Data-driven learning of dynamical systems

Hugues Garnier

March 2024

Contents

Tutorials 1

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

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.
- 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 - 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.

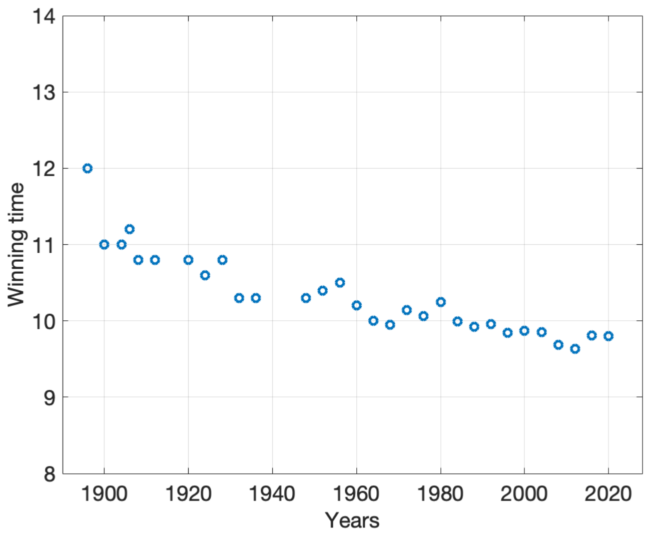

Figure 1.1: Winning men's 100 m times at the Summer Olympics since 1896 and the estimated straight line model. Note that the two world wars interrupted the games in 1914, 1940 and 1944.

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.
7. We now choose to postulate a polynomial model to capture the winning time decrease as a parabola function
$$winning\_time(years) = a \times years^2 + b \times years + c$$
Estimate by simple LS the parameters of the parabola model and forecast the Gold medal men's 100m time in 2024 in Paris.

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 data scientist available at: `machinelearnia.com/regression-lineaire-simple/`

Exercise 1.2 - 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.

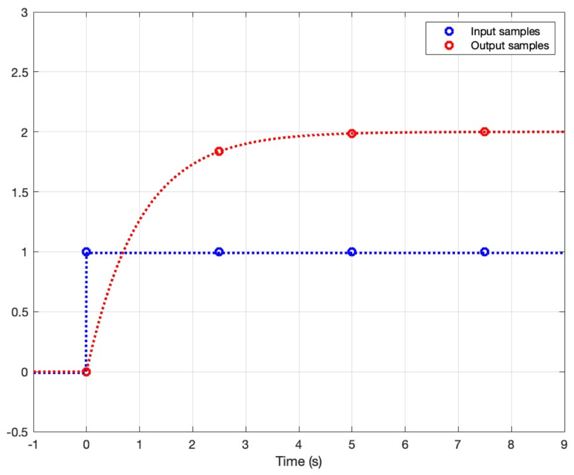

Figure 1.2: Step response of a first-order dynamic system.

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 2

Iterative model learning workflow

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 - The iterative model learning workflow by examples
The iterative model learning workflow is represented in Figure 2.1.

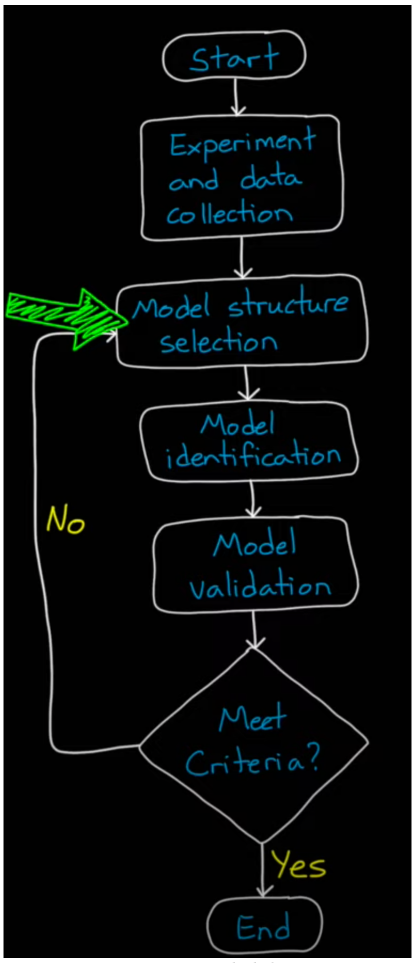

Figure 2.1: Iterative model learning workflow.

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.

Tutorials 3

Application of the data-driven model learning procedure with simulation data

This tutorial session will be evaluated through a report and an oral presentation.

Downloading of the data needed for the tutorial

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

3.1 Model order selection and model validation

One of the most difficult stages in the data-driven model learning procedure is to select a model structure and decide about the model complexity : how to choose the different polynomial degrees (or equivalently fix the number of parameters to be determined for each polynomial of the chosen model).
To select a model structure, a suitable workflow is to follow the procedure recommended during the lectures which is summarized below:

1. Start with non-parametric response or estimates (step response or frequency-responses). This can provide useful information about the time-delay, system order and bandwidth.

2. Begin with a COE model structure.

3. Identify COE models for a range of possible polynomial degrees and time-delays. Select the model order that has the most negative YIC with the highest associated coefficient of determination R_T^2 according to the flexibility/parsimony trade-off.

4. Check out the properties of the estimated COE model, like its ability to simulate the measured data and other traditional model validation tests (residual analysis, physical interpretation of the main features of the identified model, ...) and cross-validation if another dataset is available.

5. Try a more complex model structure such as a hybrid Box-Jenkins model, if necessary.

To become familiar with the recommended start-up procedure, 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 first demo is finished, select then *Determining Model order and Input Delay* and follow the demo displayed in the command window.

Quit to come back to the main menu window.

3.2 A simulated system

The goal is to test first your new model learning skills on simulation data.

From the course website, download the file named: `Simulated_data.zip` which includes the following three files:

- `Simulated_stepdata.mat` contains the step response,
- `Simulated_data1.mat` contains the system response to a PRBS input,
- `Simulated_data2.mat` contains the system response to a different PRBS input.

The input of this simulated system is stored in `u` while the output is stored in `y`. The sampling period is specified in `Ts`.

The three datasets have been generated from a simulated model of a LTI system. There is thus a "correct solution (a true model)" to it.

1. By using the start-up procedure outlined in Section 3.1, try to figure out the best model for the data.
2. Check out the properties of the estimated model, like its ability to reproduce the simulated data and other traditional model validation tests (residual analysis, etc).
3. Use the second file to cross-validate your model.
4. Compute the steady-state gain of your model (see `dcgain routine` along with the cut-off frequencies (in rad/s) and damping ratio(s) (see `damp routine` of your estimated model.
5. Present the main characteristics of your identified model in a table.