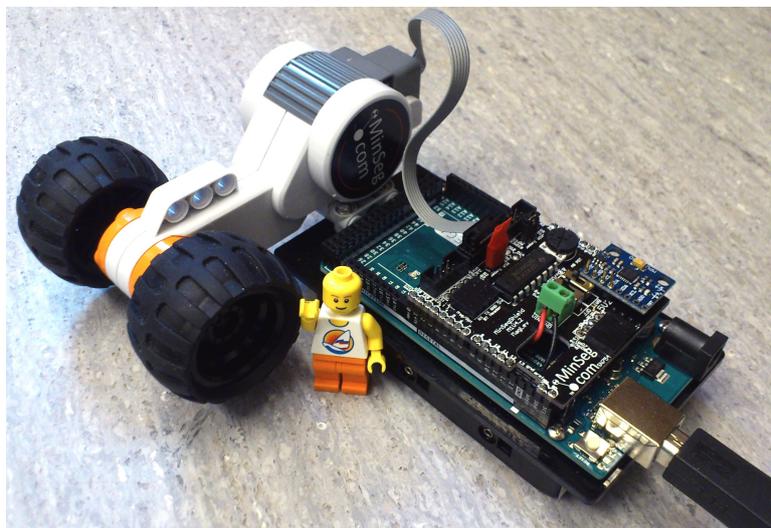


# DC-motor modelling and parameter identification

This version: November 1, 2017



Name: \_\_\_\_\_

P-number: \_\_\_\_\_

Date: \_\_\_\_\_

Passed: \_\_\_\_\_



# Chapter 1

## Introduction

The purpose of this lab is to give an introduction to a simple dynamical system, models, signals, and introduce the MinSeg robot and associated software. By performing experiments on a small DC-motor, some of its physical parameters will be identified (also called estimated), and a first-order differential equation describing the dynamical behavior of the motor will be developed. As an alternative approach, a model is also developed using a simple step-response experiment.

DC-motors are a central part of many products and mechatronic designs, and having a model of a motor is important for dimensioning, simulations, development of controllers, performance analysis etc.



**Figure 1.1:** Robotic hand using 15 DC-motors to position 5 fingers independently. Having good models of the motors is crucial when developing a high-precision control system for the hand.

## 1.1 Hardware set-up

The lab is based on three main hardware components.

To begin with, we have a standard desktop computer. This computer is used to automatically develop and deploy code using MATLAB and SIMULINK models.

To supply power to the DC-motor and perform measurements of motor angles, we use a board with an Arduino micro-controller which runs the auto-generated code. It also communicates with the desktop computer and thus allows us to look at the measurements.

The motor we experiment with is a simple DC-motor with two wheels attached. The motor is normally part of a LEGO Mindstorms kit.

The Arduino together with the motor and wheels is called the MinSeg.

## 1.2 Trouble shooting

**The wheels turn slowly and/or erratically** Make sure the tires do not rub against the motor. You can pull the wheels apart as they slide on the wheel axis.

**Complaints about COM port or connection when downloading to board** Try again. If it still fails, disconnect USB-cable and connect it again.

**Complaints about OUT OF MEMORY** Save your model and restart MATLAB.

**The motor does not run** Have you reinstalled the motor jumper after current measurements?

# Chapter 2

## Preparation

The questions below, and all questions throughout the document marked as **Preparation** must be done before attending the lab. Note that there are additional preparation exercises in Chapter 3.

Solutions to all questions should be available upon request from the lab assistant, and the preparation exercises in Chapter 3 are preferably written in this printed documented.

The scheduled time spent with the laboratory equipment is only a small part of the complete lab, as a major part is spent on the theoretical material during preparations. When the lab starts, it is assumed you have done all preparations, and have a clear idea of the tasks that will be performed during the lab.

**Preparation 1** *Read Section 2.1 and 2.6 in the course book by Ljung & Glad.*

**Preparation 2** *Verify (by differentiation, not by solving it from scratch!) that  $y(t) = K(1 - e^{-t/T})c$  is the solution to the first-order differential equation  $T\dot{y}(t) = -y(t) + Ku(t)$  for an input switching from  $u(t) = 0$  to  $u(t) = c$  at  $t = 0$  (i.e., a step-response of amplitude  $c$ ), with initial condition  $y(0) = 0$ .*

**Preparation 3** *The constant  $K$  in the form above is called the **static gain** of the system and describes how much the system amplifies constant inputs in **steady-state** (i.e., when derivatives are 0 and  $y(t)$  has converged to its stationary value). Given the differential equation with input  $u(t) = c$  and solution as above, show that the output  $y(t)$  converges to  $Kc$  when  $t \rightarrow \infty$ . Do this*

using two different strategies. One approach where you simply analyze the solution given above, and another approach where you study the differential equation and exploit the fact that you know the value of  $\dot{y}(t)$  in steady-state.

**Preparation 4** The constant  $T$  in the first-order differential equation in the above form is called the **time-constant** and describes how fast the system reacts to changes on the input. Given  $y(t) = K(1 - e^{-t/T})c$ , what value will  $y(t)$  have when  $t = T$ ?

**Preparation 5** The time-constant is more generally defined as the time it takes before the output reaches a certain percentage of its final value, when the input has made a step change. To be consistent with the definition in the first-order system above, what percentage is this?

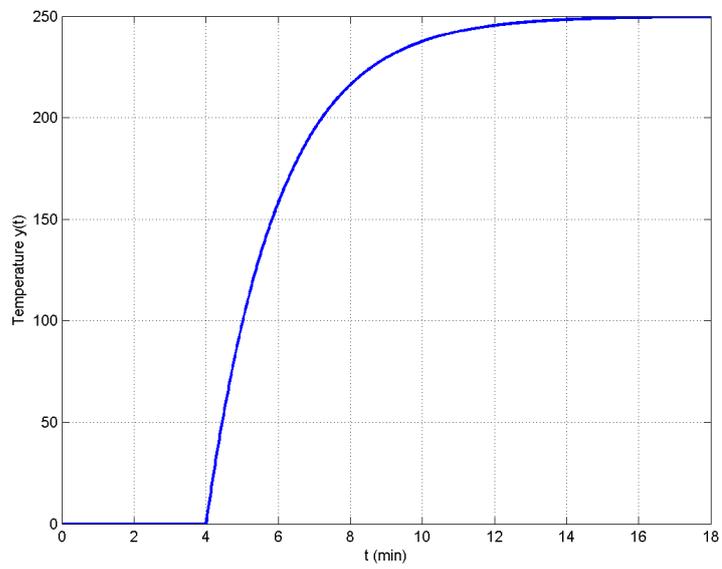
**Preparation 6** Sketch the function  $y(t) = K(1 - e^{-t/T})c$  for  $K = 3$ ,  $T = 10$  and  $c = 2$ . Particularly specify the value attained when  $t = 10$ .

**Preparation 7** An oven placed in a room with temperature  $0^\circ$  is reasonably well described by the differential equation  $T\dot{y}(t) = -y(t) + Ku(t)$  where  $y(t)$  is the oven temperature and  $u(t)$  is the supplied power. In an experiment performed to identify  $T$  and  $K$ , the oven was turned on with  $u(t) = 1000W$  at  $t = 4$  (not at  $t = 0!$ ), and the oven temperature was measured. Based on the results seen in Figure 2.1, what is the time-constant  $T$  and static gain  $K$  of the oven? **Hint: What percentage of the final temperature should have been reached  $T$  minutes after the oven was turned on)**

**Preparation 8** Write the differential equation  $a\dot{y}(t) + by(t) = cu(t)$  in the form  $T\dot{y}(t) = -y(t) + Ku(t)$ , i.e., express the time-constant  $T$  and gain  $K$  in terms of  $a$ ,  $b$ , and  $c$ .

**Preparation 9** Read the complete lab-pm. There are some theoretical questions in the pm which you are supposed to complete as preparation.

**Preparation 10** Print this document. You must bring a physical copy to the lab.



**Figure 2.1:** Step-response experiment of an oven. At  $t = 4$  the input is changed from 0 to 1000, and the oven temperature is recorded.

# Chapter 3

## The lab

As explained above, the lab will primarily consist of experimentation and data collection, using theoretical results and strategies derived during your preparation.

Items labeled **Preparation** are questions you are supposed to solve and fill out before attending the lab.

Items labeled **Task** are performed when attending the lab and you have access to the hardware.

### 3.1 DC-motor modeling

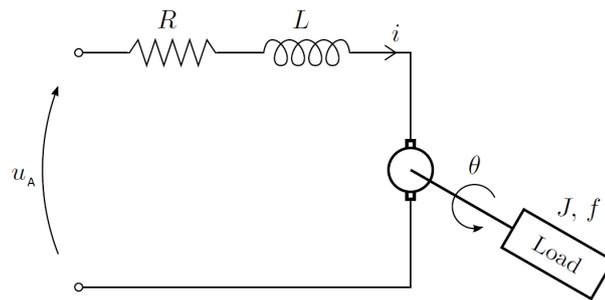
Consider a standard DC-motor model as depicted in Figure 3.1.

The electrical and mechanical differential equations governing the dynamics are

$$u_A(t) = L \frac{di(t)}{dt} + Ri(t) + k_v \omega(t) \quad (3.1)$$

$$k_a i(t) = J \dot{\omega}(t) + f \omega(t) \quad (3.2)$$

where  $u_A(t)$  is the voltage applied to the motor,  $i(t)$  is the current in the motor, and  $\omega(t) = \dot{\theta}(t)$  is the angular velocity of the motor shaft. The product  $k_a i(t)$  is the torque generated by the motor, and  $f \omega(t)$  is the velocity dependent friction in the motor. The parameters in the model are



**Figure 3.1:** DC-motor. When a voltage  $u_A(t)$  is applied on the motor, a current  $i(t)$  develops and accelerates the motor. As the angular velocity  $\omega(t) = \dot{\theta}(t)$  increases, the friction  $f\omega(t)$  in motor and drive shafts, and back-emf (electromotiv force)  $k_v\omega(t)$ , reduces the acceleration  $\dot{\omega}(t) = \ddot{\theta}(t)$ .

- L: Motor inductance, assumed to be 0.
- R: Motor armature resistance, unknown and will be measured directly.
- $k_v$ : Back-emf constant. Unknown and will be identified through steady-state analysis.
- $k_a$ : Torque constant. Unknown and will be identified through steady-state analysis. Related to  $k_v$ .
- $f$ : Friction coefficient. Unknown and will be identified through steady-state analysis.
- J: Motor inertia. Unknown and will be identified through dynamic analysis.

For a perfect motor with no energy losses, it holds that  $k_a = k_v$ . However, in practice this does not hold, and we know that

$$k_a \approx 0.65k_v. \quad (3.3)$$

Our task in this lab is to perform experiments to identify R,  $k_v$  (and thus  $k_a$ ),  $f$  and J, to obtain a dynamical model which relates the applied voltage  $u_A(t)$  to the angular velocity  $\omega(t)$ . This model will be used in later labs.

### 3.1.1 Rotary encoder

The motor is equipped with a simple encoder<sup>1</sup> to measure rotation of the motor shaft. The encoder uses  $n = 12$  holes with a so called quadrature design, meaning it has a resolution of 48 pulse changes per revolution. The motor shaft is connected via a gear with transmission ratio 15, meaning the encoder will deliver  $48 \cdot 15 = 720$  pulse changes per motor shaft revolution, leading to a maximal theoretical accuracy of  $0.5^\circ$ .



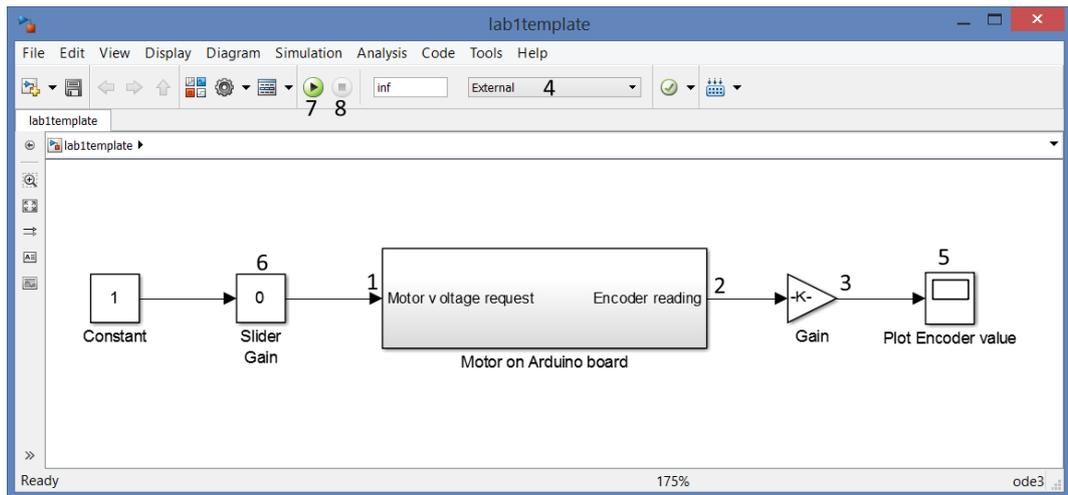
**Figure 3.2:** Encoder in motor

**Preparation 11** *One full revolution of a wheel will give us a measurement from the encoder of 720. However, we wish to work in standard units of radians, and would thus like to obtain the value  $2\pi$ . With which scaling gain should the measurement be multiplied with to accomplish this?*

Open the folder minseg on the Desktop, and start MATLAB and SIMULINK by double-clicking the SIMULINK model dcmotor/template1. This SIMULINK scheme graphically describes the code that will be generated and downloaded to the Arduino micro-controller.

Some important features are marked in Figure 3.3

<sup>1</sup>A small light-source shines at the right-most cog in the figure, and by detecting when the light is blocked or let through the holes, rotation is detected

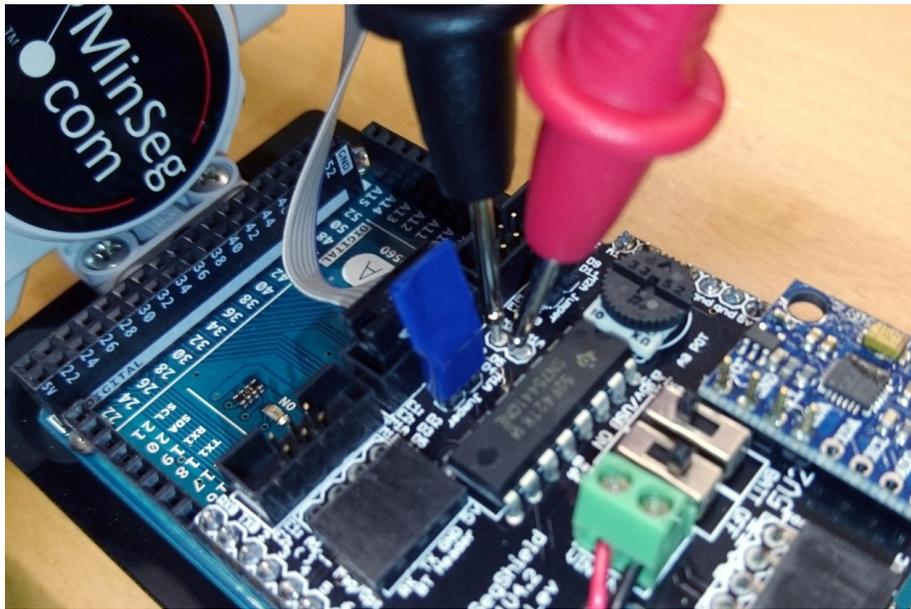


**Figure 3.3:** Template SIMULINK model

1. The value here will be sent by the Arduino micro-controller to the motor driver, and is our control signal  $u(t)$ . Due to voltage losses internally on the board in the motor driver chip, the voltage  $u_A(t)$  which actually is applied to the DC-motor is lower than  $u(t)$ .
2. The Arduino micro-controller counts pulses on the encoder, and this value can be used for computing the angle of the motor.
3. We convert the number of pulses counted to rotation angle  $\theta(t)$  in radians. This is done in the Gain block.
4. The SIMULINK model is specified to run in *External model*. This allows the Arduino to communicate with MATLAB continuously.
5. When running the model in external mode, the Arduino micro-controller can send data to MATLAB and SIMULINK. Here, we send the scaled encoder measurements to a plot scope to display them in real-time.
6. In external mode, we can also send information from MATLAB to the Arduino micro-controller. We will use a slider gain to change the requested motor voltage  $u(t)$  while the code is running. We will be able to multiply the constant 1 with values between 0 and 4.5, i.e., requesting up to 4.5V.
7. We compile, download and start the code on the Arduino micro-controller by pressing the green run button.
8. We stop the code by pressing the stop button. Must be one before editing the model

### 3.1.2 Identification of armature resistance

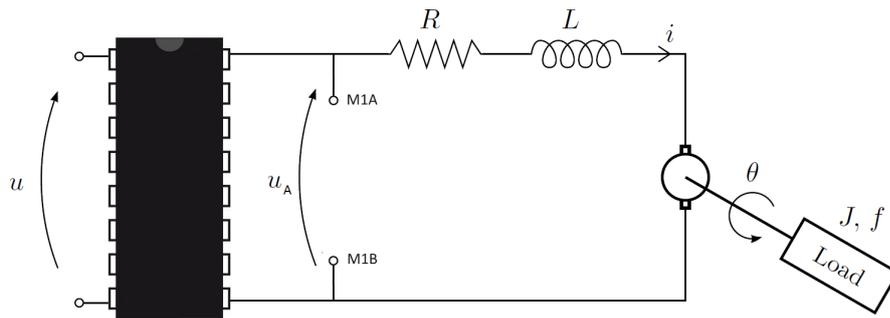
The resistance  $R$  of the motor is obtained by measuring the resistance over the motor using a multimeter. However, we do not have direct access to the cable attachments. Instead, we perform this measurement on the Arduino board. The resistance of the motor can be obtained by measuring the resistance between the two points labeled **M1A** and **M1B**, as illustrated in Figure 3.4 and Figure 3.5. These two points are in connection with the motor attachment, and placed on the board to simplify motor measurements.



**Figure 3.4:** Motor resistance and voltage measurements are done between the points labeled **M1A** and **M1B** next to the motor connection.

**Task 1 (Resistance identification)** *Make sure the Arduino board is unpowered (i.e., USB cable disconnected). Put the multimeter in resistance measurement mode ( $\Omega$ ), and measure the resistance. You typically see a value between  $3\Omega$  and  $6\Omega$ . Connect the USB cable after performing the measurements. Note that it takes the multimeter a couple of seconds to stabilize on a final value.*





**Figure 3.5:** Schematic picture of the measurement points **M1A** and **M1B** and the relationship between the requested control voltage  $u(t)$  and the actually applied voltage  $u_A(t)$ . A large voltage loss occurs in the cheap motor driver chip (actual pin configuration might not be correct, picture is only for illustration)

### 3.1.3 Steady-state measurements for identification of $k_a$ , $k_v$ and $f$

In a batch of experiments, we will study the steady-state currents and angular velocities, for different constant applied motor voltages. For a constant applied motor voltage  $u_A(t) = u_{ss}$ , we denote the corresponding obtained steady-state value of the current and applied voltage

$$i_{ss} = \lim_{t \rightarrow \infty} i(t) \quad (3.4)$$

$$\omega_{ss} = \lim_{t \rightarrow \infty} \omega(t) \quad (3.5)$$

**Preparation 12** Based on (3.1), which equation holds at steady-state, and how can this relationship be used to estimate  $k_v$  by measuring steady-state values of voltages, currents and angular velocities? You can assume  $R$  is known by now.

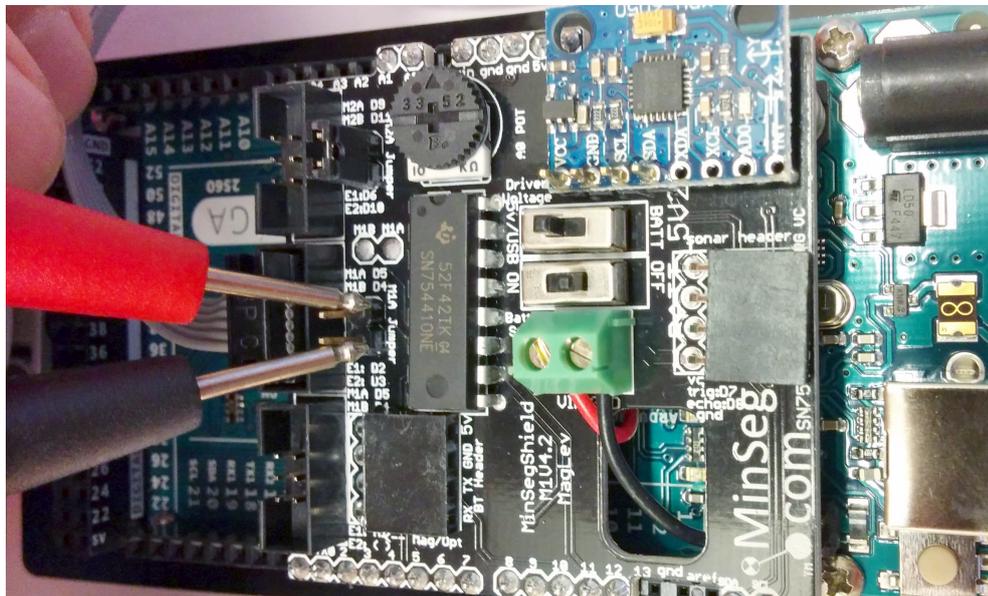
To proceed, we must obtain measurements of the voltage  $u_A(t)$  on the motor, the current  $i(t)$  in the motor, and the angular velocity  $\omega(t)$ . In the following paragraphs we explain how this is done. The actual measurements are done later!

## Measuring motor voltage

When the motor is spinning at steady-state, the applied voltage  $u_A(t)$  can be measured over the same positions as we did for the resistance measurement. To do so, we will put the multimeter in DC-voltage mode ( $V=$ ), and measure the voltage between **M1A** and **M1B**.

## Measuring motor current

To measure current in the motor, we will open the electrical circuit (voltages are measured in parallel over objects, while currents are measured in series). The board is prepared for such a measurement. On the board just in front of the motor connection, there is a jumper (with a colored tape handle) which can be removed to open the electrical circuit. To measure the current, we will put the multimeter in current (**mA**) mode and close the circuit with the multimeter.



**Figure 3.6:** By removing the jumper next to the motor connection, the circuit is opened and we can measure current over the two pins. Be careful not to bend the pins by applying excessive force.

## Measuring angular velocity

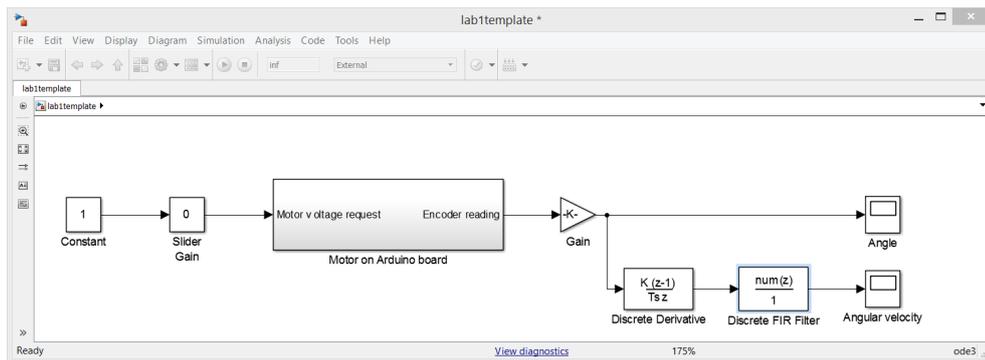
The encoder only gives us angles, so we have to estimate the angular velocity. Given two measurements of the angle  $\theta(t)$  and  $\theta(t - T_s)$  where  $T_s$  is the sampling time (in our case fixed to 0.03s), the most basic estimate of the angular velocity is the Euler backwards approximation

$$\omega(t) = \frac{\theta(t) - \theta(t - T_s)}{T_s} \quad (3.6)$$

Unfortunately, the estimate obtained from this simple computation will not behave well when the code runs in external model. The problem is that the Arduino computer is slow, and when it has to communicate with MATLAB, it will not be able to finish computations in time sometimes, which means that the actual sampling time will be different from the fixed value 0.03s that is used in the computation. As an effect, the derivative estimate will look noisy. We will counteract this by continuously taking the average of the last 20 derivative estimates (a low-pass filter called an FIR filter)

Open the SIMULINK component library (either by writing `simulink` in the MATLAB command prompt, or through the menu in your SIMULINK model **View/Library browser**).

**Task 2 (Code for velocity computation)** *Update your SIMULINK model to incorporate the derivative computations as in Figure 3.7. The block **Discrete Derivative** (which implements the Euler approximation) and **Discrete FIR filter** are both found under **Simulink/Discrete**. A so called plot scope can be found under **Sinks**, or simply copy the scope already available in the model. Note that the text under blocks is completely arbitrary, and you can change this. Double-click the FIR filter block and change the code in the field coefficients to `repmat(0.05,1,20)` (everything in boxed bold text!). This code will create a vector of length 20 with all elements equal to 0.05, which leads to a filter which takes the average of the 20 last values. Save your model.*



**Figure 3.7:** Updated SIMULINK model with derivative computation.

### 3.1.4 Experiments

Let us now perform the experiments. We will run the motor with different requested voltages  $u(t) = 2.5V$ ,  $u(t) = 3.5V$  and  $u(t) = 4.5V$  (set in the middle edit box in the slider gain while running), and record the resulting steady-state applied motor voltage  $u_A(t) = u_{ss}$ , currents  $i_{ss}$  and angular velocities  $\omega_{ss}$ .

**Task 3 (Measurements)** *Attach the USB-cable (preferably in the USB connection on the monitor) and download your code to the Arduino by pressing the green run button. For convenience, perform the experiments in two steps.*

*In a first run, record steady-state motor voltage  $u_{ss}$  (multimeter) and angular velocity  $\omega_{ss}$  (plot scope) with the the three different requested voltages. To set the requested motor voltage in the Slider gain precisely, use the middle edit box. In the angular velocity plot, you will have to right-click in the plot and select auto-scale, or press auto-scale in the plot menu. Record the values of all voltage and angular velocity measurements in the table below.*

*In a second run, remove the motor jumper and repeat the 3 experiments measuring the current  $i_{ss}$  and insert in the table. The columns for  $k_v$  and  $f$  will be filled later during the lab.*

*You should see applied voltages slightly below the requested voltages, currents in the range 20mA to 50mA, and rotational velocities up to at most 7 rad/s or so.*

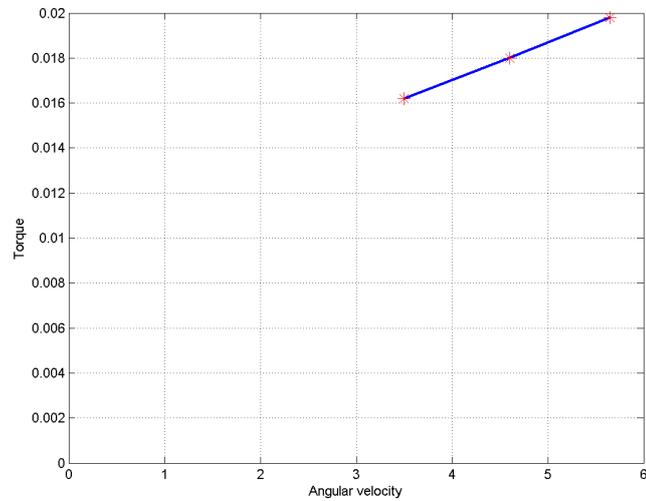
Requested $u(t)$	$u_{ss}$	$\omega_{ss}$	$i_{ss}$	$k_v$	$f$
2.5V					
3.5V					
4.5V					

**Task 4 (Compute  $k_v$  estimates)** Once you have all the measurements, you can compute estimates of the back-emf constant  $k_v$  using your result in preparation 12. Compute estimates of  $k_v$  based on the three experiments and insert in the table.

**Task 5 (Compute final  $k_v$  estimate)** To decide on a final value of  $k_v$ , one can for instance compute the average value of the three estimates. However, a more robust approach which will guard us against a failed experiment (bad measurement, mistake in computation,...), is to take the median instead, which in this case will be the middle value. What is your final estimate of  $k_v$ ?

**Task 6 (Compute  $k_a$ )** With  $k_v$  available, what is the estimated value of the torque constant  $k_a$ ?

With all measurements available, and the estimated value of  $k_a$ , we are ready to use (3.2) to estimate the friction coefficient  $f$ . However, the simple linear differential equation does not tell the whole truth. Besides the linear velocity dependent friction torque  $f\omega(t)$ , there is also a static friction in the motor which prevents the motor from starting to turn and affects the performance at low velocities. You can see this if you decrease the control signal  $u(t)$  until the motor just barely turns. The current will be significantly non-zero already at that point. You would also see this if you would plot the computed torques  $k_a i_{ss}$  against the angular velocity  $\omega_{ss}$ , as exemplified in Figure (3.8). The friction coefficient  $f$  is the slope of the curve, but the curve will not pass through the origin.



**Figure 3.8:** Typical torque and angular velocity measurements. A significant amount of current (i.e., torque  $k_a i_{ss}$ ) is lost on overcoming the static friction, the torque-angular velocity line does not go through the origin.

If we let  $i_0$  denote the current required at very low speed, our model changes to (valid when  $i(t) \geq i_0$ )

$$k_a(i(t) - i_0) = J\dot{\omega}(t) + f\omega(t) \quad (3.7)$$

**Preparation 13** How can  $f$  be computed from steady-state measurements  $i_{ss}$  and  $\omega_{ss}$  if we know  $i_0$  and  $k_a$ , using the model (3.7)



**Task 7 (Identify  $i_0$ )** Find out the current  $i_0$  which is used when the motor turns very slowly by decreasing the requested voltage until it is turning very

*slowly. When you are done, turn the voltage to 0, stop the code, and re-install the jumper.*

**Task 8 (Compute  $f$ )** *Use the model and the value of  $i_0$  to compute estimates of  $f$  from the three experiments, and complete the table. Use the middle value as an estimate of  $f$ .*

### 3.1.5 Dynamic analysis for identification of J

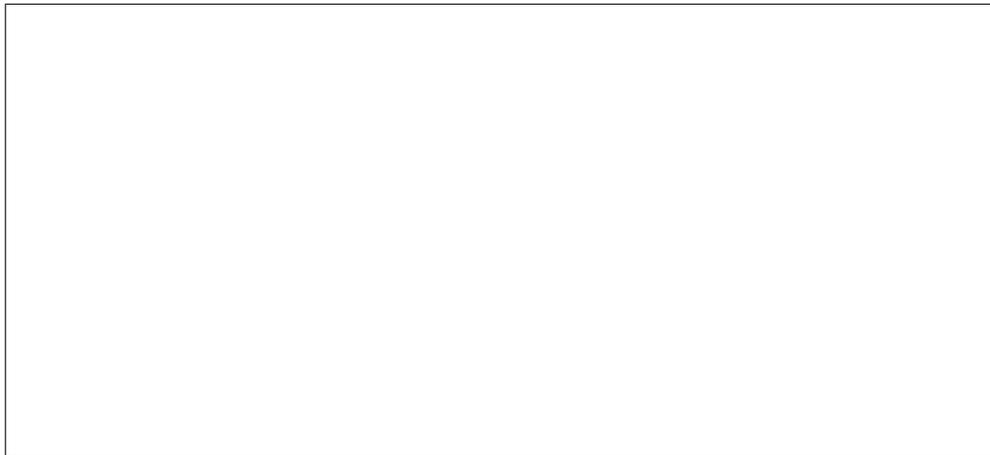
Through static experiments (i.e., analysis of steady-state signals), we have managed to identify 3 out of the 4 parameters. The parameter J requires us to study the dynamic behavior. The fact that we need dynamic analysis to estimate J is natural as it only enters our equations multiplied with  $\dot{\omega}(t)$ , which is 0 in a steady-state analysis.

The constant J effectively makes it harder to accelerate the motor. The larger (heavier wheels) J is, the longer time it will take for the motor to reach steady-state velocity. Hence, our analysis will be based on estimating the time-constant of the system from  $u_A(t)$  to  $\omega(t)$ .

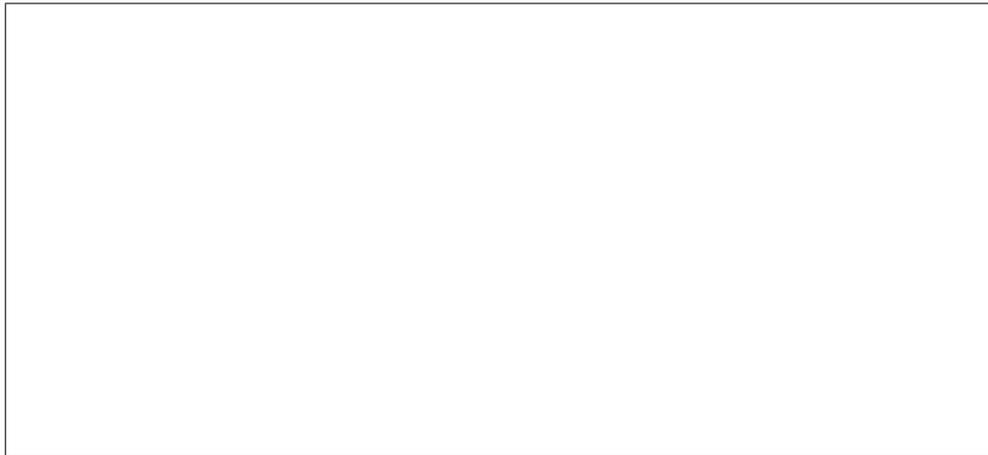
By combining (3.1) and (3.2) we arrive at the differential equation

$$RJ\dot{\omega}(t) = -(Rf + k_v k_a)\omega(t) + k_a u_A(t) \quad (3.8)$$

**Preparation 14** *Prove (3.8)*



**Preparation 15** *Write the differential-equation (3.8) as a standard first-order system  $T\dot{\omega}(t) + \omega(t) = Ku_A(t)$ , i.e., derive the time-constant and static gain of the system (3.8), in terms of the parameters R, J, f,  $k_v$  and  $k_a$ .*



**Preparation 16** *If you know the time-constant of the system, and all parameters except  $J$ , how can you compute  $J$ ?*



In our final experiment, we are going to study the transient behavior of the velocity during a step-change in applied voltage. Open the SIMULINK model `template2` which has been prepared such that it will generate a series of steps in requested voltage  $u(t)$  from 0V to 4.5V, which will lead to a series of steps in applied voltage  $u_A(t)$  following your previously developed table. The approximate angular velocity is sent to MATLAB and recorded in a variable `data` in the MATLAB workspace.

**Task 9 (Collect step-response data)** *Download and run the modified model. You should see a sequence of steps being performed on the motor. Let it perform a couple of those, and then stop the code (preferably when no voltage is applied).*

**Task 10 (Study step-response data)** *Plot the step-responses (slightly smoothed to make it easier to read values) in MATLAB by running*

```
plot(data.time,smooth(data.signals.values))
```

*Study the plot of the steps (zoom in on a single step!). What is the steady-state value of  $\omega(t)$  in the steps? What was the applied steady-state voltage  $u_{ss}$  on the motor when requesting 4.5V according to earlier experiments? (of course, the steady-state angular velocity should coincide with the value in the table also). Based on this, what is the experimentally derived static gain from applied voltage  $u_A(t) = u_{ss}$  to angular velocity  $\omega(t)$ ?*

**Task 11** *Based on the plot of the step-response of the motor, what is the experimentally derived time-constant?*

**Task 12** *Based on the experimentally derived time-constant and previously derived physical parameters, what is the value of J?*

**Task 13** *To summarize, what does the model look like (with numerical values) if you write it as a standard first-order model in the form used in preparation 2 and 7.*

## Comments on the model

A lot of effort has been placed in this lab on experimentally identifying the physical parameters  $J$ ,  $R$ ,  $f$ ,  $k_v$  and  $k_a$ . However, in the end, these 5 parameters are used in a first-order system which just as well can be defined directly from the experimentally obtained time-constant and static gain. In many situations, this modeling approach is just as useful, as it completely describes the input-output behavior of the system. A simple step-response experiment, and we have a sufficiently precise model to simulate the system, design controllers, and analyze performance. Performing step-responses on system to quickly derive models is extremely common practice, and often leads to sufficiently good models to be used for control.

The first-order model derived in this lab is used for analysis in forthcoming labs, and is part of a complete model of the MinSeg, required for the development of a balance controller.

## 3.2 Summary and reflections

Summarize and reflect on concepts in this lab.

Questions	Answers
1. The MinSeg DC-motor is	<input type="checkbox"/> a model <input type="checkbox"/> a system <input type="checkbox"/> a signal
2. The differential equation describing the MinSeg DC-motor is	<input type="checkbox"/> a model <input type="checkbox"/> a system <input type="checkbox"/> a signal <input type="checkbox"/> a parameter
3. The time-constant describes	<input type="checkbox"/> How fast the system reacts to changing inputs <input type="checkbox"/> Which constant value the output will converge to
4. The static gain K describes	<input type="checkbox"/> The ratio between the output and input amplitudes in steady-state <input type="checkbox"/> The amplitude of the output
5. The final value of the output depends on	<input type="checkbox"/> The time-constant <input type="checkbox"/> The input amplitude <input type="checkbox"/> The static gain
6. A first-order linear differential equation is	<input type="checkbox"/> Uniquely defined by a time-constant and static gain <input type="checkbox"/> Is not uniquely defined by time-constant and static gain

**Most unclear to me is still:** .....

.....