Lab 11: DC Motor Open-Loop Response

1. Introduction

In this lab students will experimentally identify the open-loop transfer function for a DC motor.

1.1 Lab Objectives

- Analyze the open-loop step response a DC motor to create a 1st order dynamic model

1.2 Project Objectives

- Create a 1st order open-loop dynamic model for your robot drive system
- Prepare for PM 12

1.3 Lab Hardware

- Lab Robot
- Multimeter

1.4 Project hardware

- Purchase hardware as needed

2. Laboratory Concepts

2.1 DC Motor Model

An electromechanical model for a DC motor is shown in Figure 1.

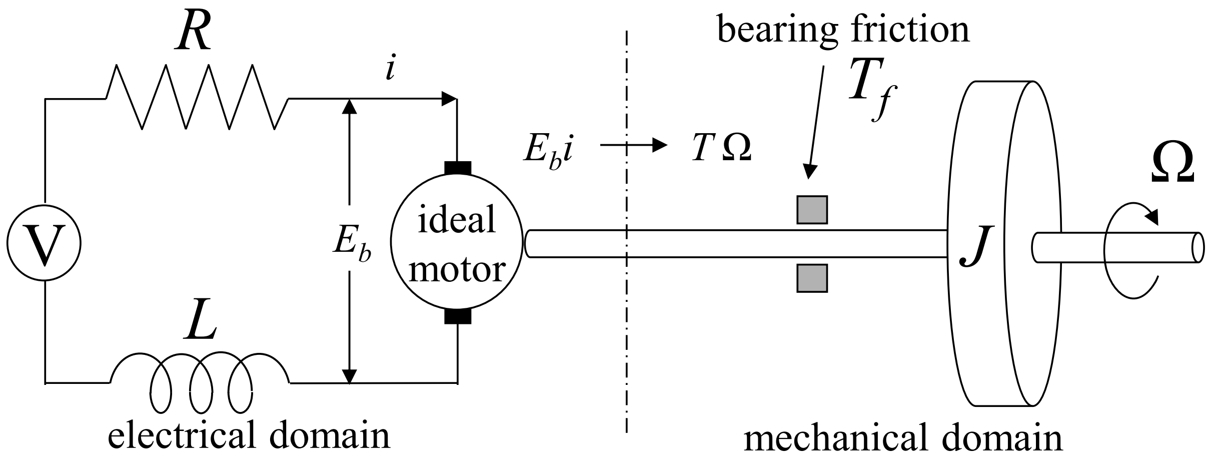

Figure 1. Lumped parameter DC motor model.

In the electrical domain, the applied voltage is *V* and the current flow is *i*. The resistance and inductance of the motor windings are *R* and *L* respectively. In the mechanical domain, the torque produced is *T* and the angular velocity is *Ω*. The mechanical load is comprised of rotational inertia *J*, and frictional torque T_f. The electrical and mechanical domains of the model are coupled by the current-torque equation (1) and back-EMF equation (2).

$$T = K_t i \tag{1}$$

$$E_b = K_b \Omega \tag{2}$$

Constants K_t and K_b are parameters for the motor that relate torque to current and velocity to back-EMF voltage. If both parameters are in standard SI units, they should be equal in value.

Model Derivation

Applying Kirchhoff's voltage law around the electrical circuit yields (3), and applying D'Alembert's Law (Newtons' second law) to the rotational inertia yields (4).

$$V = iR + L\frac{di}{dt} + K_b \Omega \tag{3}$$

$$J\frac{d\Omega}{dt} + T_f = K_t i \tag{4}$$

The simplest friction model is described by the viscous friction (damping) coefficient B_v, which results in a friction torque that increases linearly with rotational velocity as described by (5).

$$T_f = B_v \Omega \tag{5}$$

Table 1 summarizes the parameters and their units.

Table 1. DC motor model parameters

Name	Symbol	SI Units
Motor Resistance	R	Ω
Inductance	L	H
Back EMF Constant	K_b	V/[rad/s]
Torque Constant	K_t	N-m/A
Viscous Friction Coefficient	B_v	N-m-s
Rotational Inertia	J	Kg-m^2

If we neglect the motor inductance (assume $L = 0$), and make sure that the motor constants are in SI units ($K_t = K_b$), then we can use equations (3)-(5) to derive the following 1st order transfer function:

$$\frac{\Omega(s)}{V(s)} = \frac{\frac{K_t}{RJ}}{s + \frac{B_v R + K_t^2}{RJ}} \tag{6}$$

We refer to equation (6) as the *Open-Loop Transfer Function* of the system, which relates the output of the motor (in this case angular velocity) to the input of the motor (in this case voltage). The term *open-loop* refers to the fact that we have not yet included any feedback control when deriving this equation. If we can experimentally identify the numerical values of this *open-loop* transfer function, we can design appropriate gain values for feedback controllers, which we will do in Lab 12.

2.2 Step Response Parameter Identification

In the previous section, a 1st order open-loop transfer function for the motor was presented in equation (6). This model can be generalized to the form:

$$G(s) = \frac{\Omega(s)}{V(s)} = \frac{K}{\tau s + 1} \tag{7}$$

where K is the steady-state gain, and τ is the time constant, which are both combinations of the parameters in Table 1. Thus we only need to experimentally identify two parameters (K and τ) in order to model the open-loop dynamics of the motor. We can do this by applying a step input (i.e. a step in Voltage) and measuring the output (angular velocity) vs. time.

If the input V(s) is a step input of magnitude *d* (Volts), then:

$$V(s) = \frac{d}{s} \tag{8}$$

$$\Omega(s) = V(s)G(s) = \frac{d}{s}\left(\frac{K}{\tau s + 1}\right) \tag{9}$$

Taking the inverse Laplace transform gives an exponentially converging time response:

$$\Omega(t) = Kd\left(1 - e^{-\frac{t}{\tau}}\right) = \Omega_{ss}\left(1 - e^{-\frac{t}{\tau}}\right) \tag{10}$$

where τ is the time constant and Ω_{ss} is the steady-state speed. A typical plot of $\Omega(t)$ is shown in Figure 2, which is a generic first order response.

In the lab, you will collect experimental data and plot it versus time, t, which will also resemble Figure 2. The time constant τ can be approximated by finding the time when the response has reached 63.2% of the steady state speed Ω_{ss}.

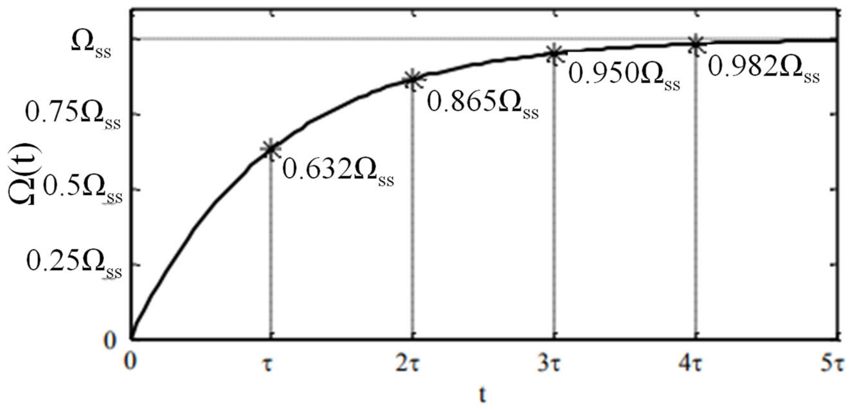

Figure 2. Generalized step response of a 1st order system used to find the time constant.

Since the data is usually noisy, a better method for estimating the time constant is to use a semi-log regression. First take the natural log of both sides of (10). This results in:

$$\ln\left(1 - \frac{\Omega(t)}{\Omega_{ss}}\right) = -\frac{1}{\tau}t \tag{11}$$

Hence, plotting t on the x-axis and $\ln(1-\Omega(t)/\Omega_{ss})$ on the y-axis should provide a line with slope $-\frac{1}{\tau}$ and a bias close to zero. **Practically, this only works if the step response portion of the data is used for regression since Ω(t) will be noisy and may exceed Ω_{ss}, which makes the natural log undefined. Thus, it is wise to only use the data before this occurs, which is typically $t < 4\tau$.** Given typically noisy data, a regression like this results in a more robust estimation of the time constant than just estimating it from a single point on the step response, $\Omega(t)$.

By finding Ω_{ss} and the time constant, τ, we have now completely identified the open-loop transfer function of equation (7), where the steady-state gain $K = \Omega_{ss}/d$.

2.3 Pulse Width Modulation (PWM) and Motor Drivers

In the preceding sections, we have assumed that we are able to command the motor voltage $V(t)$ as if it were an analog signal. However, Arduinos do not have a true analog output, and instead rely on Pulse Width Modulation (PWM) to approximate an analog output using a digital I/O pin. When you use the *AnalogWrite()* command, the Arduino generates a PWM signal, which is a periodic high-frequency digital pulse train as shown in Figure 3. The *pulse width* is the amount of time during each pulse *period* that the voltage is high (5V) versus low (0V). The *Duty Cycle* is used to describe the percentage of this time (e.g. $Duty\ Cycle = \frac{pulse\ width}{period}$x100%), such that a signal with 0% duty cycle is always off and 100% duty cycle is always on. The low-voltage low-power digital I/O signal from the Arduino is then amplified by a motor driver to a create high-voltage high-powered PWM signal suitable to drive the motor.

One might suspect that the choppy nature of the PWM signal would cause the motor to vibrate. However, the PWM frequency is usually designed to be much faster than the electrical/mechanical dynamics of the motor, such that the motor itself acts as a low-pass filter that averages out the voltage. Thus for practical purposes, the PWM signal acts as an average voltage, $V_{average}$, whose magnitude is determined by the duty cycle: $V_{average} = Duty\ Cycle * V^+$ where V^+ is the maximum output voltage of the motor driver, which is slightly below the battery supply voltage due to a small voltage drop across the motor driver transistors. The *AnalogWrite()* command on the Arduino varies the duty cycle to effectively vary $V_{average}$. A smaller duty cycle results in a lower $V_{average}$, while a larger duty cycle results in a larger $V_{average}$, as illustrated in Figure 3. One subtlety is that if the system is connected to a battery, as in your project, the battery supply voltage will decrease as the battery discharges and when large loads are applied due to battery internal resistance. Hence, the Arduino and motor driver will not necessarily produce consistent output voltages or motor responses, which is why feedback control is important.

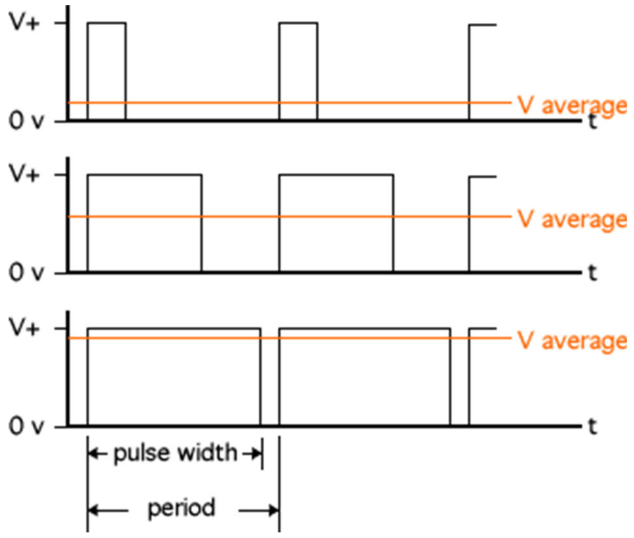

Figure 3. Pulse Width Modulation Signal

Finally, note that when you are commanding a motor driver using library functions like *SetSpeed()*, you are not actually commanding a speed. As with the *AnalogWrite()* command, you are actually commanding the Duty Cycle of the PWM signal, and therefore the $V_{average}$.

3. Pre-Lab Exercises

1. Starting from equations (3) and (4) derive the transfer function (6).

2. By equating the transfer functions in equations (6) and (7), derive algebraic expressions for K and τ in terms of the individual motor parameters (K_t, R, B_v, J).

3. Suppose a motor has the following parameter values:

K_t	0.05 N·m
R	2 Ω
B_v	0.015 N·m/rad/s
J	0.001 N·m/rad/s^2

   a. Use the parameter values in the table to compute the numerical values of K and τ for this motor.

   b. Use the *tf()* and *step()* commands in MATLAB to simulate the theoretical step response of this motor. Plot the voltage vs. time and velocity vs. time for a unit step of 1 Volt ($d=1$). Verify that the steady-state speed $\Omega_{ss} = K/d$.

   c. Plot $\ln(1-\Omega(t)/\Omega_{ss})$ vs. time for $t < 4\tau$ and verify that the slope $= -\frac{1}{\tau}$

4. A PWM signal with a frequency of 20kHz is used to command a motor driver that is powered by a 9 Volt battery. The *setSpeed()* command in the motor driver library accepts a number from 0 to 400, where 400 represents a 100% duty cycle. If you send a *setSpeed()* command of 300, what duty cycle does this correspond to, and what will be average motor voltage? What is the corresponding pulse width (in seconds) of the PWM signal?

4. Laboratory Exercises

4.1 Motor Step Response

4.1.1 Begin with ***Lab11_template,*** but save it as a new sketch ***Lab11_StepInput.*** Modify it to do the following:

- $Motor\ command = \begin{cases} 0 < t < 1 & 0\,V \\ 1 < t < 2 & 5\,V \\ t > 2 & 0\,V \end{cases}$
- It should print these values to the serial monitor: time (sec), Velocity (rad/s), and the motor command for 0 < t < 2 seconds. There should be a new line after the motor command. Print the values for each variable with no labels. You should print a tab character between each value.
- Upload the sketch to the Mega. Verify that the serial printing is in the correct format.

4.1.2 Download the MATLAB script ***ArduinoSerialPlot.m***

- This script will read the serial monitor from your Arduino and plot it as long as each variable is printed with a tab character between values.
- By changing the ***COM*** string, you can pick which device is being read from. Change this to match your Arduino's COM channel. (Lower case)
- By changing the ***BaudRate*** value, you can pick the communication speed of the serial communication. Change this to match your Arduino's baud rate.
- Running this script is like opening the serial monitor. It initiates the sketch running on the Arduino.

4.1.3 Run the ***ArduinoSerialPlot.m.*** It will run your Arduino sketch. It will also plot the data assuming your first variable per line is time. It will then plot all other data vs time. The data from the Arduino will also be saved in the workspace as the variable, ***data***.

4.1.4 Save the MATLAB variables **`t`**, **`V`**, and **`M`** (time, velocity, and motor command) for later. You will need them for the post lab. If you don't want to save three files you can save ***data*** which contains these three variables.
Hint: Right-click the variables you want to save and select "Save As".

4.1.5 Show the TA your step response.

Have your TA check your progress

4.2 Robot Step Response

4.2.1 Begin with ***Lab11_StepInput***, and save a new copy as ***Lab11_DrivingStepInput.*** Modify it to do the following:

- Apply a simultaneous step input to both motors
- Read both encoders and compute the velocity of both wheels in rad/s
- Compute the average velocity of the two wheels in rad/s
- Print the average velocity in rad/s to the serial monitor instead of the velocity of a single wheel

- With your robot still up on blocks, run the ***ArduinoSerialPlot.m*** to verify that everything works. Then place your robot with the wheels on the benchtop and run ***ArduinoSerialPlot.m*** to collect a step response for the robot actually driving. Save your data for later (make sure to not mix this data up with your data from 4.1). Note that the transfer function you get from this step response will be different, because it will include the inertia of the entire robot, and not just the inertia of the wheel

4.2.2 Show the TA your step response.

Have your TA check your progress

4.3 Comparison of Theoretical vs. Experimental Transfer Functions

Complete the post-lab problems in lab. As always, lab partners may share data and consult with each other, but each student should work on their own computer to generate their own plots.

Have your TA check your results

5. Post-Lab Exercises

Complete the following exercises per pre- and post-lab submission policy posted on Canvas. Remember, post-lab exercises are completed independently by each student unless otherwise specified. Include your name(s) and lab section in every plot title.

1. Using the step response data you measured in part 4.1:
   a. Plot the step response rotational velocity (rad/s) vs time (sec). Place a data point at the estimated time constant (τ, 0.632* Ω_{ss}). Display the estimated time constant on plot. Include the properly labeled plot.
   b. Now estimate the time constant using the semi-log method described in the lab documentation. Include your properly labeled plot and regression.
   c. Estimate the transfer function in equation (7) for your system using the time constant and steady-state speed from your step response.
   d. Use the *tf()* and *step()* commands in MATLAB to simulate the theoretical step response of your transfer function and plot it overtop of your experimental step response. Discuss any discrepancies. (Note: The *step()* function assumes an input of 1, so be sure to scale the output based on our input of *d).*
2. Using the step response data you measured in part 4.2:
   a. Plot the step response average velocity (rad/s) vs. time (sec), and estimate the steady-state average velocity.
   b. Use the semi-log method to estimate the time constant.
   c. Using the steady-state average velocity and time constant, write a 1st order transfer function for your robot., $\left(\frac{\Omega(s)}{V(s)}\right)$, where Ω is the average velocity of the wheels, and V is the motor voltage. Note that this transfer function will be slightly different than the one in exercise 4.1, because it will now include the inertia of the entire robot, rather than just the inertia of the wheel.
   d. Use the *tf()* and *step()* commands in MATLAB to simulate the theoretical step response of your transfer function and plot it overtop of your experimental step response.

6. Project Milestone 11

In preparation for PM 12, teams will develop a transfer function for your robot and present the current state of your robot in a 5-10 minute PowerPoint presentation. You will submit your presentation with your team number in the filename on Canvas to the PM11 assignment.

Transfer Function Estimation

In Lab 11 you learned how to collect step response data for a geared DC motor (70:1). From this step response you developed a 1st order transfer function. You will use this knowledge to develop a transfer function for your robot drive system, $\left(\frac{\Omega(s)}{V(s)}\right)$, where Ω is the average rotational velocity of your wheels in rad/s, and V is the motor voltage. For two drive motors, you will have to simultaneously apply step inputs to both motors, as in lab exercise 4.2.

Similar to the code you wrote in lab, you should apply a step voltage to your robot drive motor(s) while it is on the ground. You should measure the encoder(s) of each drive motor and calculate the translational speed of the robot (for two motors/encoders, just use the average translational speed from both). Using ***ArduinoSerialPlot.m*** or printing to the Serial monitor may be useful for collecting data.

You will present your experimental method, your experimental step response, and your 1st order model (compare your experimental step response to the simulated step response of your model) in PM11. Collecting video documentation of your experiment is advised.

Presentation

- Discuss each team members contributions for this milestone
- Transfer Function Estimation
  - Discuss your experimental method
  - Show your step response plots
    - Plot the voltage vs. time and the and average wheel velocity vs. time
  - Show your transfer function equation
    - Include values for your steady-state gain K and time constant τ
  - Plot your transfer function response over your experimental step response data so they can be compared.
  - Discuss the accuracy of your models
- Your current progress for completing course objectives.
  - Discuss where your robot needs to be for PM 12 and compare to where it is currently. What are your major problems?
  - What is your plan for reaching your final goal?