AERSP304 Funduino Project
Assignment 1

Spring 2026

Due: Date

1 Simulating a DC Motor

This following project is intended to present some basic ideas of system identification and control. The actuator that will be used is an electric DC motor. Your first task in this series of experiments will be to identify a transfer function depicting the relationship between input (voltage) and output (angular speed) of the motor.

A DC-motor with an encoder is used for this Arduino assignment. The electrical and mechanical differential equations governing the dynamic system will be derived later. For now, let's understand how this actuator works.

An electrical motor is a machine that converts electrical energy (voltage) in mechanical energy (torque). Figure 1 shows a schematic diagram for a DC-motor. The basic principle is that whenever a current carrying conductor is placed in a magnetic field, it experiences a mechanical force. When an input DC voltage V_a is applied to its terminals, a current i_a develops and accelerates the motor due to the force generated due to the magnetic field. In most general cases, the magnetic field is generated by permanent magnets positioned inside the structure of the motor. A commutator is used in the interface between the static part and the rotational part in order to achieve unidirectional torque. As long as the DC voltage supply stays connected and providing enough power, the shaft will keep rotating with an angular speed ω. Once there are also conductors rotating in the magnetic field, a back- EMF V_g is generated in the opposite direction trying to compensate that excitation.

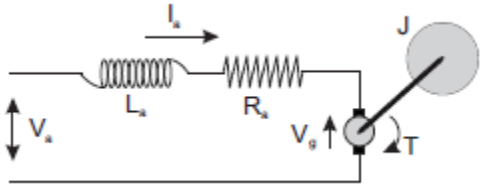

Figure 1: Schematics for the motor.

Other parameters of the motor are its inductance L_a and the armature resistance R_a. On the mechanical side there is the shaft inertia J. For a simplistic model, the torque associated with friction, T is negligible.

1.1 Mathematical model of the DC motor

Once the current is developed through the conductor, there are three voltage drops that can be represented by mathematical equations in the electrical side of the actuator. Using Kirchhoff's voltage law:

$$L_a \frac{di_a}{dt} = V_a - R_a \cdot i_a - V_g \tag{1}$$

When the armature is conducting current in the magnetic field, it produces a mechanical torque on the shaft proportional to the current. So:

$$T = K_m \cdot i_a. \tag{2}$$

As it rotates in the magnetic field it also generates a voltage proportional to the speed of rotation:

$$V_g = K_g \cdot \omega. \tag{3}$$

The angular velocity is defined as:

$$\omega(t) = \frac{d\theta}{dt} = \dot{\theta}(t). \tag{4}$$

On the mechanical side, $F = m \cdot a$ is applied in rotational form and related to (2) that:

$$J \frac{d\omega}{dt} = K_m \cdot i_a \tag{5}$$

These equations are the basis for mathematical model of the DC motor in this project.

2 Transfer functions

Prior to experimentally determining the transfer function, use equations (1) to (5) to derive theoretically (keep all parameters in a generic form):

1. the transfer function between the angular velocity and the input voltage $G(s) = \frac{\omega(s)}{V_a(s)}$;

2. the position transfer function $H(s) = \frac{\theta(s)}{V_a(s)}$;

3 Transfer Function Identification

Even though, we can mathematically model the motor but we do not know the values of different parameters appearing in these equations. In this section, we will use the experimental data to identify the equivalent transfer function for the motor. The basic idea is to apply different voltages in the input and then measure the response of the motor in its angular velocity. Also, make sure that you use the appropriate time interval for both saving the voltage as well as for identifying and saving the angular velocity. A good time interval will provide enough points in every change of level. Make sure that you also wait until the motor settles in the new set point before changing it.

The next two sections explain how to record the input (voltage) and output (angular velocity) of the motor.

3.1 Rotary encoder

Generally motor is equipped with an encoder which provides the angular speed of the motor. As an example here, the model RK-370SD-2470 DC 6V is equipped with a simple encoder 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. To compute the counts per revolution of the gearbox output, multiply the gear ratio by 48. The gear ratio for this model is 9.68:1.

A good strategy is to use fast input counters or interruption to read pulses from the encoder. That will help to achieve a more accurate model.

3.2 Measuring voltage

The other important information is the voltage applied to the motor. There are several ways to do that but the simplest one (not the most accurate though) is converting the Pulse-Width Modulation (PWM) signal to voltage levels by simply multiplying the volt/level ratio. It will be necessary to know what is the DC supply voltage and the total number of levels (8 bit, 16 bit or other) of the PWM signal.

A second option to measure the voltage applied to the motor is using analog inputs. Be careful because some Arduino boards allows a 5V maximum level in the analog inputs. Applying values higher than those could damage the board. If that is the case, it is necessary to use some conditioning circuit as explained in your earlier getting started document.

3.3 MATLAB and Arduino

You can use both MATLAB or Arduino code to generate the signals to be used on the Arduino, as well as to analyze the output of data from the Arduino during the experiments. Feel free to chose the one you are comfortable with. You will need to save and read data. To keep control over the format the data is saved in, low level read and write functions will be used. Make sure you check the help for all the functions described in the following!

If using MATLAB, for example, to open a file simply use the *fopen* command. But don't forget to close the file with *fclose* once your finished with writing or reading. Writing is done using *fprintf* and reading is done using *fscanf*. The easiest is to look at an example on how to use them. First we will write a vector *a* into a file *write.txt*: Reading is done in a similar way. We will read the data

```
a = 1:0.1:10;
a = a';                          % to save the vector in transposed
                                 % form, IMPORTANT

fidw = fopen('write.txt','w');
fprintf(fidw,'%.6f\r\n',a);
fclose(fidw);                    % IMPORTANT
```

```
fidr = fopen('write.txt','r');
b = fscanf(fidr,'%f');
fclose(fidr);                    % IMPORTANT
```

we just created in a vector *b*:

Another way to do is to send the information to MATLAB and then just store it in a variable for processing. An example is provided as two separate files: *recv.m* is a script in MATLAB that plots two variables received by serial from Arduino. *Example.ino* is the code to run inside the Arduino board in order to send those two variables to MATLAB. It is a 10s simulation. Feel free to use and modify those programs.

3.4 Transfer Function Identification

This section lists the main steps for you to identify the transfer function for the DC motor.

1. Write an Arduino or Matlab program to command a voltage input to the motor and read the encoder data to get the RPM of the motor. Command a step input corresponding to 80% of maximum voltage which your motor can sustain. Since the motor has dynamics (inertia, friction, etc.), it doesn't instantaneously reach top speed when turned on and doesn't immediately come to a stop when turned off. It takes time for the motor to react. You should record at least 10 seconds worth of data for each commanded voltage input. Motor RPM recorded may be noisy depending upon the prescribed sampling time. Vary the sampling time between 0.1 second to 0.01 second to find the appropriate sampling time for which motor RPM data is not very noisy. Save the encoder data (RPM) along with the commanded voltage input in a file.

2. Open the saved data file from the previous step in Matlab and plot RPM vs. time. You will notice that RPM vs. time graph looks like a first order model response than the second order model. This is due to the fact that the motor is overdamped. This is typical of a DC motor where the mechanical dynamics are much slower than the electrical dynamics, and hence dominate the response. So, we can estimate the motor transfer function as:

$$G(s) = \frac{\omega(s)}{V_a(s)} = \frac{K}{\tau s + 1} \tag{6}$$

   where K is the DC gain, i.e., ratio of steady state RPM to applied voltage. For example, if we apply a voltage of 6V and motor RPM is estimated to be 170 RPM, then $K = \frac{170RPM}{6V}$. τ is the time it takes for motor to reach the 63.2% of its steady state RPM.

3. From RPM vs. time plot, record the motor RPM in steady state. You will need to "eye-ball" a fitted line to the motor's RPM graph. Also, note down the time, τ, it takes to reach the approximately 63.2% of its total RPM. Use these values to identify the motor transfer function.

4. Use the identified transfer function to predict motor RPM in Matlab. Compute error between measured motor response (i.e, encoder data) and predicted motor response from the previous step.

5. Repeat steps 1-4 at least five times to compute the parameters of the transfer function by varying the amplitude of commanded input voltage. Average the transfer function parameters, K and τ to get the best estimate of these parameters. Also, compute the standard deviation of these parameters.

6. Use the *chirp* command in Matlab to generate an input voltage profile for the motor. The *chirp* command provides a sinusoidal function of specified frequency range, sample time and length. Predict motor response for the *chirp* signal while making use of identified transfer function. Note that you can easily create a chirp signal in Arduino by making use of the following equation:

$$V_a(t) = A\sin\left(\phi_0 + 2\pi\left(\frac{c}{2}t^2 + \omega_0 t\right)\right), \quad c = \frac{\omega_1 - \omega_0}{T}$$

   Apply the chirp input voltage to the motor and record the motor response. Compute error between measured motor response (i.e, encoder data) and predicted motor response. You are encouraged to pick different *chirp* signals by varying the frequency and amplitude.

4 Generating Report

Show all your steps to produce your transfer function estimates. Clearly indicate the results of your transfer function identification as well as the error analysis conducted. Explain the signals used to determine the system and why those inputs were chosen. Comment on your results qualitatively. Include any figures or plots used for the analysis.

If any methods beyond what are outlined above are used to complete this assignment, please outline them in the report. For example, if you produce the signals directly in Arduino for the inputs to the motor, please detail how this was done and what the results were. The explanations above for the methods to produce the inputs and outputs are not the sole way to do them for this assignment. You are welcome and encouraged to use whatever approach provides the results that are expected.

5 Potentially Useful Links

1. https://www.youtube.com/watch?v=dTGITLnYAY0&ab_channel=CurioRes
2. https://www.mathworks.com/discovery/arduino-programming-matlab-simulink.html
3. https://www.mathworks.com/hardware-support/arduino-matlab.html
4. https://www.mathworks.com/help/signal/ref/chirp.html
5. https://www.arduino.cc/en/Reference/FileRead