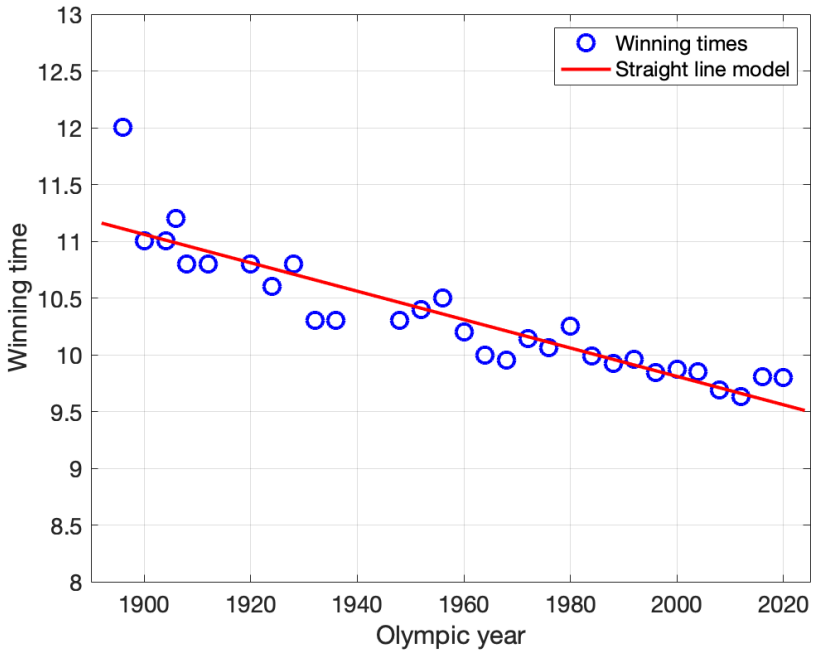

Figure 2.1: Winning men's 100 m times at the Summer Olympics since 1896 and the estimated straight line model. Note that the two world wars interrupted the games in 1914, 1940 and 1944.

The goal is to model the winning time decrease over the years and predict the winning men's 100 m time at the Summer Olympic Games in Paris in 2024.

1. Load the dataset in Matlab.
2. Plot the time series and observe the winning time decrease over time.
3. We choose to postulate a polynomial model to capture the winning time decrease as a basic straight line function

$$winning\_time(years) = a \times years + b$$

4. Determine by using simple least squares the parameters of the straight line model.
5. Compute the residuals along with the RMSE performance indice..
6. Use your model to forecast the winning men's 100 m time at the Summer Olympic Games in Paris in 2024.

Bonus ! For those who want to review simple linear regression (least-squares, normal equation, gradient descent, etc) discussed during the last two lectures and tutorials, watch the 10 mn video and read the excellent explanation written in **French** by Guillaume SAINT-CIRGUE, a lead data scientist available at: `machinelearnia.com/regression-lineaire-simple/`

Exercise 2.2 - Use of least squares to estimate the continuous-time and discrete-time model of a dynamic system from step response data
The file `step_reponse_data.mat` includes 4 samples of the input and output of a first-order dynamic system.

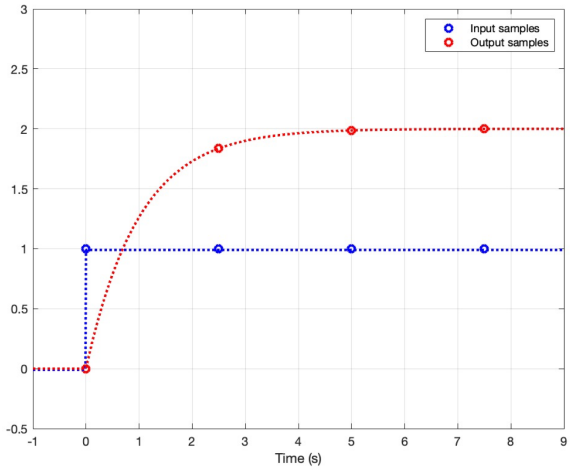

Figure 2.2: Step response of a first-order dynamic system.

The goal is to fit a continuous-time and discrete-time transfer function model from the step response data.

1. Load the dataset in Matlab.
2. Plot the step response and observe the input and output samples over time.
3. We choose to postulate a continuous-time first-order transfer function model to capture the step response of the dynamic system.

$$G(s) = \frac{b}{s+a}$$

4. Determine by using simple least squares the parameters of the continuous-time transfer function model.
5. Compare the simulated model output with the deterministic step response.
6. We now choose to postulate a discrete-time first-order transfer function model to capture the step response of the dynamic system.

$$G(z) = \frac{b_1 z^{-1}}{1 + a_1 z^{-1}}$$

7. Determine by using simple least squares the parameters of the discrete-time transfer function model.
8. Compare the simulated model output with the deterministic step response.
9. Repeat the model fit by using the noisy output and time-derivative signals.
10. Conclude about the use of simple least squares in presence of noisy output measurements.

Tutorials 3

Iterative model learning workflow and trolley crane pilot identification

Downloading of the data needed for the tutorial

- Download the zipped file **Tutorial3_SYSID.zip** from the course website and save it in your Matlab working directory.
- Start Matlab.
- By clicking on the browse for folder icon  , **change the current folder of Matlab so that it becomes your Tutorial3_SYSID folder** that contains the files needed for this lab.
- It is recommended to use the Matlab Live Editor. In the Live Editor, you can create live scripts that show output together with the code that produced it.

Exercise 3.1 - The iterative model learning workflow by examples
The iterative model learning workflow is represented in Figure 3.1.

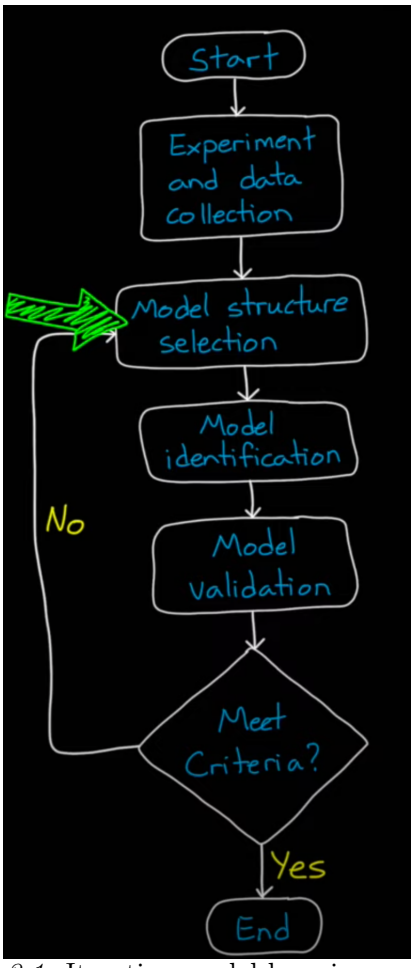

Figure 3.1: Iterative model learning workflow.

1. Watch the video (18 mn) by Brian Douglas entitled **Linear System Identification - Part 2** available at:
   `www.youtube.com/watch?v=qC_C04SEV1E`

2. Run the file `linear_sys_example.mlx` in the live editor and reproduce the results obtained from the simulation example presented in the video.

3. Pay attention in particular to the selection of the model order (number of poles and zeros for the transfer function model) of the iterative model identification workflow.

4. Run the file `heat_transfer_sys_id.mlx` in the live editor and reproduce the results obtained from the heat exchanger presented in the video.

Exercise 3.2 - Identification of a one input - two outputs transfer function model of the trolley crane pilot

1. Run the file `import_crane_data.mlx` in Matlab that loads the data from a PRBS test applied to the trolley crane pilot.

2. Use the estimation and validation dataset to estimate a transfer function model of the one input - two outputs trolley crane pilot, as shown in Figure 3.2.

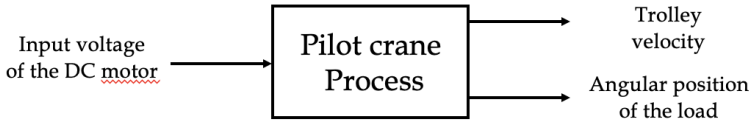

Figure 3.2: Main input and outputs of the trolley crane pilot.

This exercise is evaluated from a written report which should be readable independently. It can be written in French or in English but do not mix both languages.
Your report should be a scientific document. As such, it should be self-contained: that is, someone who has never seen the tutorial instructions should be able to understand the problem that you are solving and how you are solving it. This means that all of your choices should be thoroughly explained and motivated. Your final linear model must capture the essential behavior of the dynamic system. However, the most important thing is not that you succeeded in producing the absolute best models, but that you can explain the shortcomings and merits of your model. These shortcomings and merits can come from different aspects and can be of a different nature, such as total fit, fit from input to output, noise model or not, physical interpretation. Therefore, be as clear as possible in your explanations. Also include relevant plots of the phenomena that characterize your choices (and explain what you can see in them). Which final model structure you recommend and why?
Below is a list of the steps that should be included in your report:

1. Short description of the process (input, measured outputs, etc)

2. Description of the experiment design (type of input, possible constraints, etc)

3. Description of the raw dataset

4. Data pre-processing (informative data selection, discard of the section with nonlinearity effects, etc)

5. Model structure selection (method used and results)

6. Parameter estimation (methods used and results)

7. Model validation (see the slides presenting the practical aspects of data-driven modelling available on the course website).

8. Conclusion. Discuss the strengths/weaknesses of the identified model(s) and specify which model do you recommend and why.

9. You can in an appendix, but this is not mandatory, suggest a control strategy (give at least the closed-loop block diagram) to move the trolley from a given to a desired position in the quickest way while keeping the pendulum angle the smallest possible.

The length of your report is limited to 6 pages (7 pages if you include the appendix). It must be submitted electronically in a .pdf format to facilitate its handling without your name appearing on the front page so that an anonymous peer reviewing process can be implemented. The deadline for the submission of your lab report to the instructor is **Friday, February 25 at 1:00 pm**.

Peer review report Peer review is an exercise in which students learn from other course members and help each other to improve their report. A few hours after you have submitted your report, you will be assigned the report of another group of students in an anonymous and randomized way. Read the report carefully and write a review according to the points below. The deadline for the submission of your review report to the instructor is **Tuesday, March 1 at 1:00 pm**.

The review report should be 1-2 pages and submitted electronically to the instructor in a .pdf format without your name appearing on the front page.

In your peer review report, you should answer and discuss the following questions:

1. Is the description of data collected clear? Are the methods used for data collection described in sufficient detail so that the experiments can be performed by someone else?
2. Is there a clear description of the solutions to all identification workflow steps ?
3. Do the conclusions agree with the data, the experiments and the results? Do you agree with the conclusions? What would you like to change in the conclusions based on the data and the results?
4. What is the greatest strength of the report? Discuss the content, not the format.
5. What suggestions do you have for improving the quality of the report? Discuss clarity, readability and technical accuracy.

The questions should be answered as a discussion, presenting arguments for your positions and proposing alternative methods and improvements. Specific tips:

- Use constructive criticism.
- Use a positive tone and accommodating language.
- Describe the things that can be improved clearly.
- Avoid comments that may be perceived as insulting or inappropriate.

Most people ignore feedback that they find negative, vague, or confusing. Therefore, try to give constructive comments, it will make them more useful to those who wrote the report.

The final version of your report taken into account the comments of the peer review comments should be sent to the instructor by **Friday, March 4 at 2:00pm**.

Summary of the main deadlines

- First version of your report: February 28 at 1:00 pm
- Peer review report: Tuesday, March 1 at 1:00 pm
- Final version of your report: Friday, March 4 at 2:00pm

Tutorials 4

Identification of linear transfer function models with the CONTSID toolbox

Some general comments

- For this tutorial session, you will mainly use the CONTinuous-Time System IDentification (CONTSID) toolbox which is a collection of m-files written in Matlab.
- The toolbox can be run in command-line mode or in a graphical user interface (GUI) mode or in any combination of these. It is however recommended to run/solve the exercices with the command-line mode.
- The exercices are intended for the CONTSID toolbox 7.4 to be run with Matlab version R2021 equipped with the System Identification and the Control toolboxes.

Downloading of the data needed for the tutorial

- Download the zipped file **Tutorial4_SYSID.zip** from the course website and save it in your Matlab working directory.
- Start Matlab.
- Change the current folder of Matlab so that it becomes your Tutorial4_SYSID folder that contains the files needed for this lab.

Exercise 4.1 - Tutorial introduction to the CONTSID toolbox
This first exercise is meant as an introduction to computer-based system identification with the CONTSID toolbox. You will run an identification demonstration program available in the toolbox. The chosen demonstration includes several steps that will be discussed thoroughly later on in the other exercices, so do not expect to understand all steps taken in the demonstration.

Start Matlab and run the CONTSID main demonstration program through the command:

```
>> contsid_demo
```

Select *Tutorials* in the menu window.
In the new menu window, select *Getting started* and follow the demo displayed in the command window.
When the demo is finished, select *Quit* to come back to the main menu window.

Run one or more other demos to get a feel for the different options available in the CONTSID toolbox.
You can also get familiar with the basic functions of the CONTSID toolbox in its GUI mode.

Run the CONTSID GUI through the command:

```
>> contsid_gui
```

By clicking on the Load data button of the left Data Panel, import the dataset from the trolley crane pilot and use the CONTSID GUI to identify a transfer function model for both outputs.
Exercise 4.2 - Parameter estimation of a simulated transfer function model The basic problem in system identification is to find the dynamics from input/output measurements. Of course, in practice the true system is unknown (otherwise there would be no need for system identification) and only the measurements are available. We are here using a simulated system for:

(i) generating data

(ii) evaluating the performance of various parameter estimation methods studied during the lectures.

Open the file `linear_sys_contsid_example.mlx` in the live editor and execute the different sections to get the results discussed in the paragraphs below.

Data-generating system

We will consider estimation from data given by the following system:

$$\mathcal{S}\begin{cases} x(t) = \dfrac{B_o(s)}{F_o(s)}u(t) \\ y(t_k) = x(t_k) + e(t_k) \end{cases} \tag{4.1}$$

where s represents here the differentiation operator $s = \frac{d}{dt}$ which is the notation used in the Matlab System Identification and CONTSID toolboxes.
u is the system input, x is the noise-free output and y the noisy output.

The disturbance $e(t_k)$ is a discrete-time white Gaussian noise, independent of the input u, and of zero mean and variance σ_e^2.
The polynomials $B_o(s)$ and $F_o(s)$ are defined by:

$$\begin{cases} B_o(s) = 2 \\ F_o(s) = s^2 + 4s + 3 \end{cases} \tag{4.2}$$

1. From (4.1) recall the model structure of the true system (CARX, COE or CBJ)
2. Deduce from (2) the main features of the true system: steady-state gain, time-constants and related cut-off frequencies (in rad/s)

For a PRBS input and measurement noise with $\sigma_e = 0.2$, we compute now the noise-free and noisy system responses. To define the true system and generate the data, type (or copy and paste) the following commands in a mlx script:

```
s = tf('s');
Go = 2/(s^2 + 4*s + 3)% The true transfer function model of the system
Ts=0.01; %  The sampling period
N=1500; % The number of data
t=(0:N-1)'*Ts; % The time vector
u=prbs(4,100); % The PRBS input
e=0.2*randn(N,1); %  The Gaussian noise of standard deviation 0.2
x=lsim(Go,u,t); % The noise-free output
y=x+e; % The noisy output
```

For later use with the CONTSID toolbox estimation routines (LSSVF, IVSVF, TFSRIVC), it is convenient to combine the input and output data into `iddata` structures

```
data0=iddata(x,u,Ts); % The noise-free data object
data=iddata(y,u,Ts); % The noisy output data object
figure(1)
idplot(data0)
figure(2)
idplot(data)
```

We assume that the "correct" model order is known and let us try to estimate the parameters from the two sets of data by the different direct continuous-time methods studied during the lectures.
Note that the PRBS input `u` is a deterministic signal. The noise signal `e` on the other hand is stochastic, so each time you run the commands above, you will obtain a different realization of the noise signal and hence also of the output. Your model estimate should therefore slightly differ for each noise realization and also from your neighbor.

Model learning by the basic LSSVF - Your own implementation

We first assume a CARX model structure for the system:

$$\mathcal{M}_{\text{CARX}} : y(t_k) = \frac{B(s)}{A(s)}u(t_k) + \frac{1}{A(s)}e(t_k) \tag{4.3}$$

or

$$\mathcal{M}_{\text{CARX}} : A(s)y(t_k) = B(s)u(t_k) + e(t_k) \tag{4.4}$$

The model polynomials $B(s)$ and $A(s)$ are given by:

$$\begin{cases} B(s) = b_0 \\ A(s) = s^2 + a_1 s + a_2 \end{cases} \tag{4.5}$$

1. Assuming N samples of input/output data, formulate the linear regression problem in matrix form. Give the dimension of each matrix.

2. Recall the LSSVF solution (see lecture slides).

3. Set the cut-off frequency of the SVF filter to $\lambda = 3$ rad/s and implement your solution on Matlab for the noise-free data and verify that the parameters are very close to the true parameters.

4. Compute then your solution with Matlab for the noisy output data and verify that the parameters are not equal to the parameters of the true system.

5. Run several times your program to get a feel for the overall performance of the simple LSSVF estimates.

Model learning by the basic LSSVF - CONTSID toolbox implementation

The CONTSID `lssvf` routine estimates the parameters of a CARX model by the basic LSSVF. The command to be used is:

```
Glssvf=lssvf(data,[na nb nk],lambda);
```

where `na` et `nb` represent the number of parameters to be estimated for the $A(s)$ and $B(s)$ polynomials respectively and `nk` denotes the number of samples for the time-delay.
`lambda` represents the cut-off frequency of the SVF filter (in rad/s).

1. Type the following commands:

   ```
   Glssvf0=lssvf(data0,[2 1 0],3);
   present(Glssvf0)
   Glssvf=lssvf(data,[2 1 0],3);
   present(Glssvf)
   ```

   The estimated parameters can be obtained from the model objet from the following command:

   ```
   Glssvf.param'
   ```

2. For a given noise realization, compare your self-made and CONTSID LSSVF estimates. They should be identical or very close.

Model learning by the two-step IVSVF - CONTSID toolbox implementation

We now consider the identification of a **CARX** model by the basic two-step IV-based SVF method.
The CONTSID `issvf` routine estimates the parameters of a CARX model by IVSVF. The command to be used is:

```
Givsvf=ivsvf(data,[na nb nk],lambda);
```

1. Type the following commands:

   ```
   Givsvf=ivsvf(data,[2 1 0],3);
   present(Givsvf)
   ```

2. For a given noise realization, compare the CONTSID LSSVF and IVSVF estimates. The latter should be much closer to the true parameters

3. Both LSSVF and IVSVF learning algorithms require the user to specify a hyperparameter which is the cut-off frequency λ of the SVF filter. It is recommended to choose it a bit larger than the system bandwidth (which is about 3 rad/s here but is usually not known a priori). To investigate the sensitivity of the IVSVF method against its user parameter, compute the IVSVF estimates when the SVF cut-off frequency is set to `lambda=0.3` and `lambda=30`. Run several times your program to get a feel for the effect of the SVF filter cut-off frequency on the IVSVF estimates.

Model learning by the iterative TFSRIVC - CONTSID toolbox implementation

We now consider the identification of a COE model by the Simple Refined IV method for Continuous-time transfer fonction model (TFSRIVC):

$$\mathcal{M}_{\text{COE}} : y(t_k) = \frac{B(s)}{F(s)} u(t_k) + e(t_k) \tag{4.6}$$

The CONTSID `tfsrivc` routine can be used as follows:

```
Gtfsrivc=tfsrivc(data,np,nz);
```

where `np` et `nz` represent the number of poles and zeros of the transfer function to be estimated (a time-delay can also be estimated along with the transfer function coefficients but this is not exploited here).
The estimated model parameters along with their standard deviations can be displayed by using the following command:

```
present(Gtfsrivc);
```

1. Type the following commands:

   ```
   IODelay=0;
    Gtfsrivc=tfsrivc(data,2,0,'TdMax',IODelay);
    present(Gtfsrivc);
    present(Go);
   ```

The upper time-delay boundary 'TdMax' is set to 0 so that no time-delay is estimated.

2. Compare the TFSRIVC estimates with those obtained by using `lssvf` and `ivsvf`. Run several times your program to get a feel for the overall performance of the SVF-based and TFSRIVC estimators.

3. Set the standard deviation of the additive measurement noise to 0.4 instead of 0.2 and investigate the robustness of the SVF-based and TFSRIVC estimation methods against the measurement noise. Run several times your program to get a feel for the overall performance of the TFSRIVC estimator.

Tutorials 5

System Identification of the Blue Robotics T200 Thruster

Downloading of the data needed for the tutorial

- Download the zipped file **Tutorial5_SYSID.zip** from the course website and save it in your Matlab working directory.

Exercise - Identification of a transfer function model of the Blue Robotics T200 Thruster

The T200 Thruster from Blue Robotics, shown in Figure 5.1, is one of the world's most popular underwater thruster for ROVs, AUVs or surface vessels. Its flooded motor design makes it compact but efficient and powerful.

Figure 5.1: The T200 Thruster from Blue Robotics.

The T200 thruster is in use on thousands of marine robotic vehicles including the BlueROV2 as shown in Figure 5.2.

Figure 5.2: The marine robotic vehicle BlueROV2 - `www.bluerobotics.com`.

1. Run the file `import_thruster_data.mlx` in Matlab that loads dataset[1] from two different inputs (sine and square chirp signals) applied to a Blue Robotics T200 Thruster. You can watch a short video showing one of the two experimental tests at: `www.youtube.com/watch?v=RU_LOtrE0Bo` (go to the 17mn25).

2. Choose one dataset to estimate a transfer function model of the T200 thruster and the second dataset to validate your model.

This exercise will be evaluated from an oral presentation.

Below is a list of the steps that should be included in your presentation:

1. Short description of the process (input, measured output, etc)

2. Description of the experiment design (type of inputs, dataset available, possible constraints, etc)

3. Description of the raw dataset

4. Data pre-processing (informative data selection, discard of the section with nonlinearity effects, pre-filtering, etc)

5. Model structure selection (method used and results)

6. Parameter estimation (method(s) used and results)

7. Model validation.

8. Conclusion. Discuss the strengths/weaknesses of the identified model(s) and specify which model do you recommend and why.

Your presentation should be limited to 10 mn followed by 5 mn of questions. The number of slides should be therefore limited to 12.

Send your presentation in a pdf format along with the .mlx file you developed to obtain the estimate and validate your model to the instructor by **Thursday, March 10 at 8:00 am**.

[1] MathWorks Student Competitions Team (2022). MATLAB and Simulink Robotics Arena : From Data to Model (https://www.mathworks.com/matlabcentral/fileexchange/65919-matlab-and-simulink-robotics-arena-from-data-to-model), MATLAB Central File Exchange