



# Identification of dynamical systems

Hugues Garnier

February 2025

# Contents


1	Simple least squares for estimating the parameters of time-series and static models	1
2	Simple least squares for estimating the parameters of linear dynamical models	3
3	Iterative model learning workflow	5
4	Identification of continuous-time transfer function models with the CONTSID toolbox	7

# Tutorials 1

---

## Simple least squares for estimating the parameters of time-series and static 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 - Modeling the thrust of the Crazyflie nano-quadcopter

The file `thrustdata.mat` contains the thrust (in N) for different values of the rotor speed (in rpm) generated by a Crazyflie nano-quadcopter as shown is Figure 1.1.

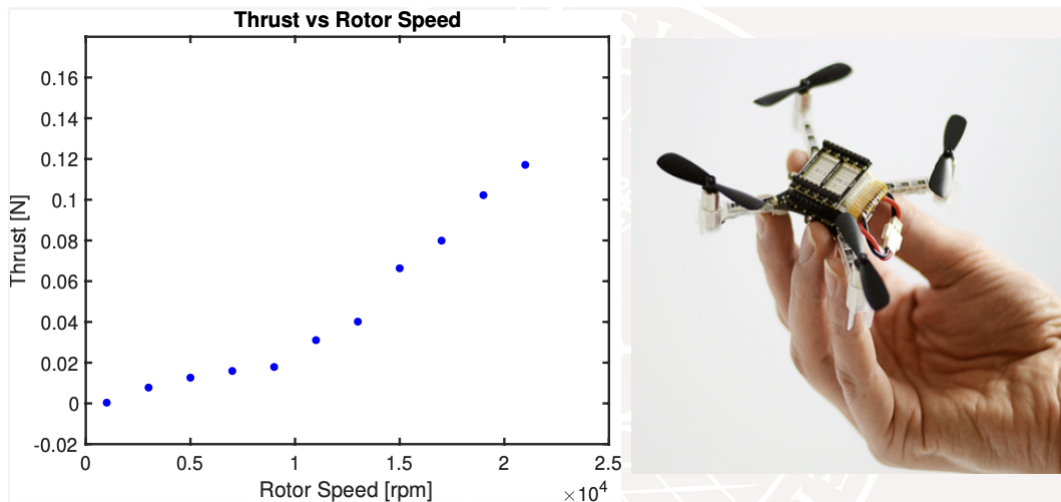


Figure 1.1: Thrust versus rotor speed for the Crazyflie nano-quadcopter

The goal is to model the relationship between the rotor speed and the generated thrust

1. Load the dataset in Matlab.
2. Plot the data and observe the thrust increase in terms of the rotor speed.
3. We choose to postulate a polynomial model of order 1 between the two signals.

$$\text{thrust} = a \times \text{speed} + b$$

Determine by using simple least squares (LS) the parameters of the first order polynomial model.

4. Plot the estimated and the measured thrust in the same figure.
5. Compute the residuals along with the RMSE performance indice.
6. We now choose to postulate a polynomial model to capture the thrust increase as a parabola function of the motor speed

$$\text{thrust} = a \times \text{speed}^2 + b \times \text{speed} + c$$

Estimate by simple LS the parameters of the parabola model


7. Plot the estimated and the measured thrust in the same figure. Compute the RMSE.
8. Increase the order of the polynomial model up to 10 and estimate the parameters.
9. For every order, plot the estimated and the measured thrust in the same figure. Compute the RMSE.
10. Select the best order of the polynomial model. Explain your choice.

# Tutorials 2

---

## Simple least squares for estimating the parameters of linear dynamical models

### 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 estimate the continuous-time and discrete-time model of a dynamical system from step response data**

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

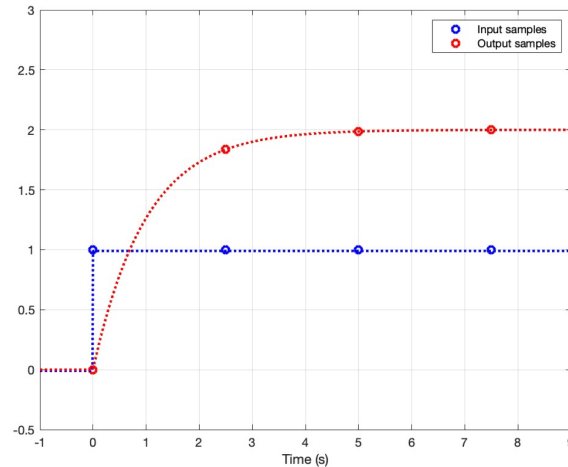


Figure 2.1: Noise-free step response of a first-order dynamical system.

The file `step_response_data.mat` includes 4 samples of the step response of a first-order dynamical system as shown in Figure 2.1.

1. Load the dataset in Matlab.
2. Plot the noise-free input and output step response data. Observe the input `u` and output samples stored in `y`. The output time-derivative samples are also stored in `ydot`.
3. We first choose to postulate a continuous-time first-order transfer function model to capture the system dynamics from the step response data.

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

4. Look at the lecture slides and determine by using simple least squares the parameters  $a$  and  $b$  of the continuous-time transfer function model.
5. Compare the estimated values with the true continuous-time model parameters.
6. We now choose to postulate a discrete-time first-order transfer function model to capture the system dynamic from the step response data.

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


7. Look at the lecture slides and determine by using simple least squares the parameters of the discrete-time transfer function model.
8. Compare the estimated values with the true discrete-time model parameters.
9. Repeat the continuous-time and discrete-time model identification by using the noisy output `ynoisy` and noisy output time-derivative `ydotnoisy`.
10. Conclude about the use of simple least squares in presence of noisy output measurements.

# Tutorials 3

---

## Iterative model learning workflow

### 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.

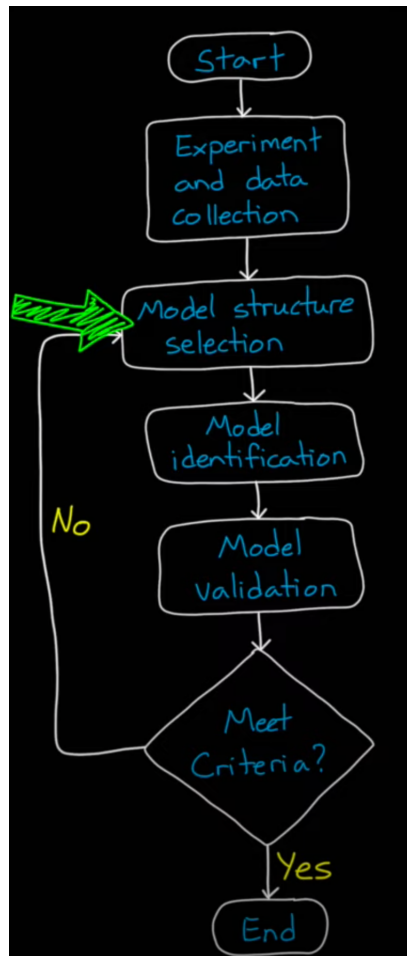


Figure 3.1: Iterative model learning workflow.

1. Watch the video (18 mn) by Brian Douglas entitled **Linear System Identification - Part 2** available at:  
[www.youtube.com/watch?v=qC\\_C04SEV1E](http://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 4

---

## Identification of continuous-time transfer function models with the CONTSID toolbox

- We will mainly use the CONTInuous-Time System IDentification (CONTSID) toolbox which is a collection of m-files written in Matlab. The toolbox has been developed at Polytech Nancy.
- It is freely available for academic researchers and students. It can be downloaded from: [www.cran.univ-lorraine.fr/contsid/](http://www.cran.univ-lorraine.fr/contsid/)
- The toolbox can be run in command-line mode or in a graphical user interface (GUI) mode or in any combination of these. We will use the command-line mode only.
- The exercices are intended for the CONTSID toolbox 7.5 to be run with Matlab version R2024a equipped with the System Identification toolbox and the Control toolbox.

### Downloading of the data needed for the tutorial

- Download the zipped file of the **CONTSID** toolbox from its 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.

### Exercise 4.1 - Tutorial introduction to the CONTSID toolbox

This first exercise is meant as an introduction to data-driven system identification with the CONTSID toolbox. You will run an identification demonstration program available in the toolbox.

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.

### 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.

#### Data-generating system

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

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

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  $\sigma_e^2$ .

The polynomials  $B_o(s)$  and  $F_o(s)$  are defined by:

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

1. Determine from (4.1) and (4.2) the main features of the true system: order, steady-state gain and time-constants

For a PRBS input and white 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 `.m` or `.mlx` script:

```
s = tf('s');
S = 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(S,u,t); % The noise-free output
y=x+e; % The noisy output
```

For later use with the CONTSID toolbox estimation routines (PROCSRIVC and 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 available in the CONTSID toolbox from the two simulated datasets.

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.

### Low-order process model identification by PROCSRIVC

Consider first the identification of a low-order (2nd order here) PROCess model by the Simple Refined IV method for Continuous-time models (PROCSRIVC):

$$\mathcal{M}_{\text{procsrivc}} : y(t_k) = \frac{K_p}{(1 + T_{p1}s)(1 + T_{p2}s)} u(t_k) + e(t_k) \quad (4.3)$$

1. The CONTSID `procsrivc` routine can be used as follows. Type the following commands:

```
Model_type = idproc("P2");
Mprocsrivc=procsrivc(data,Model_type);
```

The estimated model parameters along with their standard deviations can be displayed by using the following command:

```
present(Mprocsrivic);
```

2. Compare the PROCSRIVC estimates with the true system parameters

```
present(S);
```

3. Run several times your program to get a feel for the overall performance of the PROCSRIVC model estimate.
4. Set the standard deviation of the additive measurement noise to 0.4 instead of 0.2 and investigate the robustness of the PROCSRIVC estimation methods against the measurement noise. Run several times your program to get a feel for the overall performance of the PROCSRIVC estimation routine.

### Transfer function model identification by TFSRIVC

We now consider the identification of a 2nd order Transfer Function by the Simple Refined IV method for Continuous-time models (TFSRIVC):

$$\mathcal{M}_{\text{tfsrivic}} : y(t_k) = \frac{b_0}{s^2 + a_1s + a_2} u(t_k) + e(t_k) \quad (4.4)$$

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

```
Mtfsrivic=tfsrivic(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(Mtfsrivic);
```

1. Type the following commands:

```
IODelay=0;  
Mtfsrivic=tfsrivic(data,2,0,'TdMax',IODelay);  
present(Mtfsrivic);
```

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

2. Compare the TFSRIVC estimates with the true true system parameters.

```
present(S);
```

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