Lab 2: Identification of a Dynamical System

1 Objectives

In this lab, you will learn how to identify a second-order LTI system from it's time and frequency responses. As an example, we will use a MATLAB class that simulates a DC motor. We will treat this as a black-box system (i.e., a system with inputs and outputs but without any knowledge of its internal working). The final objective is to identify the model parameters of the DC motor using the properties of its time response.

Intermediate objectives are:

- Understand the mathematical representation of a DC motor.
- Understand how to model a continuous time system in discrete time using MATLAB.
- Identify the transfer function of a dynamical system in time domain.
- Identify the transfer function of a dynamical system in frequency domain.

2 Introduction

2.1 Model of a DC motor

A DC motor is a device that converts armature current into mechanical torque. A schematic of a DC motor is shown in Figure 1.

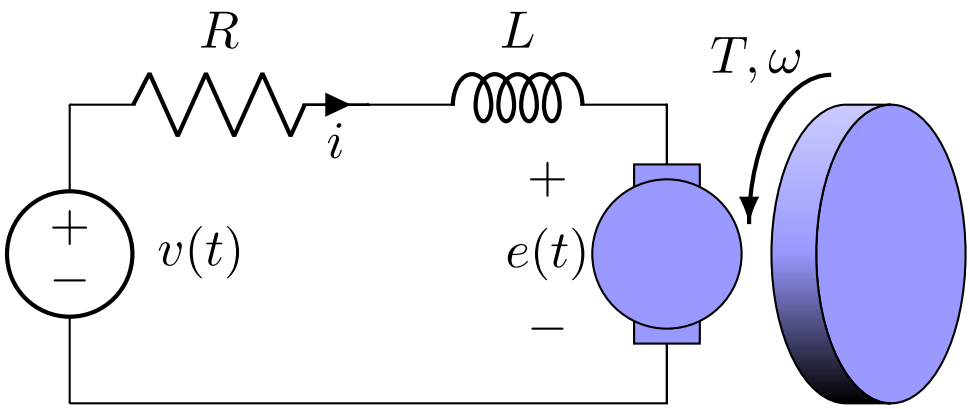

Figure 1 DC motor

Let i denote the armature current and T denote the mechanical torque. The motor torque T is proportional to the armature current i. In particular,

$$T = K_t i$$

where K_t is a proportionality constant. The rotation of the motor shaft produces a back electromotive force e which is proportional to the angular velocity ω of the shaft. In particular,

$$e = K_e \omega$$

where K_e is a proportionality constant.

Let v denote the voltage applied to the motor's armature, R denote the electric resistance, L denote the electric inductance, J denote the moment of inertia of the rotor, and b denote the motor's viscous friction constant.

Then, according to Newton's second law and Kirchoff's voltage law, we get

$$J\frac{d\omega}{dt} + b\omega = K_t i$$

$$L\frac{di}{dt} + Ri = v - K_e\omega$$

Taking the Laplace transform of the above equations, we get

$$(Js + b)\Omega(s) = K_t I(s)$$

$$(Ls + R)I(s) = V(s) - K_e\Omega(s)$$

Eliminating $I(s)$ from the two equations, we get

$$\frac{(Js+b)\Omega(s)}{K_t} = \frac{V(s) - K_e\Omega(s)}{(Ls+R)} \implies (Js+b)(Ls+R)\Omega(s) = K_t V(s) - K_t K_e \Omega(s)$$

Thus, the transfer function of the DC motor is

$$H(s) = \frac{\Omega(s)}{V(s)} = \frac{K_t}{(Js+b)(Ls+R) + K_t K_e}$$

We can write this in the canonical form as follows

$$H(s) = K\frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

where the gain $K = \frac{K_t}{bR + K_t K_e}$, the natural frequency $\omega_n = \sqrt{\frac{bR + K_t K_e}{JL}}$, and a damping coefficient $\zeta = \frac{1}{2}\frac{bL + RJ}{\sqrt{JL\,(bR + K_t K_e)}}$.

2.2 Simulated model of a DC motor

We provide a MATLAB class simulating a DC Motor based on the model presented above. The following explains how to get the necessary MATLAB files:

1. Log in to `myCourses` and select the content folder Labs.
2. Download `lab_02_matlab.zip`.
3. Unzip the content in your MATLAB workspace.
4. Open MATLAB and navigate to the path where you have `Lab_02_SystemIdentification_Simulation.m`.
5. Run `Lab_02_SystemIdentification_Simulation.m`. You shouldn't have any errors.

The simulation of the DC Motor can be found in `Interface/SimDCMotor.m`. For the purpose of this laboratory, we will treat `SimDCMotor.m` as a blackbox. There is no need to understand its implementation to do the laboratory experiments.

The script file `Lab_02_SystemIdentification_Simulation.m` contains commented code to guide you through the lab. To see the detailed documentation of `SimDCMotor`, use:

```
doc SimDCMotor       % General documentation about SimDCMotor
```

3 Identification by Step Response Analysis

Recall that a dynamical system whose input-output behavior is described by a constant coefficient linear differential equation is always an LTI (linear and time-invariant) system. If the input of a continuous-time system is taken as a step signal, the corresponding output is called the step response. The first method for system identification is to analyse the step response of a dyanmical system. First we will recall key properties of the step response of second order systems, then we will analyse the step response of `SimDCMotor.m`.

3.1 Second order systems

A generic second-order system is described by a second-order linear differential equation

$$\frac{d^2y(t)}{dt^2} + b_1\frac{dy(t)}{dt} + b_0 y(t) = a_0 u(t)$$

The transfer function of this system is given by

$$H(s) = \frac{Y(s)}{U(s)} = \frac{a_0}{s^2 + b_1 s + b_0} \tag{1}$$

This transfer function be written in the canonical form as

$$H(s) = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

where

- K is the `DC gain`, also called `steady state gain`. The `DC gain` represent the amplitude ratio between the steady state step response and the step input.
- ζ is the `damping ratio`.
- ω_n is the `natural frequency`.

Figure 2 demonstrates the unit step response of an underdamped ($\zeta < 1$) second-order system.

In the time domain, the form of the step response $s(t)$ of a second-order system depends on the damping ratio. For an underdamped or undamped ($0 \leqslant \zeta < 1$) second-order system, it has the following form:

$$s(t) = K\left(1 - \frac{e^{-\zeta\omega_n t}}{\sqrt{1-\zeta^2}}\sin\left(\sqrt{1-\zeta^2}\omega_n t + \arccos(\zeta)\right)\right)\mathbb{1}(t)$$

Moreover, it is possible to define an underdamped or undamped ($0 \leqslant \zeta < 1$) second-order system by the following requirements:

- **Rise time (t_r)**: is the time it takes the system to reach to rise from 10 % to 90 % of its final value. This value could be related to the natural frequency by

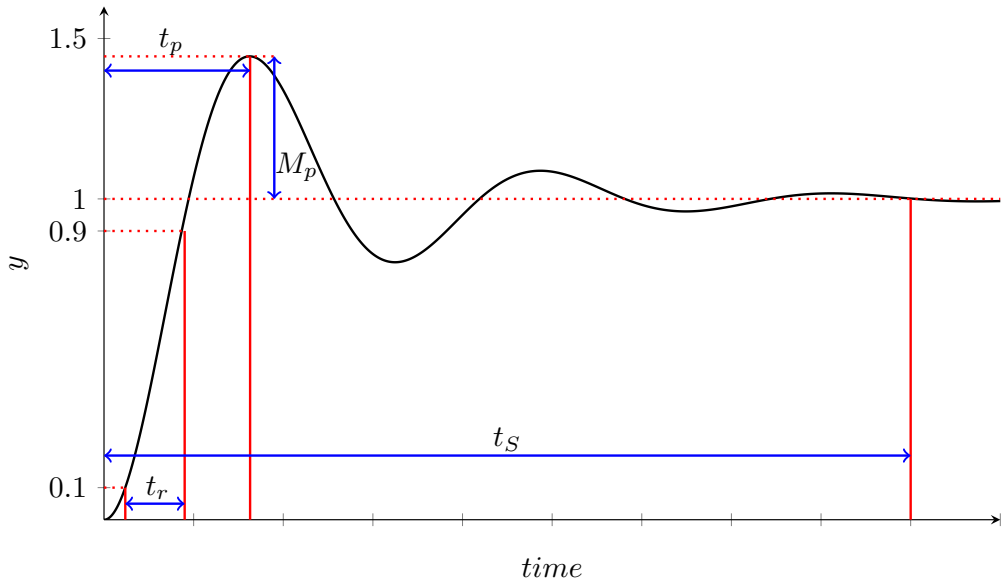

Figure 2 Step response of a second order system

$$t_r \approx \frac{1.8}{\omega_n}$$

- **Peak time (t_p)**: is the time until the response hits its maximum overshoot[1].

$$t_p = \frac{\pi}{\omega_d} = \frac{\pi}{\omega_n \sqrt{1-\zeta^2}}$$

- **Overshoot (M_p)**: is the maximum peak value of the system response curve measured from the final steady-state value.

$$M_p = e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}}$$

- $\Delta\%$ **Settling time (t_s)**: is the time required for the response curve toreach and stay within a $\pm\Delta\%$ range around the steady-state value.

$$t_s \approx -\frac{\log(\Delta\%)}{\zeta\omega_n}$$

3.2 Question 1

Consider a general second-order system with $K = 1$ and $\omega_n = \pi$ rad/s. Use MATLAB to plot the step response for $\zeta \in \{0, 0.5, 1, 1.5\}$. In each case, compute the rise time, maximum overshoot, and 5% settling time from the plots.

3.3 Step response of `SimDCMotor`

In the following, instructions of how to analyse the step response of `SimDCMotor` is detailed.

[1] $\omega_d = \omega_n \sqrt{1-\zeta^2}$ is the damped frequency.

1. In order to use the `SimDCMotor` object, we initialize it by specifying the values of resistance `R` and moment of inertia `J`. Other model parameters take a default value. The code snippet below shows how to initialize `SimDCMotor`.

```
% Set parameters for DC motor
Parameters.Resistance = 1.5;  % Terminal resistance (Ohm)
Parameters.Inertia    = 5e-4; % Rotor inertia (kg m^2)

% Create an object handle for SimDCMotor with specified parameters
Motor = SimDCMotor(Parameters);
```

   The `Motor` parameters can be seen by typing:

```
Motor.Parameters
```

2. The input interface to a `SimDCMotor` object is the function `drive()`. The `drive()` function takes 3 arguments, the voltage applied to the motor's armature in volts, the time when this voltage is applied in seconds, and the duration of the drive function in seconds.

   For instance,

```
Motor.reset();  % reset the motor internal variable (including time)
% Drive Motor with a voltage (2 V) for a duration (5 sec) starting from delay (1 sec)
Motor.drive(2,1,5)
```

   will drive the DC Motor with a $2V$ input voltage at time equals 1 second during 5 seconds.

3. The outputs of a `SimDCMotor` object are 4 values in a form of arrays:
   - Use `Motor.time()` to retrieve the time history.
   - Use `Motor.current()`to retrieve the history of the armature current `i` in amper.
   - Use `Motor.velocity()` to retrieve the history of the DC motor angular velocity in radian per second.
   - Use `Motor.angle()` to retrieve the history of the DC motor angle in radian.

3.4 Question 2

1. Using MATLAB script `Lab_02_SystemIdentification_Simulation.m` simulate a step response of value 1 V for 5 seconds to a `SimDCMotor` object with the following parameter:
   - Resistance: 1.5 Ω.
   - Inertia: 5×10^{-4} kgm^2.
2. Using MATLAB, plot in the same graph both the voltage input $u(t)$ and the angular speed of the `Motor` $y(t)$.

3. From the plot, identify the parameter of the transfer function of this `Motor`: DC gain, natural frequency, and damping ratio. Observe that this is a step response of a second order system.
4. Using the DC motor mathematical model and given the proportionality constants $K_t = K_e = 0.05$, provide an approximation of the motor viscous friction constant b.

4 Identification by Frequency Analysis

4.1 Frequency analysis of a linear system

If a cosine wave $u(t) = A_u \cos(\omega t)$ is injected into a linear system at a given frequency, the system will respond at that same frequency ω with a certain magnitude A_u and a certain phase angle relative to the input ϕ. The steady-state system output can be written as $y(t) = A_y \cos(\omega t + \phi)$. A system Bode plot represents the magnitude and phase shift between the input and output (from the system) at each frequency.

For a given system, it is possible to draw a "point-by-point" Bode plot by injecting a cosine wave with a fixed frequency and measuring the magnitude and phase shift of the output after it reaches its steady-state.

4.1.1 Lissajous method for phase shift

The Lissajous (ellipse) method dates from the the analog oscilloscope time. Most oscilloscopes have a "XY mode" where it is possible to plot one signal in function of another. Using this view with sinusoidal signals, it is possible to accurately measure the phase shift between the two signals.

Let's assume an input signal $u(t) = A_u \cos(\omega t)$ and its output signal $y(t) = A_y \cos(\omega t + \phi)$. By eliminating time between those signal:

$$\frac{y}{A_y} = \frac{u}{A_u}\cos(\phi) \pm \sqrt{1 - \left(\frac{u}{A_u}\right)^2}\sin(\phi)$$

The shift phase ϕ satisfies:

$$\cos(\phi) = \frac{y_{u=A_u}}{A_y}$$

where $y_{u=A_u}$ is the value of y when u passes its maximum. The fraction $\frac{y_{u=A_u}}{A_y}$ can be measured empirically in the XY plot of $y(t)$ in function of $u(t)$.

4.2 Frequency analysis of `SimDCMotor`

This time we need to send a cosine wave input signal to `SimDCMotor`. In the following we provide an example based on the MATLAB script `Lab_02_SystemIdentification_Simulation.m`:

1. `SimDCMotor` implements a discrete time model with a chosen sampling time. In order to drive a `SimDCMotor` object we can generate a discrete signal for the input, then send it following a fixed time interval. One can generate a discrete time vector by the following:

```
dt = 0.1;        % Sampling time  for input signal (in seconds)
T  = 5;          % Total duration of simulation (in seconds)
time  = 0:dt:T;  % A vector containing all time samples
```

2. Using a `for` loop, we generate the input signal $u(t) = \cos(\omega t)$ and drive the `Motor` for one time sample.

```
Omega = pi; % input signal frequency (rad/s)
for t = time
   % Generate a cosine wave input at current time
   u = cos(Omega*t);

   % Drive motor for a duration of dt at time t
   Motor.drive(u, t, dt);
 end
```

   Note that the input signal sampling time should be small enough to generate a smooth signal.

3. Apart from the input signal sampling time, the `SimDCMotor` also pocesses an internal sampling time which dictates when to process a new voltage input and when to return a new measurement. The default value of this sampling time can be seen along with other `Motor` options by typing:

```
Motor
```

   A new internal sampling time can be set by typing:

```
Motor.setSamplingTime(0.01) % Set a new sampling time for SimDCMotor
```

   Note that even if you decide to drive the `Motor` with a new voltage faster than its internal sampling time, the motor will only processes the last voltage recieved at the end of its internal sampling time interval.

4.3 Question 3

1. Using MATLAB, simulate a cosine wave input to `simDCMotor` of magnitude 1 and phase 0 ($u(t) = \cos(2\pi f t)$), and find the magnitude and phase shift for different frequencies:

Frequency (Hz)	A_y	$y_{u=1}$	Magnitude (db)	Phase (deg)
0.01				
0.1				
0.2				
0.4				
0.5				
0.7				
1.0				
2.5				
5.0				

You may need to adjust the duration of simulation and the sampling time depending of the chosen frequency.

2. Draw the resuling Bode plot (Magnitude (db) vs Frequency (Hz) on a semilog plot and Phase (deg) vs Frequency (Hz) on a semilog plot).
3. From the Bode plot, what is the system order?
4. From the Bode plot, measure the DC gain.
5. From the Bode plot, measure the cut-off frequency (Frequency at which the magnitude drops by a factor of $-20\log_{10}(\frac{1}{\sqrt{2}}) = -3$(db) from the DC gain).

5 Assignement

In a report format, answer the laboratory questions. The report should contain:

- An introduction and a conclusion, outlining the purpose of the laboratory and what you have learned.
- Explanation of the steps to answer the laboratory questions.
- All figures should have a legend and a caption.
- Include your code in the report appendix.