

Modeling and Estimation for Control

Lab topics

Instructors:

Authors:

Emmanuel WITRANT

Ioan LANDAU

Technical support:

Marie-Pierre VAILLANT

University Grenoble Alpes,
UFR PhITEM,
International Master in Systems, Control & IT

September 16, 2025

Contents

1	Organisation: $6 \times 4h30$	2
2	Vibration Isolation for Heavy Trucks.	3
2.1	Simplified model (lab 1)	3
2.1.1	Perform the three phases of modeling	3
2.1.2	Model with bond graph	4
2.2	Design a simulator for the heavy truck (optional)	4
3	Identification of an Active Vibration Control Benchmark Using Matlab	6
3.1	System description and preliminary work	6
3.2	Experiment design and frequency-response analysis	7
3.3	Spectral analysis	7
3.4	Estimating the disturbance spectrum	7
3.5	Conclusions	7
4	Linear Modeling of an Electric Throttle Valve (2 labs)	8
4.1	Electric throttles in powertrain systems	8
4.2	Control of the electric throttle valve with Arduino	9
4.3	Lab preparation	9
4.4	Preliminary experiments	9
4.5	Experiment design for system identification	12
4.6	Parameter estimation in linear models	15
4.7	Digital control design	17
4.8	Appendix (excerpt from [1])	21
5	Modeling and Estimation of an mBot Mobile Robot (2 labs)	25
5.1	System Description	25
5.2	Modeling and transfer functions	26
5.3	Robot Programming	26
5.4	Parameter estimation	27
5.5	Conclusions	27

Chapter 1

Organisation: $6 \times 4\text{h}30$

Isolation for Heavy Trucks. Reference material: Lessons 1-4, Principles of Physical Modeling, Some Basic Relationships in Physics and Bond Graphs.

Modeling and Estimation of an mBot Mobile Robot (lab 1) Reference material: Lessons 1-4, Principles of Physical Modeling, Some Basic Relationships in Physics and Bond Graphs. Up to item 3 in 5.3 in the simulation lab.

Identification of an Active Vibration Control Benchmark Using Matlab Reference material: Lessons 7-8, Experiment Design for System Identification and Non-parametric Identification.

Linear Modeling of an Electric Throttle Valve (2 labs) Reference material: Lessons 7-8, Experiment Design for System Identification and Non-parametric Identification (first lab) then Lesson 9 Parameter Estimation in Linear Models and Digital Control (review).

Modeling and Estimation of an mBot Mobile Robot (lab 2) Reference material: Lesson 9 and 12 Parameter Estimation in Linear Models and Recursive Estimation Methods

Chapter 2

Vibration Isolation for Heavy Trucks.

Reference material: Lessons 1-4, Principles of Physical Modeling, Some Basic Relationships in Physics and Bond Graphs.

A company has developed a vibration isolation system for the use on the back of heavy truck cabs. These cabs are pivoted with stiff mounts on the fronts and the vibration isolators are located at the left and right hand side of the cab at the rear. The isolators consists of fluid filled displacers (piston/cylinder devices) attached by a fluid filled tube (with an orifice) to an air compressed accumulator tank. The displacers are about 10 cm in diameter and the tube is about 2.5 cm in diameter.

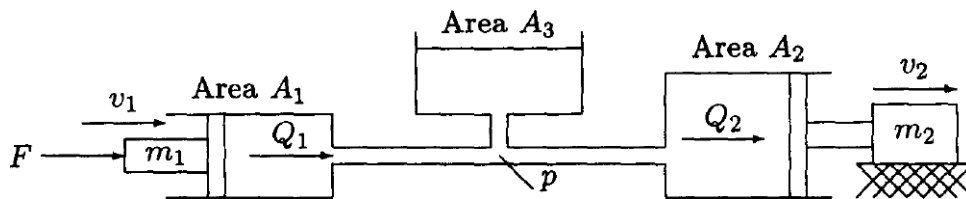


Cabs pivoted at front

Vibration isolator mounted here

2.1 Simplified model (lab 1)

First consider a mechanical-hydraulic example as shown in:



The force F acts on a piston with mass m_1 and cross section A_1 . This results in a flow Q_1 , which is partly stored in a tank with cross section A_3 and partly goes on as Q_2 to a piston with cross section A_2 . This piston moves the mass m_2 with velocity v_2 over a surface with friction $F_f = bv_2$. The pressure in the connecting tube is p . Starting with the left cylinder, we see that the energy is partly stored in the mass m_1 and partly transferred to the hydraulic subsystem. This transfer of energy is, as we have seen, represented by a transformer.

2.1.1 Perform the three phases of modeling

1. structuring the problem:
 - (a) divide the system into subsystems
 - (b) define the important variables
 - (c) for each subsystem, identify the interaction variables
 - (d) draw a block diagram of the complete system

2. setting up the Basic Equations:

- (a) recall the main physical laws that are necessary to solve this problem
- (b) which blocks have a dynamic behavior? which ones are static?
- (c) associate an equation to each block

3. forming the State-Space Model:

- (a) combine the equations obtained previously to get an algebro-differential set of equation
- (b) clearly define the state variables
- (c) rewrite the model as a set of first order differential equations

2.1.2 Model with bond graph

For this system:

1. obtain the bond graph model;
2. simulate this model with 20-SIM, using the numerical values below, and plot the response of the mass velocity (v_2) to two different input force F profiles:
 - a unit step;
 - sinusoids at different frequencies;

show your bond graph, simulation plots and model equations in your report.

3. write the state-space equations using the automatic translation of bond graphs to equations and discuss a configuration of sensors that guarantees the observability of the system.

NB: for the numerical application you can take:

$$\begin{array}{lll} A_1 = \pi(3 \times 10^{-2})^2 \text{ m}^2 & A_2 = \pi(5 \times 10^{-2})^2 \text{ m}^2 & A_3 = \pi(20 \times 10^{-2})^2 \text{ m}^2 \\ m_1 = 5 \text{ kg} & m_2 = 100 \text{ kg} & \rho = 1000 \text{ kg/m}^3 \\ g = 9.8 \text{ m/s}^2 & b = 10 & C_f = A_3/(\rho g) \end{array}$$

2.2 Design a simulator for the heavy truck (optional)

Consider the schematic representation of the vibration insulation system proposed in Fig. 2.1 and the following simplifications:

- H1. since we are only interested in how the system deviates from an equilibrium state and the system is linear, you can disregard the influence of gravity on the cab and the isolation system, e.g. v_{in} is the only input in the system. This means that the accumulator tank behaves like a standard tank and you can denote its fluid capacitance by C_f ;
- H2. the wheel dynamics can be disregarded;
- H3. inertance in the cylinders can be disregarded, i.e. the only inertance in the fluid system is in the tube;
- H4. friction in the tube can be neglected;
- H5. the mass in the pistons can be neglected;

Perform the following:

1. Draw a bond-graph of the isolation system model, first without the simplifying hypotheses and then with H2 to H5 (keep the gravity effect). Provide the simplification steps. The velocity v_{in} from the road surface is an input in the system.
2. Based on the bond-graph description, determine a state-space model for the system. The velocity v of the truck cab is the output.
3. Simulate the truck vibrations with Simulink, setting arbitrarily $A = \pi(.05)^2$, $R_f = 10$, $L_f = .01$, $C_f = 5$, $m = 500$ and v_{in} as a noisy sine function with a 50 Hz period.

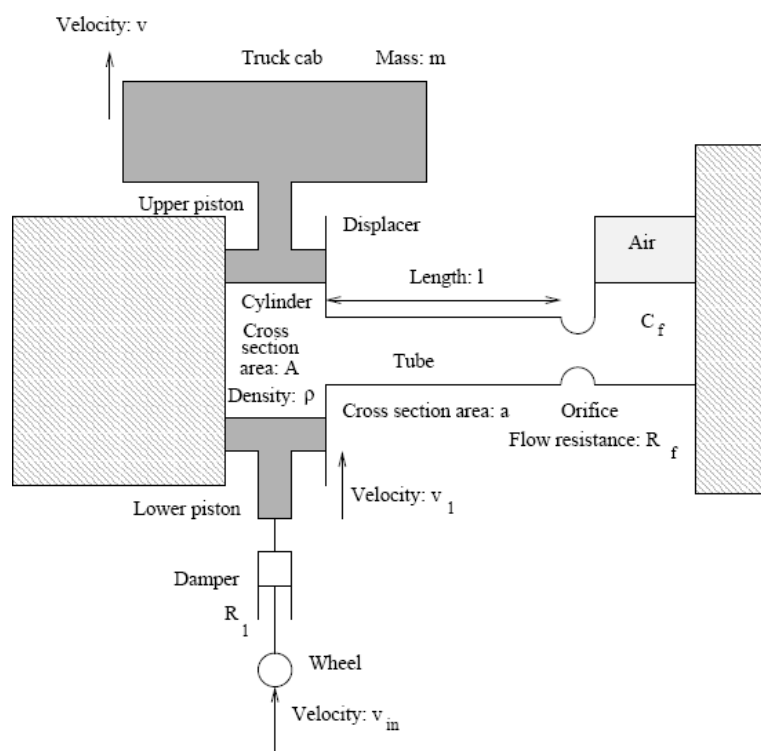


Figure 2.1: Vibration isolation system.

Chapter 3

Identification of an Active Vibration Control Benchmark Using Matlab

Reference material: Lessons 7-8, Experiment Design for System Identification and Non-parametric Identification.

As stated in *Wikipedia*, “active vibration control is the active application of force in an equal and opposite fashion to the forces imposed by external vibration. With this application, a precision industrial process can be maintained on a platform essentially vibration-free. Many precision industrial processes cannot take place if the machinery is being affected by vibration. For example, the production of semiconductor wafers, or for reducing vibration in helicopters, offering better comfort with less weight than traditional passive technologies. In the past, only passive techniques were used. These include traditional vibration dampers, shock absorbers, and base isolation.”

In this lab, you will learn how to use *System Identification* methods seen in class in order to model a benchmark on vibration control recently proposed by *GIPSA-lab* [2].

3.1 System description and preliminary work

The active vibration compensation experiment is depicted in Figure 3.1. It can be basically considered as a LTI system with two inputs (the controlled force u and the disturbance force u_p) and one output: the residual force y .



Figure 3.1: *GIPSA-lab* vibration benchmark.

In order to prepare the lab properly:

1. Download all the files in the class website and read through the system description `BenchmarkV1.1.pdf`. Check for additional information in [2].
2. Synthesize the information (2 pages summary) that is made available concerning the system properties (number of input and output channels, derivative effects), data (sampling frequencies, inputs properties) and simulation files.

In order to illustrate the key concepts seen in class, do the following.

3.2 Experiment design and frequency-response analysis

1. Load a data set (i.e. `data_prim1.mat`) and visualize the input data `u`:
 - (a) which kind of signal is it?
 - (b) superpose this signal every $N = 2^{10} - 1$ samples, i.e. using
`i = 1:N; plot(i,u(i),i,u(i+N),i,u(i+2*N))`
what can you conclude on the input sequence?
 - (c) estimate its power spectral density (i.e. with `pwelch`) and conclude on the frequency contents;
 - (d) discuss the properties and advantages of such an input for system identification.
2. Input/output relationships:
 - (a) load and comment the function `freq_resp_cal` provided on the website;
 - (b) plot on the same graph the magnitude of the frequency obtained by spectral analysis for the primary and secondary channels;
 - (c) what happens when the PRBS clock frequency is divided by two?
 - (d) discuss the similarities and differences between the two channels (common poles/zeros, order, etc.).
3. Conclude with some guidelines concerning the model architecture.

Note: if the *Matlab Signal Processing Toolbox* is not installed on your computer, you may download the *octave-signal* free software at <http://octave-signal.sourceforge.com/>

3.3 Spectral analysis

1. Empirical transfer-function estimate (ETFE):
 - (a) compute the ETFE (i.e. with `etfe`) of both channels and discuss your results;
 - (b) smooth the ETFE thanks to Hamming window and vary the window parameter γ (i.e. with `spa`, `getff` and `bode`);
 - (c) for each γ , compute $M(\gamma)$ and $\bar{W}(\gamma)$. Relate these parameters to the smoothed ETFEs obtained above.
2. Autocorrelation analysis:
 - (a) compute the correlation between inputs and outputs for both channels;
 - (b) what is the periodicity of the peaks (in number of samples)? What is the implication of this observation?
 - (c) by zooming on the cross correlations (choosing one period), how many past values affect the actual value (horizon in terms of sampling periods)?
 - (d) what can be concluded on the model order, or the presence of delays, for both channels?

Note: the material *Digital Signal Processing* proposed in [3] may be useful for this analysis.

3.4 Estimating the disturbance spectrum

1. Identify (with *Matlab Identification Processing Toolbox*) the decoupled data thanks to a 6th order state-space model with to the prediction-error method.
2. Using this model, compute the residual and its spectrum.
3. Discuss the model choice and compare it with the other models easily available.
4. Set the experimental inputs as common inputs in the model provided in `simulator_bench.mdl` and in the previous state-space model, and compare the two outputs with the experimental data.

3.5 Conclusions

Summarize your main results and conclude.

Chapter 4

Linear Modeling of an Electric Throttle Valve (2 labs)

Reference material: Lessons 7-8, Experiment Design for System Identification and Non-parametric Identification (first lab) then Lesson 9 Parameter Estimation in Linear Models and Digital Control (review).

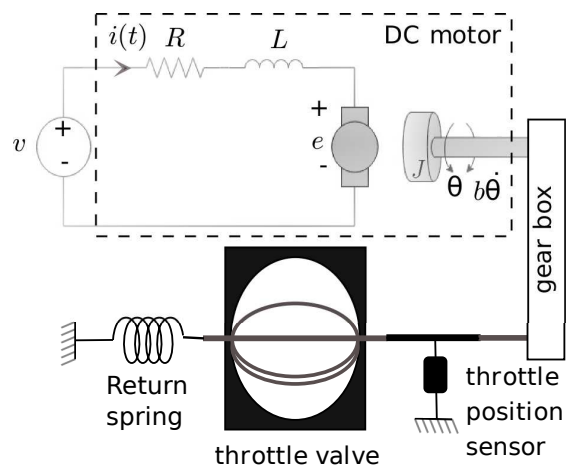
4.1 Electric throttles in powertrain systems

As stated in [4]:

“ The electric throttle is an important component of the powertrain system of nowadays automotive vehicles whose advanced features, such as cruise control, are at the core of an enjoyable driving experience. It consists of a throttle flap, a DC motor, one or two springs (a low limit switch spring and a high limit switch spring), reduction gears, and a group of sensors, as depicted in Figure 4.1. Its job is to regulate the amount of air entering an internal combustion engine, hence playing an important role in its performance, fuel economy, and pollution aspect. When the rotating flap is fully-open, it allows 100 % of air flow into the engine when it's fully-closed, it allows no air flow. A reliable throttle model is the foundation of its design and control, because it permits to test and improve the design early enough in the construction process to save much time and money. However, the electric throttle presents a modeling challenge, despite its simplicity, due to its nonlinearities, present in its dual spring system which functions alternatively under the limp-home position and above it, its gear system, its coulomb friction torque, and preload torque. Good modeling of an electric throttle is worth the effort because of its importance in the overall engine control, in the model-in-the-loop (MIL) phase, and in the hardware-in-the-loop (HIL) phase. Moreover, the electric throttle along with the actuators of the electric drive chain, such as the high-pressure and low-pressure exhaust-gas-return (EGR) valve and the air-intake valve in diesel engines, share the same control strategy and the same structure of one or two return springs, a reduction gear, and a valve. Having accurate models and easily-calibrated controllers for these actuators is crucial in the calibration of the gasoline engines subject to the upcoming Euro7 emission norm (which defines the maximum pollution level of the vehicles produced after 2025). ”



(a) Picture of the valve used in the lab



(b) Scheme of a single-spring valve (adapted from [4])

Figure 4.1: Picture and scheme of the electric throttle valve used in this lab.

4.2 Control of the electric throttle valve with Arduino

The lab process is depicted in Figure 4.2. The electric throttle valve has the commercial reference 03L128063, a valve that equipped some Audi cars before 2010 and has only one spring (no non-zero limp home position). As the spring directly restrains the rotation of the shaft of the valve, the torque exerted by the motor results in an angular position (torsional energy storage by the spring, which integrates the velocity). The position of the valve is thus regulated by modulating the input voltage (acting on the input effort, to link this with the bond graph/energy nomenclature), which is done by generating a piecewise modulation (PWM) of the voltage with an Arduino Mega 2560.

The valve is equipped with a programmable magnetic angle sensor KMA221, which provides an analog output ratio-metric to the supply voltage, set to 5 V to match the specification of Arduino's analog inputs. The analog-to-digital converter of the Arduino then converts this signal into a 10-bits one when using the command `analogRead()`.

The 12 V DC motor of the valve is controlled using the SHIELD-MD10, a Cytron 10A motor driver shield for Arduino. This shield uses an NMOS H-Bridge to achieve a speed control PWM frequency up to 10 kHz. The default PWM frequency is increased to avoid an annoying high frequency noise generated by the valve (the free-running timer 3 of the ATmega32 is modified using TCCR3B). The motor is controlled through two digital pins of the Arduino board (selected by the mini jumpers on the shield): one for direction (`pinDIR`, set to HIGH or LOW) and one for the velocity (`pinPWM`, between 0 and 100 %).

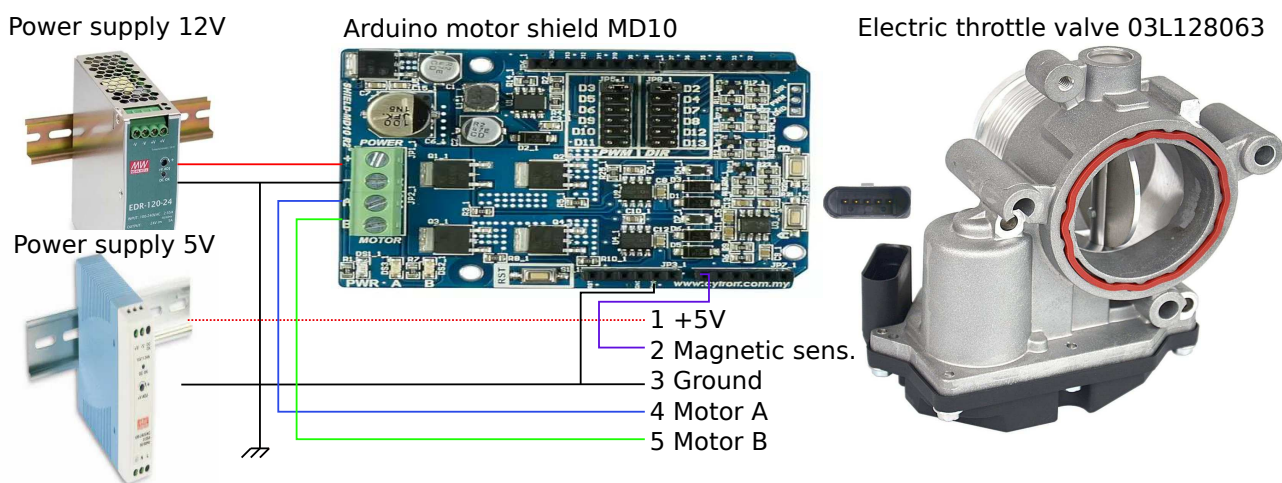


Figure 4.2: Power supply and data acquisition for the electric throttle valve.

4.3 Lab preparation

For the first lab, complete the questions 2a in Section 4.4, 1a and 1b in Section 4.6. For the second lab, complete the questions 1, 2 and 3 in Section 4.7.

4.4 Preliminary experiments

Your first goal is to investigate the input-output properties of this system from simple experiments. To do this,

1. understand the following Arduino code:

```
#define pinSens 0
#define pinPWM 3
#define pinDIR 2

double MeasSens, MeasAngle;
int Sens_Pos_TOP = 1023;
int Sens_Pos_FLR = 91;
int index = 0;
double angle, Input;
int count = 0;
int speed_inc = 5;
```

```

long int time;
long int now;
int ts = 5; // sampling time, [ms]
static int iSpeed = 0; // Speed of the motor.

//-----
void setup() {
  Serial.begin(115200);
  pinMode(pinSens, INPUT);
  pinMode(pinPWM, OUTPUT); // enable the PWM output (you now have a PWM signal on digital pin 3)
  pinMode(pinDIR, OUTPUT);
  time = millis();
  Serial.println("Time,iSpeed,MeasAngle");

  int myEraser = 7; // this is 111 in binary and is used as an eraser
  TCCR3B &= ~myEraser; // this operation (AND plus NOT), set the three bits in TCCR2B to 0
  int myPrescaler = 1; // this could be a number in [1, 6] = [62k, <20] Hz (default 3 = 98
  TCCR3B |= myPrescaler; //this operation (OR), replaces the last three bits in TCCR2B with our ne
}

//-----
void loop() {
  now = millis();
  if (now >= time + ts) {
    time = now;

//-----
//--- To be filled with the appropriate code ---
//-----

    // Set the desired input signal iSpeed on the motor
    if (iSpeed >= 0) {
      analogWrite(pinPWM, iSpeed);
      digitalWrite(pinDIR, LOW);
    }
    else {
      analogWrite(pinPWM, -iSpeed);
      digitalWrite(pinDIR, HIGH);
    }

    // Get the measurement and convert into an angle [degrees]
    MeasSens = analogRead(pinSens);
    MeasAngle = ((MeasSens - Sens_Pos_FLR) / (Sens_Pos_TOP - Sens_Pos_FLR) ) * 90;

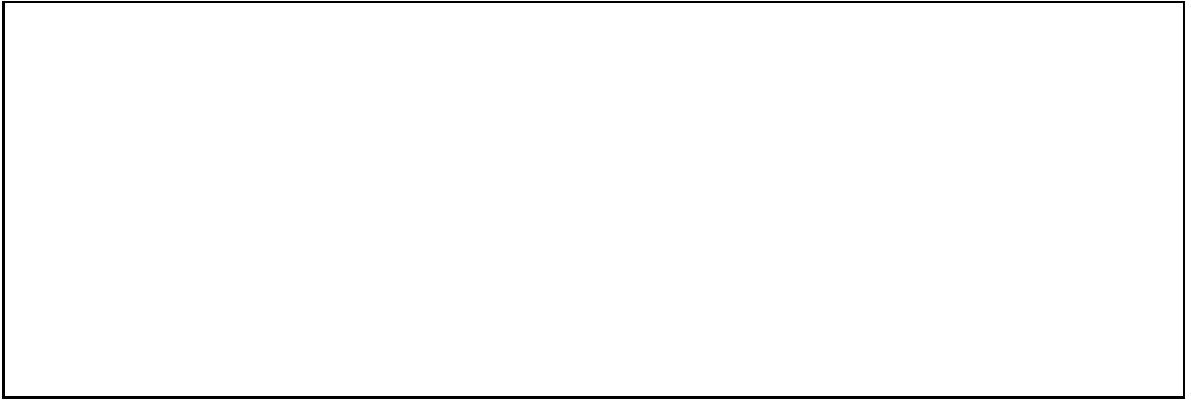
    Serial.print(time);
    Serial.print(",");
    Serial.print(iSpeed);
    Serial.print(",");
    Serial.println(MeasAngle);
  }
}

```

and implement it to test your experiment.

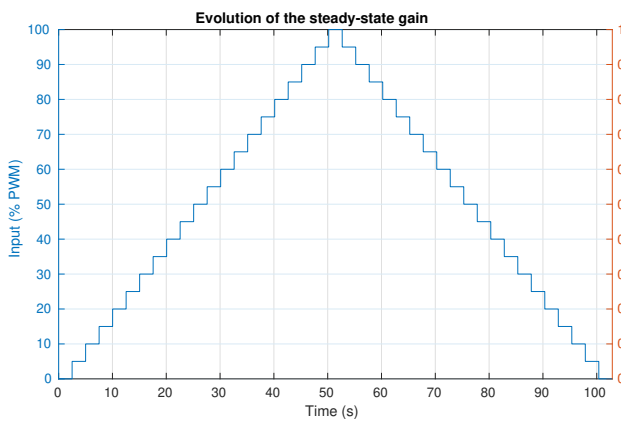
2. Investigate the steady-state behavior of the valve by getting the response of a sequence of steps going back and forth, as depicted in Figure 4.3:

- (a) write the appropriate code that defines iSpeed (missing in the previous one);

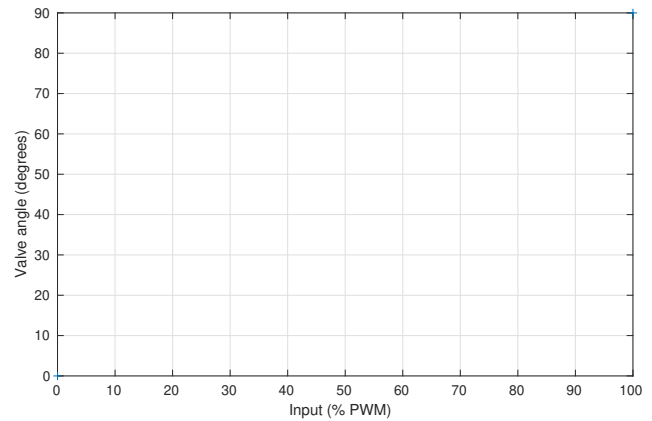


(b) draw the time evolution of the system on Figure 4.3a;

(c) repeat this sequence three times and draw the input-output map on Figure 4.3b, distinguishing the closing and opening tasks;



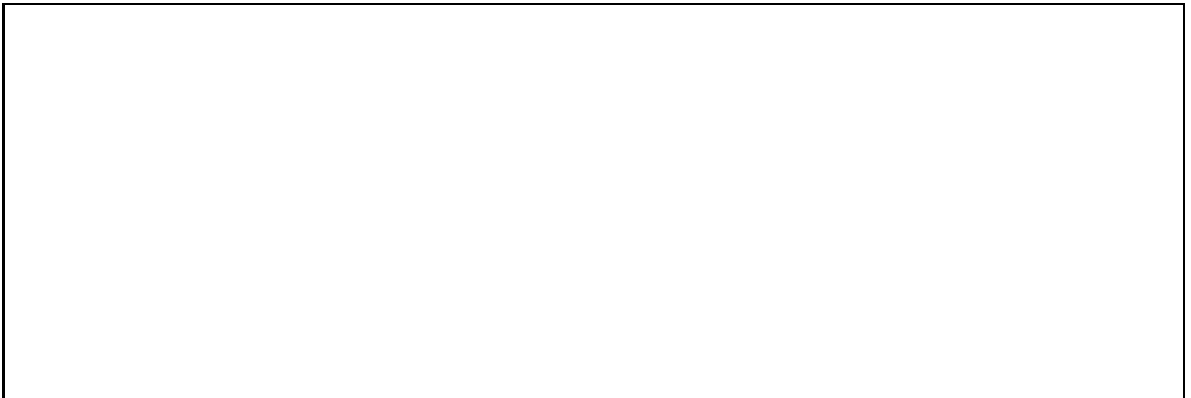
(a) Time evolution



(b) Input-output map

Figure 4.3: Preliminary experiment for the steady-state gain analysis.

(d) conclude on the system properties.



3. Get some first insight on the transient behavior by zooming on step responses at several times when the system is actually moving in response to the input variation:

(a) draw several of these responses on Figure 4.4;

(b) conclude on a “reference rise time” and appropriate choice of the sampling time.

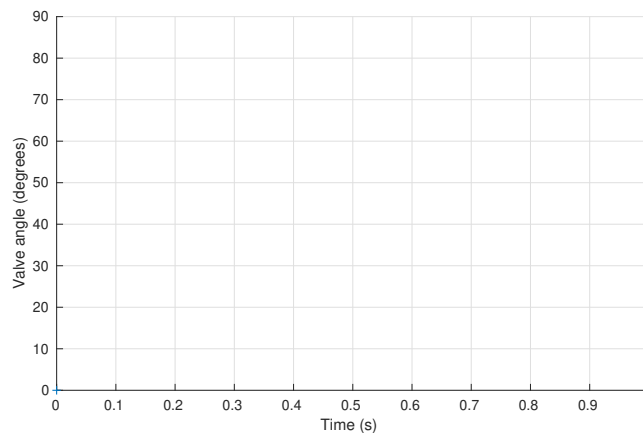


Figure 4.4: Preliminary experiment for the transient analysis: step response.



4.5 Experiment design for system identification

Read the PRBS description and implementation guidelines given in Appendix 4.8. Your goal is to identify the system dynamics for valve angle around 40° .

1. What are the essential tuning parameters for the PRBS sequence? Choose these parameters according to your results obtained in Section 4.4. Hint: you may wish to check also the Arduino memory that is available for your code.



2. Implement the PRBS capability in your code as follows:

- in the preamble (before the `setup()` section), add (where ? have to be replaced by you own values):

```
// PRBS
float tR = ?; // Rise time [ms]
float ValUinit = 0;
float ValMag = ?;
float ValOffset = ?;
const int Valp = ?;
```

```

int ValN = 0;
unsigned long prbs_length = 1;
byte * signal_prbs;

//-----
// Fonction : Creer_prbs( )
// Génère la Séquence Binaire Pseudo-Aléatoire dans un tableau.
// Auteur : Adaptech
// Création du fichier : 14/02/2000
// Modification: E. Witrant, 9/11/2021
//-----
void Creer_prbs(float ValUnit, float ValMag, float ValOffset, int ValN,
int Valp, unsigned long prbs_length)
{
    /* The input parameters are:
    ValUnit: Initial value of the signal
    ValMag: Magnitude of the variation
    ValOffset: Offset
    Valp: ratio between PRBS and sampling frequencies p (between 1 and 3, <=4)
    ValN: nb of register cells (at PRBS frequency) N (between 2 and 32)
    prbs_length: number of PRBS values at sampling frequency = p*(2^N-1)
    After creating the signal, the output is the vector signal_prbs, which contains the
    PRBS values to apply at each sampling time.
    The table
    https://www.digikey.fr/fr/articles/use-readily-available-components-generate-binary-sequences
    is used for 11 < N <= 32
    */

    int k1, k2, j, uiu, prbs[ValN+1];
    int i;

    k1 = ValN - 1;
    k2 = ValN;
    if (ValN == 5) k1 = 3;
    if (ValN == 7) k1 = 4;
    if (ValN == 9) k1 = 5;
    if (ValN == 10) k1 = 7;
    if (ValN == 11) k1 = 9;
    if (ValN == 17) k1 = 14;
    if (ValN == 18) k1 = 11;
    if (ValN == 20) k1 = 17;
    if (ValN == 21) k1 = 19;
    if (ValN == 23) k1 = 18;
    if (ValN == 25) k1 = 22;
    if (ValN == 28) k1 = 25;
    if (ValN == 29) k1 = 27;
    if (ValN == 31) k1 = 28;

    for (i = 1; i <= ValN; i++)
        prbs[i] = 1;

    i = 1;
    while (i <= prbs_length) {
        uiu = -prbs[k1] * prbs[k2];
        if (ValN == 8) uiu = -prbs[2] * prbs[3] * prbs[4] * prbs[ValN];
        if (ValN == 12) uiu = -prbs[1] * prbs[4] * prbs[6] * prbs[ValN];
        if (ValN == 13) uiu = -prbs[1] * prbs[3] * prbs[4] * prbs[ValN];
        if (ValN == 14) uiu = -prbs[1] * prbs[3] * prbs[5] * prbs[ValN];
        if (ValN == 16) uiu = -prbs[4] * prbs[13] * prbs[15] * prbs[ValN];
        if (ValN == 24) uiu = -prbs[17] * prbs[22] * prbs[23] * prbs[ValN];

```

```

    if (ValN == 26) uiu = -prbs[1] * prbs[2] * prbs[6] * prbs[ValN];
    if (ValN == 27) uiu = -prbs[1] * prbs[2] * prbs[5] * prbs[ValN];
    if (ValN == 30) uiu = -prbs[1] * prbs[4] * prbs[6] * prbs[ValN];
    if (ValN == 32) uiu = -prbs[1] * prbs[2] * prbs[22] * prbs[ValN];
    j = 1;

    while (j <= Valp) {
        if (uiu > 0) {
            signal_prbs[i / 8] |= 1 << (i % 8);
        }
        else {
            signal_prbs[i / 8] &= ~(1 << (i % 8));
            // signal_prbs[i - 1] = uiu * ValMag + ValOffset;
        }
        i++;
        j++;
    }
    for (j = ValN; j >= 2; j--)
        prbs[j] = prbs[j - 1];
    prbs[1] = uiu;
}
}

```

- in the setup() section, complete the function with:

```

ValN = ceil(tR/ts/Valp);
if (ValN == 32) Serial.println("The nb of register cells is too large");
for (int i = 1; i <= ValN; i++) {
    prbs_length = prbs_length*2;
}

```

```

Serial.println("");
Serial.print("N = ");
Serial.print(ValN);
Serial.print("; ");
Serial.print(Valp);
Serial.print("; ");

```

```

prbs_length = Valp*(prbs_length-1);
signal_prbs = new byte[(prbs_length / 8)+1];

```

```

Serial.print("PRBS length = ");
Serial.println(prbs_length);

```

```

Creer_prbs(ValUinit, ValMag, ValOffset, ValN, Valp, prbs_length);
Serial.println("PRBS created");

```

- in the loop() section, use:

```

iSpeed = (signal_prbs[count / 8] & (1 << (count % 8)) ? 1 : -1) * ValMag + ValOffset;

```

Note that a for loop is used instead of the pow() function because prbs_length needs to be an integer while pow() returns a double, which tends to give unfaithfull results when converted to an int.

- call the PRBS table in the loop() function and write the corresponding code in your report.

3. Superpose the inputs and outputs of several PRBS sequences. Limit the time range so that we see something and have it checked by your lab teacher. What can you conclude?

4. The Empirical Transfer Function Estimates for two values of p are depicted on Figure 4.5:

- (a) What is the order of the dynamics that you expect to identify?
- (b) Which time constant can be estimated right away?
- (c) How do you interpret the high frequency behavior and how should it be handled?

- (d) Do you think that the simpler response obtained with $p = 3$ is sufficient for control purposes?

4.6 Parameter estimation in linear models

Our goal is now to identify an ARX linear model of arbitrary order

$$y(t) = a_1 y(t-1) + \dots + a_{n_a} y(t-n_a) + b_1 u(t-1) + \dots + b_{n_b} u(t-n_b) + e(t),$$

while minimizing the simplest least squares criterion considered in class:

$$V_N(\theta, Z^N) = \frac{1}{N} \sum_{t=1}^N \frac{1}{2} \|y(t) - \phi^T(t)\theta\|_2^2 \quad (4.1)$$

To do this perform the following.

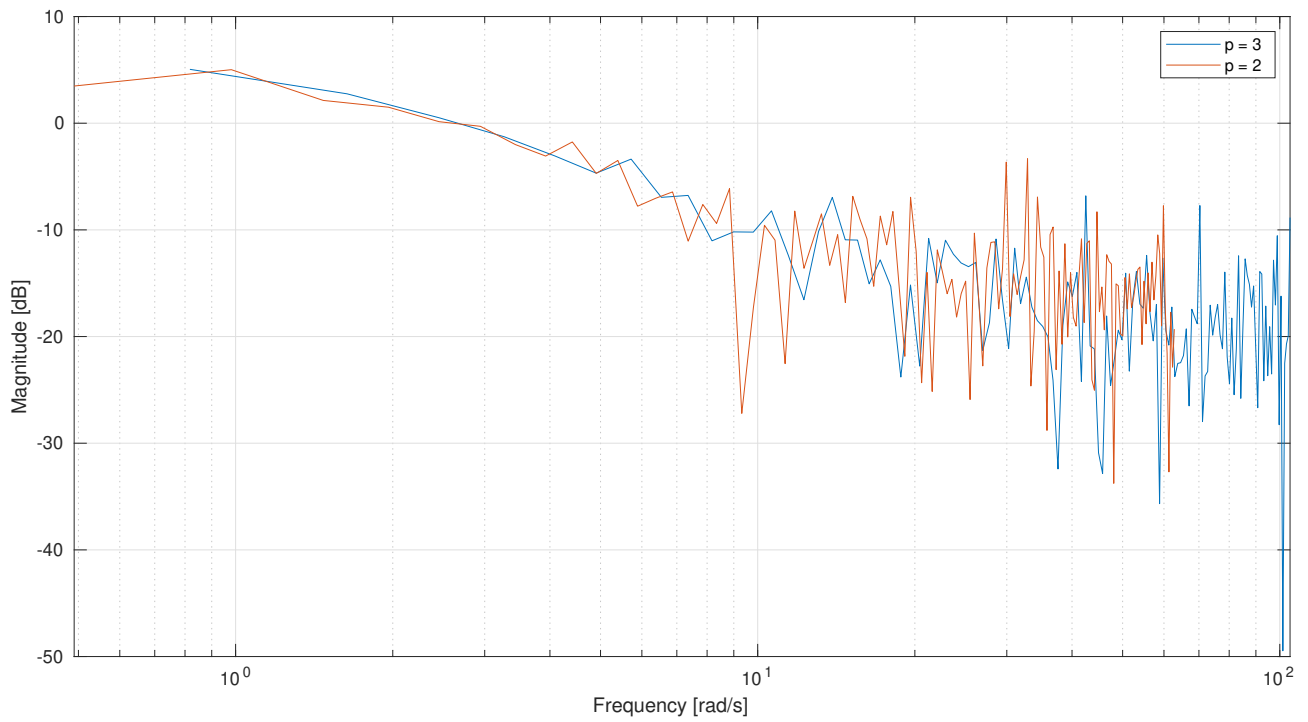


Figure 4.5: Empirical Transfer Function Estimates of the system response to two different PRBS, obtained with $p = 2$ and $p = 3$.

1. First note that the criterion (4.1) cannot be computed in the presented form and doesn't include the multi-experiments aspect.

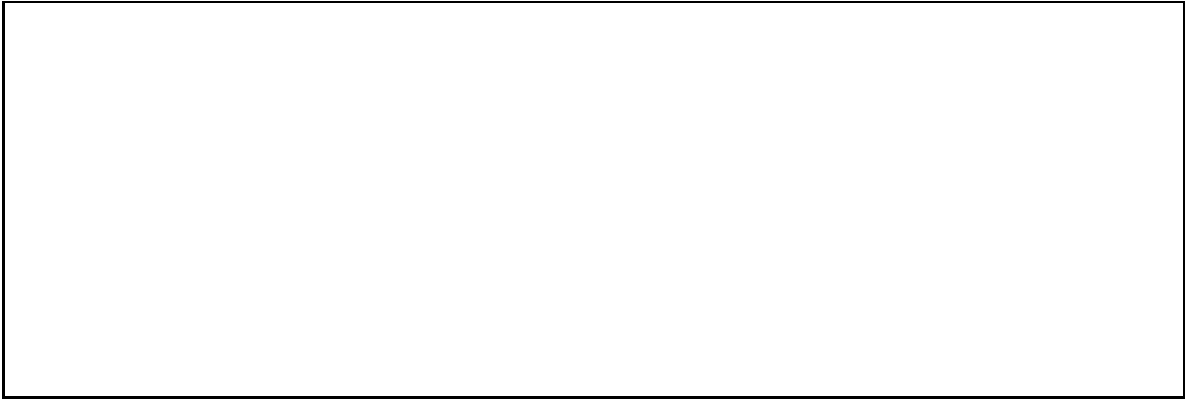
(a) obtain the regressor corresponding to the ARX model and rewrite (4.1) such that it can be computed;



(b) include the possibility to consider an input delayed by d samples;



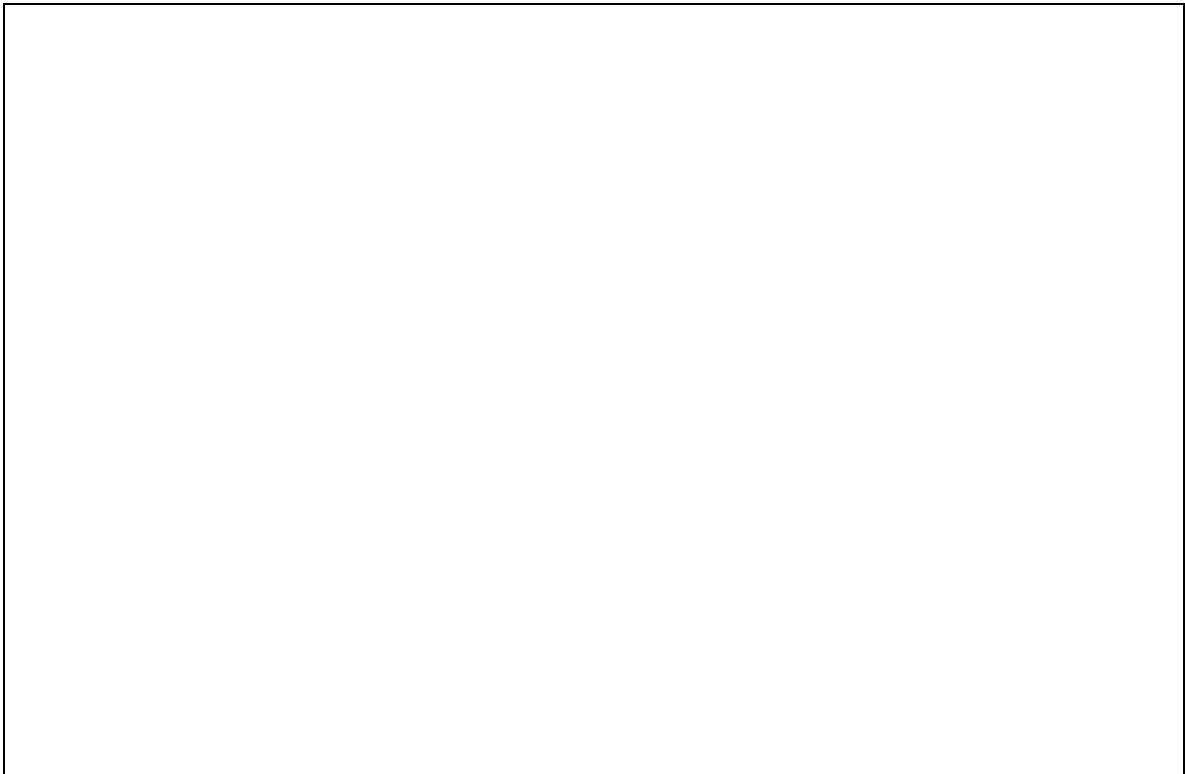
(c) consider the multi-experiments case by considering that they are carried in a continuous loop (e.g. we don't have to consider an initialization of the system between each experiment) and give the vector of parameters that minimizes this criterion;



2. Write an algorithm that computes the set of parameters.



3. Generate a random data sequence in the model validity range (e.g. using the Arduino function `rand()`) and evaluate your identified model on it.
4. Play with the model orders and conclude.



4.7 Digital control design

The aim of this section is to design and implement simple digital controllers using basic knowledge on PID tuning methods.

Consider the continuous-time PI controller:

$$u(t) = K \left[1 + \frac{1}{pT_i} \right] (r(t) - y(t))$$

where $u(t)$ is the control input, $r(t)$ the reference, $y(t)$ the measured output, K the proportional gain and K/T_i the integral gain. Approximating $p \approx (1 - q^{-1})/T_s$ (with Euler backward's method), we obtain the discretized version (see [5], p 60):

$$u(t) = u(t-1) - r_0 y(t) - r_1 y(t-1) + r_0 r(t) + r_1 r(t-1)$$

Considering a first order system discretized, with zero-order hold, as:

$$H(q^{-1}) = \frac{b_1 q^{-1}}{1 + a_1 q^{-1}}$$

the denominator of the closed-loop transfer function is (see [5], p 65):

$$1 + (a_1 - 1 + r_0 b_1) q^{-1} + (b_1 r_1 - a_1) q^{-2} = 1 + p_1 q^{-1} + p_2 q^{-2}$$

and we have that:

$$r_0 = \frac{p_1 - a_1 + 1}{b_1} \quad r_1 = \frac{p_2 + a_1}{b_1}$$

A pole placement design specifying p_1 and p_2 can thus be directly implemented in the controller.

Design the following controllers:

1. a controller that sets the closed-loop response as the response of a first order system with a time constant half of the initial rise time (hint: use Table 2.4 excerpt from [5] given in the Appendix);

2. a controller that sets the closed-loop response as the response of a second order system with a time constant half of the initial rise time and an overshoot of 2-3 % (hint: use Figures 1.10 and 1.11 excerpt from [5] given in the Appendix);

3. from the transfer function that you obtained with $n_a = 1$, $n_b = 2$ and $d = 0$, calculate the fractional delay and the equivalent transfer function:

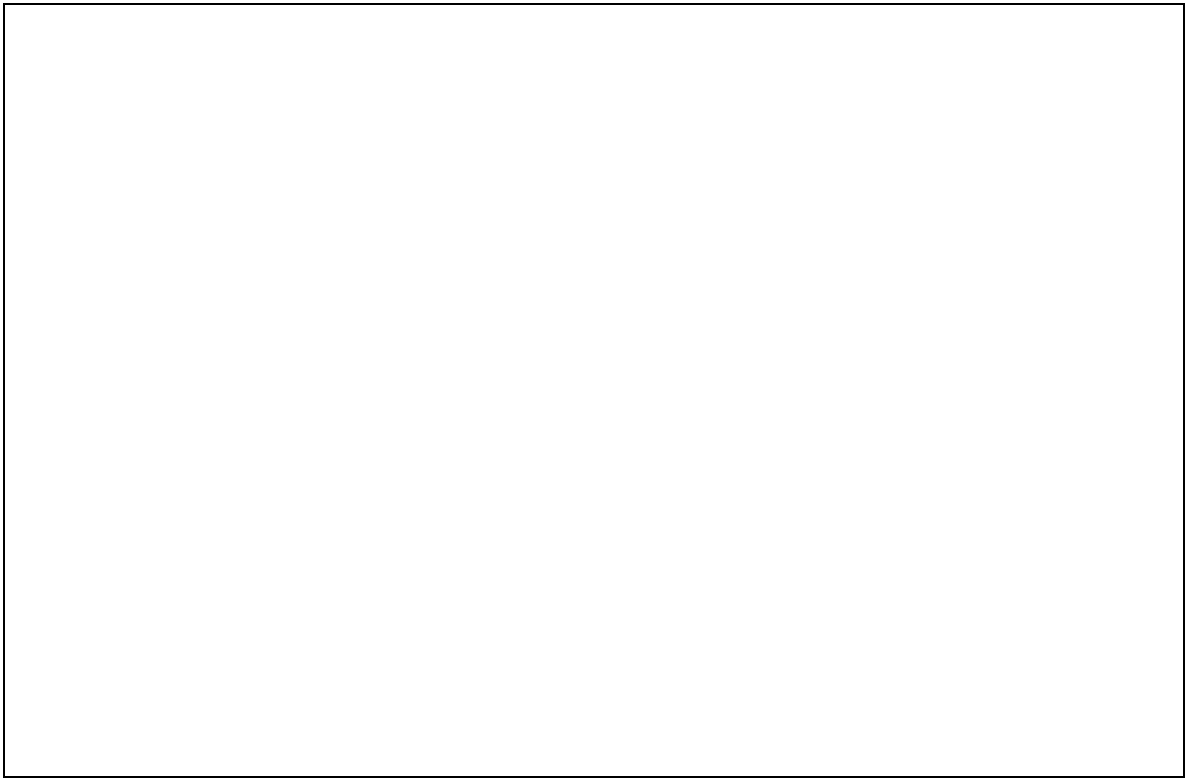
$$H(s) = \frac{G e^{-sL}}{1 + sT}, \quad L < T_s$$

Implement the Skogestad/Simple Internal Model Control (SIMC) [6, 7]:

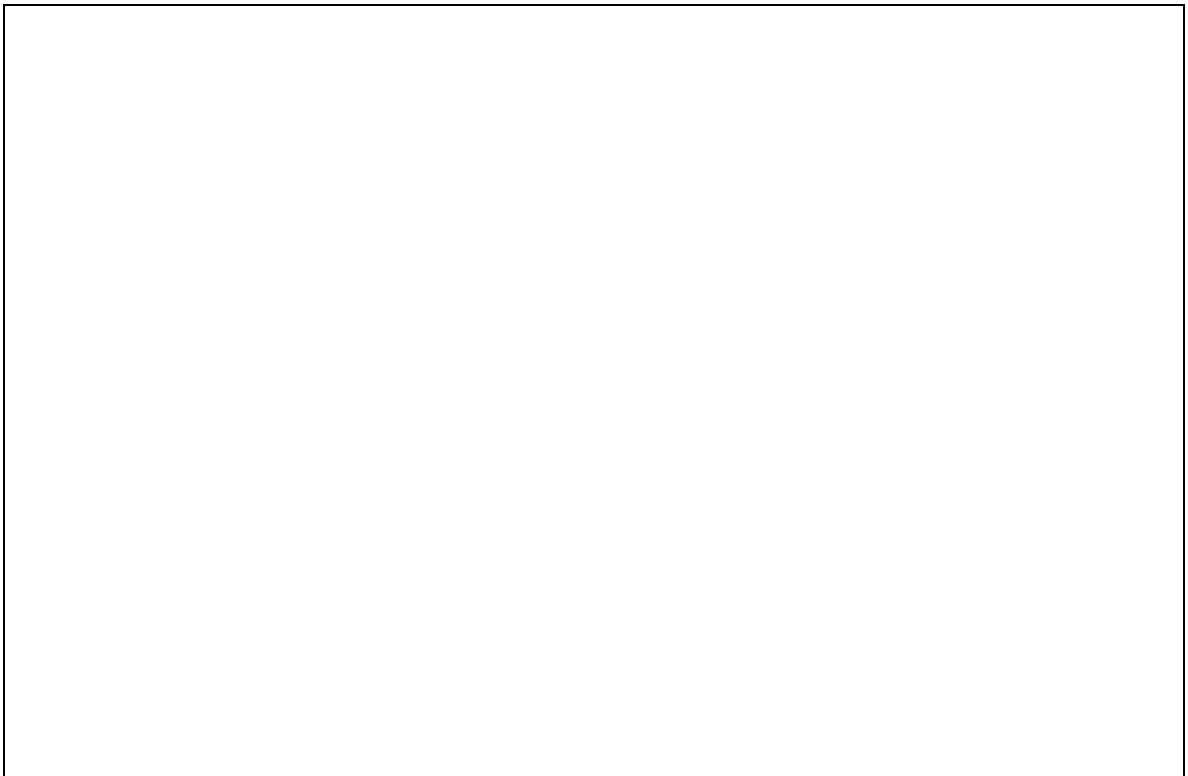
$$K = \frac{1}{G} \frac{T}{\tau_c + L}, \quad T_i = \min(T, 4(\tau_c + L))$$

where τ_c is the time constant of a “smooth” desired first-order closed-loop response.

a trade-off between output performance (small) vs. robust & input usage (large), guideline: $\tau_c = L$.



4. compare the three controllers and conclude;



5. pick your best controller and test the closed-loop response to an input disturbance.

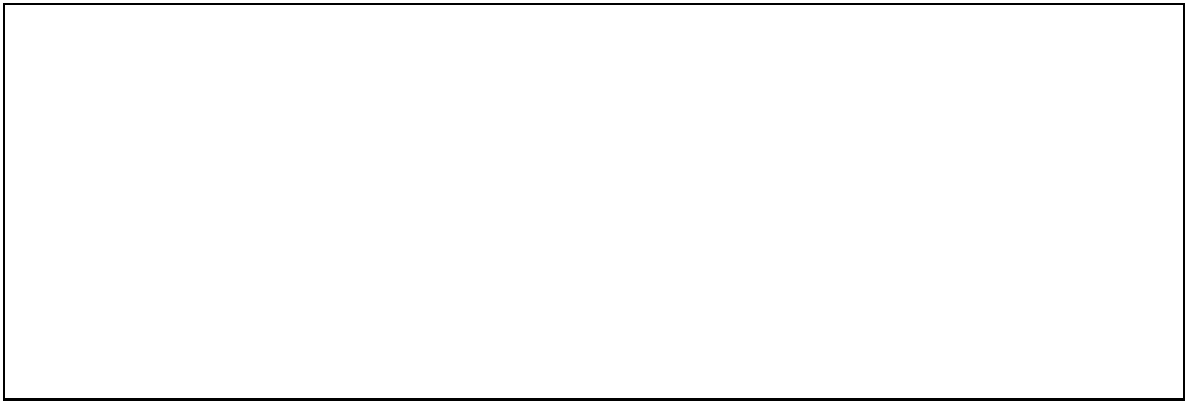


Fig. 5.4 Choice of a maximum duration of a pulse in a PRBS

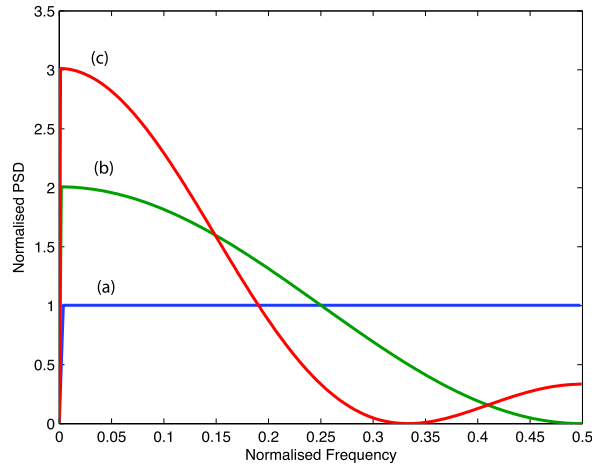
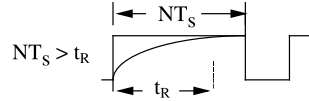


Fig. 5.5 Power spectral density of a PRBS sequence, (a) $N = 8$, $p = 1$, (b) $N = 8$, $p = 2$, (c) $N = 8$, $p = 3$

be greater than the rise time t_R of the plant. The maximum duration of a pulse being NT_s , the following condition results:

$$NT_s > t_R \quad (5.104)$$

which is illustrated in Fig. 5.4. From condition (5.104), one determines N and therefore the length of the sequence, which is $2^N - 1$. Furthermore, in order to cover the entire frequency spectrum generated by a particular PRBS, the length of a test must be at least equal to the length of the sequence. In a large number of cases, the duration of the test L is chosen equal to the length of the sequence. If the duration of the test is specified, it must therefore be ensured that:

$$(2^N - 1)T_s < L \quad (L = \text{test duration}) \quad (5.105)$$

Note that the condition (5.104) can result in fairly large values of N corresponding to sequence lengths of prohibitive duration, either because T_s is very large, or because the system to be identified may well evolve during the duration of the test. This is why, in a large number of practical situations, a submultiple of the sampling frequency is chosen as the clock frequency for the PRBS. If:

$$f_{PRBS} = \frac{f_s}{p}; \quad p = 1, 2, 3, \dots \quad (5.106)$$

then condition (5.104) becomes:

$$pNT_s > t_R \quad (5.107)$$

This approach is more interesting than the increase of the sequence length by increasing N in order to satisfy (5.104). Indeed, if one passes from N to $N' = N + 1$, the maximum duration of a pulse passes from NT_s to $(N + 1)T_s$ but the duration of the sequence is doubled $L' = 2L$. On the other hand, if $f_{PRBS} = f_s/2$ is chosen, the maximum duration of a pulse passes from NT_s to $2NT_s$ for a doubled sequence duration $L' = 2L$. From a comparison of the two approaches, it results that the second approach (frequency division) enables a pulse of greater duration to be obtained for an identical duration of the sequence and thus of the test. If p is the integer frequency divider, one has in the case of clock frequency division ($d_{\max} = \text{maximum pulse duration}$)

$$d_{\max} = pNT_s; \quad L' = pL; \quad p = 1, 2, 3, \dots$$

By increasing N by $(p - 1)$ and thus the length of the sequence (without changing the clock frequency), one obtains:

$$d_{\max} = (N + p - 1)T_s; \quad L' = 2^{(p-1)}L; \quad p = 1, 2, 3, \dots$$

Note that dividing the clock frequency of the PRBS will reduce the frequency range corresponding to a constant spectral density in the high frequencies while augmenting the spectral density in the low frequencies. In general, this will not affect the quality of identification, either because in many cases when this solution is considered, the plant to be identified has a low band pass or because the effect or the reduction of the signal/noise ratio at high frequencies can be compensated by the use of appropriate identification techniques. However, it is recommended to choose $p \leq 4$.

Figure 5.5 shows the normalized power spectral density (PSD) of PRBS sequences generated with $N = 8$ for $p = 1, 2, 3$. As one can see, the energy of the spectrum is reduced in the high frequencies and augmented in the lower frequencies. Furthermore, for $p = 2$ and $p = 3$ we have no excitation at f_s/p .

Up to now, we have been concerned only with the choice of the length and clock frequency of the PRBS; however, the magnitude of the PRBS must also be considered. Although the magnitude of the PRBS may be very low, it should lead to output variations larger than the residual noise level. If the signal/noise ratio is too low, the length of the test must be augmented in order to obtain a satisfactory parameter estimation. Note that in a large number of applications, the significant increase in the PRBS level may be undesirable in view of the nonlinear character of the plants to be identified (we are concerned with the identification of a linear model around an operating point).

5.8 Model Order Selection

In the absence of a clear a priori information upon the plant model structure, two approaches can be used to select the appropriate orders for the values of d, n_A, n_B characterizing the input-output plant model

Table 24. Pulse transfer functions for continuous-time systems with zero-order hold

$H(s)$	$H(z^{-1})$
$\frac{1}{s}$	$\frac{T_s z^{-1}}{1 - z^{-1}}$
$\frac{G}{1 + sT}$	$\frac{b_1 z^{-1}}{1 + a_1 z^{-1}} ; b_1 = G(1 - e^{-T_s/T}) ; a_1 = -e^{-T_s/T}$
$\frac{Ge^{-sL}}{1 + sT} ; L < T_s$	$\frac{b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1}} ; b_1 = G(1 - e^{(L - T_s)/T}) ;$ $b_2 = Ge^{-T_s/T} (e^{L/T} - 1) ; a_1 = -e^{-T_s/T}$
$\frac{\omega_0^2}{\omega_0^2 + 2\zeta\omega_0 s + s^2}$ $\omega = \omega_0 \sqrt{1 - \zeta^2}$ $\zeta < 1$	$\frac{b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$ $b_1 = 1 - \alpha \left(\beta + \frac{\zeta\omega_0}{\omega} \vartheta \right) ; b_2 = \alpha^2 + \alpha \left(\frac{\zeta\omega_0}{\omega} \vartheta - \beta \right)$ $a_1 = -2\alpha\beta ; a_2 = \alpha^2$ $\alpha = e^{-\zeta\omega_0 T_s} ; \beta = \cos(\omega T_s) ; \vartheta = \sin(\omega T_s)$

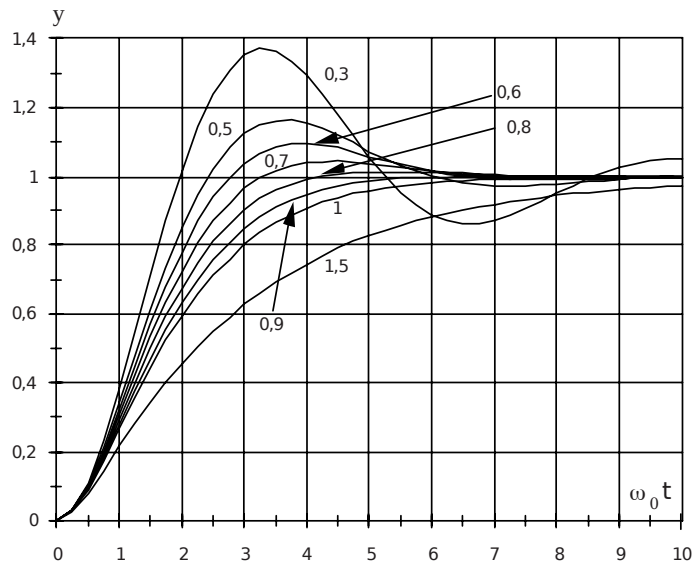


Figure 1.10. Normalized frequency responses of a second-order system to a step input

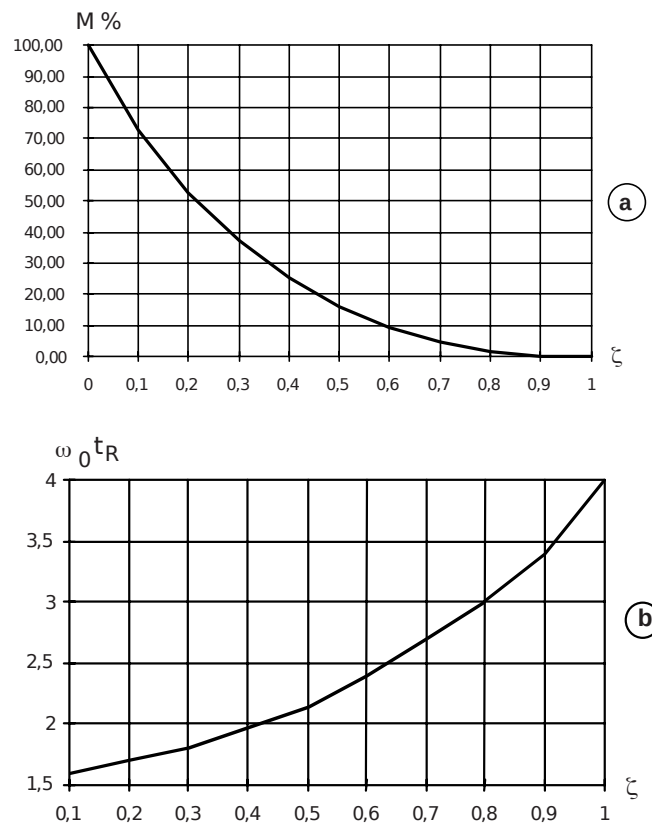


Figure 1.11. Second-order system: **a)** maximum overshoot M as a function of the damping factor ζ ; **b)** normalized rise time as a function of ζ

Chapter 5

Modeling and Estimation of an mBot Mobile Robot (2 labs)

Reference material: Lessons 1-4, Principles of Physical Modeling, Some Basic Relationships in Physics and Bond Graphs (lab 1) then Lesson 9 and 12 Parameter Estimation in Linear Models and Recursive Estimation Methods (lab 2)

This lab combines Arduino IDE, Matlab and an mBot robot as a platform for doing theoretical and practical work in dynamic system modeling, Kalman filter design and parametric identification, and optimal control. The modeling of the mBot robot is developed by the students as well as the design of a Kalman filter to estimate the angular and translational velocities. The student can then design an optimal tracking procedure to follow a desired angular velocity (e.g. using LQR).

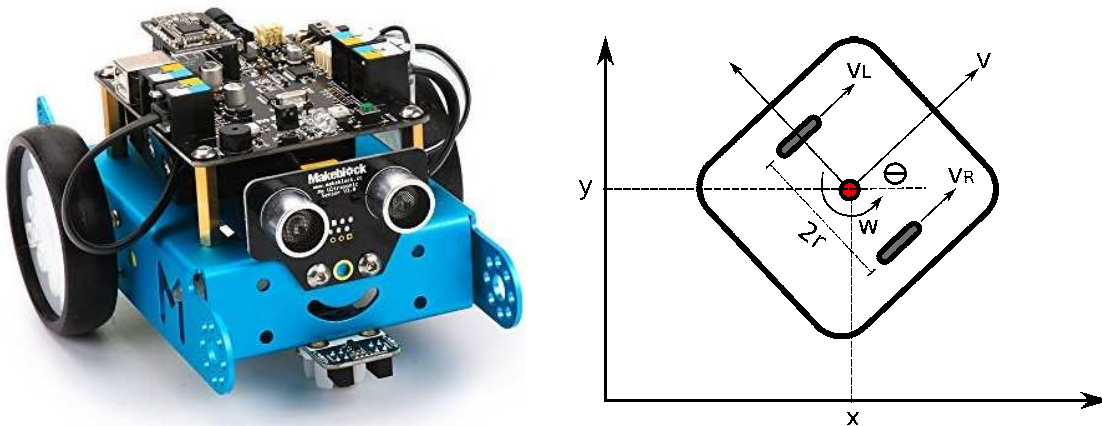


Figure 5.1: Picture and scheme of the mBot robot with three wheels.

5.1 System Description

The mBot robot (presented in Fig. 5.1) is an education robot which disposes of basic robotic machinery and electronic components ideal to teach the fundamentals of the control of mobile robots.

The mBot is equipped with the following set of sensors [8]:

- an Ultrasonic sensor, to detect distance and avoid obstacles;
- a Line follower sensor, which measures the reflection of an infrared light;
- a 3-Axis accelerometer and a gyro sensor, for motion and posture detection.

For our purpose, we are interested in particular in the accelerometer and the gyro sensor to control the trajectory of the robot. Mbot is controlled using the micro-controller ATmega328, which is commonly used in Arduino development platform and has a crystal of 16 MHz. The sensors are connected to the embedded microcontroller and provide information on the board motion to generate control inputs for the wheel motors according to a specific desired trajectory. Yet, it is commonly acknowledged that position control for a robot requires to implement encoders on each wheel of the robot as in [9] and [10]. Indeed using inertial sensors such as an accelerometer for the position control problem is difficult

due to unknown sensor bias, which leads to position-estimate divergence. In particular, the integration of angular velocity measurements from gyrosensors to obtain Euler angles as well as double integration of accelerometer measurements leads to linearly or quadratically increasing position errors.

5.2 Modeling and transfer functions

We are interested on the modeling of a three wheel robot idealized as the scheme depicted on the right side of Fig. 5.1. Two of the wheels are independently powered by electric motors and one is left free with the purpose of letting the robot turn. The way of controlling the robot position is to change the velocities of the motorized wheels. Note that:

- the translation of the robot is produced by the sum of the rotational speed of the two wheels;
- the rotation is produced by the difference between these speeds.

Using this information, you can decompose the impact of wheels velocities on rotational and translational energies, and find a dynamic input-output model for the robot. To do this, perform the following steps:

1. draw a first (complete) bond graph that models the robot dynamics, taking into account different friction coefficients for the wheels, the robot translational and rotational velocities, and friction effects on the front wheel;
2. draw a second, simplified bond graph with the following assumptions:
 - neglect the inertias of the wheels;
 - neglect the friction of the front wheel;
3. translate the simplified bond graph into equations, and the state-space model into transfer functions;
4. simulate the resulting model (with arbitrary values of the physical constants) using Matlab.

Note that the only available measurement is the rotational velocity from the gyro sensor.

5.3 Robot Programming

The aim of this lab is to get the main robot properties and get familiar with its programming and the model validation.

Consider the simplified model:

$$\frac{d}{dt} \begin{bmatrix} v \\ \dot{\theta} \end{bmatrix} = -\frac{1}{R^2} \begin{bmatrix} \frac{f_{w1}+f_{w2}}{m} & \frac{r}{m}(f_{w1}-f_{w2}) \\ \frac{r}{J}(f_{w1}-f_{w2}) & \frac{r}{J}(f_{w1}+f_{w2}) \end{bmatrix} \begin{bmatrix} v \\ \dot{\theta} \end{bmatrix} + \frac{1}{R} \begin{bmatrix} \frac{1}{m} & \frac{1}{m} \\ \frac{r}{J} & -\frac{r}{J} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad (5.1)$$

where $R = 3.2\text{ cm}$ is the radius of the wheels, $r = 5.6\text{ cm}$ the distance from the cart centrum to the middle of the wheel as depicted in Fig. 5.1 (right), J is the momentum of inertia of the cart (suppose that it is rotating around its center of mass, e.g. see https://en.wikipedia.org/wiki/List_of_moments_of_inertia, has a length of 17 cm and a width of 9 cm), $m = 400\text{ g}$ is the mass of the cart and $f_{w1,w2}$ are the frictions of each wheel (e.g. see https://en.wikipedia.org/wiki/Rolling_resistance).

Your tasks are as follows:

1. considering the steady-state behavior design a method that allows you to evaluate the frictions of each wheel from input-output data;
2. create a Matlab function that loads the data set `steps1000_entrance.csv` from the restricted area of MiSCIT website (modeling class) and creates the state-space representation of your system (after evaluating the frictions);
3. discretize the dynamics and compare your model with the reference data set;
4. run experiments on your own mBot (programmed on Arduino IDE) with an appropriate choice of inputs to get its friction coefficients (not necessarily the same those obtained from the reference data set, give these coefficients in your report);
5. create a set of data files (.csv format) to record those experiments;
6. compare the experimental results with your model simulations and conclude on the validity of our simplified model.

To run simple experiments on the mBot:

- download `mBot.zip` from the restricted area of MiSCIT website (modeling class);

- start Arduino IDE (switch to English if necessary: Fichier/Preferences/Choix de la langue/English and then restart) and open mbot_IO.ino;
- configure the board from the Tools menu by selecting: Board: "Arduino/Genuino Uno" and Port: COM X where X is the second port available (typically ≥ 10).
- compile and upload the code;
- view the results from the Serial Monitor or Serial Plotter

5.4 Parameter estimation

In this lab, you wish to estimate the parameters of a simplified model where you suppose that $f_{w1} = f_{w2}$. With this hypothesis:

1. rewrite (5.1) as a decoupled model, discuss the case when measuring $\dot{\theta}$ is sufficient to identify a parameterized model and acquire the appropriate data;
2. using explicit time discretization, express the model in a regressor form and solve the least-squares estimation problem with your data set;
3. evaluate your model with the validation tools seen in class;
4. write a recursive identification algorithm and implement it on the Arduino.

5.5 Conclusions

Conclude and comment on the different results obtained in this lab.

Bibliography

- [1] Ioan Doré Landau, Rogelio Lozano, Mohammed M'Saad, and Alireza Karimi. *Adaptive Control: Algorithms, Analysis and Applications*. Communications and Control Engineering. Springer, 2011.
- [2] I.D. Landau, M. Alma, J. Martinez-Molina, G. Buche, and A. Karimi. Benchmark on adaptive regulation: Rejection of unknown/time-varying multiple narrow band disturbances. http://www.gipsa-lab.grenoble-inp.fr/~ioandore.landau/benchmark_adaptive_regulation/, October 2009.
- [3] O. Hinton. Digital signal processing, ch.6 - describing random sequences. <http://www.staff.ncl.ac.uk/oliver.hinton/eee305/Chapter6.pdf>, October 2009.
- [4] Sandra Hamze. *Model-based System Engineering for Powertrain Systems Optimization*. Theses, Université Grenoble Alpes, November 2019.
- [5] Ioan D. Landau and Gianluca Zito. *Digital Control Systems: Design, Identification and Implementation (Communications and Control Engineering)*. Springer-Verlag, Berlin, Heidelberg, 2006.
- [6] Sigurd Skogestad and Ian Postlethwaite. *Multivariable feedback control: Analysis and Design*. John Wiley, Hoboken, US-NJ, 2005.
- [7] Skogestad S. and Grimholt C. *PID Control in the Third Millennium*, chapter The SIMC Method for Smooth PID Controller Tuning. Advances in Industrial Control, Springer. Vilanova R., Visioli A., London, 2012.
- [8] mBot datasheet. <https://www.makeblock.com/steam-kits/mbot>, 2019.
- [9] C. Kuo, H. Chou, F. L. Chernousko, V. G. Gradetsky, N. Bolotnik, Y. Aniroh, and E. Setijadi. Trajectory tracking of a wheeled wall climbing robot using pid controller. In *2015 International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA)*, pages 143–146, 2015.
- [10] A. De Luca, G. Oriolo, and M. Vendittelli. *Control of Wheeled Mobile Robots: An Experimental Overview*, pages 181–226. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.