Lab 7: System ID

Overview

The primary goal of this lab is to find the transfer function of the DC motor system that has been used in the last several labs. The form of the transfer function will be derived based on first principles. The coefficients of the transfer function will be estimated by curve fitting open-loop pulse responses. The final transfer function will be verified by using it to predict the closed-loop step response under proportional control and comparing the model prediction with experimental results.

Background and Theory

DC motors generate a torque between the rotor and stator that is related to the applied voltage or current. When a voltage is applied, the resulting current generates a torque causing the rotor to accelerate. For any voltage and load on the motor there tends to be a final angular velocity due to friction and drag in the motor. For a given voltage the ratio between steady-state torque and speed will be a straight line as seen in Figure 1.

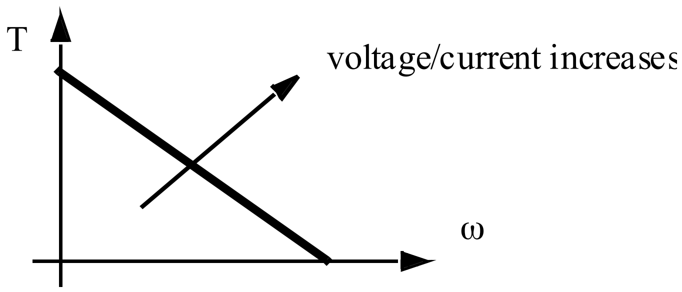

Figure 1: Typical motor characteristic curve relating motor torque to speed.

The basic equivalent circuit model for the motor is shown in Figure 2. Equations can be developed for this model. This model must also include the rotational inertia of the rotor and any attached loads. On the left hand side is the resistance of the motor and the "back emf" dependent voltage source. On the right hand side the inertia components are shown. The rotational inertia J_1 is the motor rotor, and the second inertia is an attached disk.

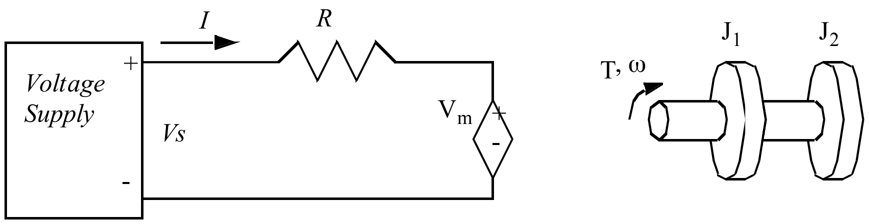

Figure 2: Model and circuit representing a typical brushed DC motor.

Because a motor is basically wires in a magnetic field, the electron flow (current) in the wire will push against the magnetic field. The torque (force) generated will be proportional to the current. A constant K relates the torque to the current:

$$\tau_m = Ki$$

The electrical and mechanicl power of the motor must be the same:

$$P = v_m i = \tau_m \dot{\theta} = Ki\dot{\theta}$$

So,

$$v_m = K\dot{\theta}$$

Summing moments for the motor shaft gives

$$\tau_m - \tau_{load} = J\ddot{\theta}$$

or

$$\tau_m = J\ddot{\theta} + \tau_{load}$$

These equations can be manipulated to give the differential equation involving θ and v_s, the inputs and outputs of the system:

$$i = \frac{v_s - v_m}{R}$$

$$\frac{\tau_m}{K} = \frac{v_s - K\dot{\theta}}{R}$$

$$\frac{J\ddot{\theta} + \tau_{load}}{K} = \frac{v_s - K\dot{\theta}}{R}$$

Resulting in

$$\ddot{\theta} + \frac{K^2}{JR}\dot{\theta} = v_s\left(\frac{K}{JR}\right) - \frac{\tau_{load}}{J}$$

Transfer Function Assignment

Find the transfer function for the DC motor system. What assumptions do you have to make?

Open-Loop Pulse Tests

Run several tests with different pulse widths and amplitudes. Save the data to csv files so that you can use it in curve fitting. Be sure to use inverse deadband compensation in all of your tests for this lab as shown in Figure 3. The transfer function we are seeking is θ/pwm, so that small pwm values should still cause θ to change. This means that the transfer function model can safely ignore deadband because we are compensating for it in software.

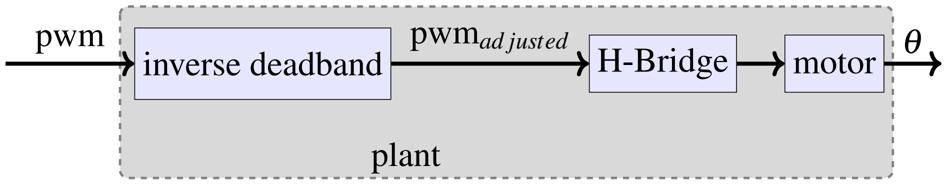

Figure 3: Block diagram of DC motor plant, including inverse deadband compensation

Open-Loop Arduino Requirements

- you should be able to re-use recent code
  - serial interaction code
  - encoder interrupt function
  - function to command the H-bridge for motion in either direction
    - positive and negative inputs
- eliminate your timer ISR if you have one
- your code should prompt the user for pulse width and amplitude
- your Arduino **must** continue to print data after the pulse has switched off
  - the motor will still be coasting after the pulse turns off
  - you **must** capture data until the motor stops moving
  - either prompt the user for how long the test should run or set your own stopping condition in the code
- when a test is running, your Arduino must do the following each time through the `loop` function:
  - get the current time in microseconds using the `micros()` command
  - determine the pwm command to send to the motor
  - print the relevant data to serial

        * time, pwm command, θ, and any other variables you think are important
    - determine number of micro seconds to wait until the next time step

- use **inverse deadband** in all of your testing
- an example pulse response plot is shown in Figure 4

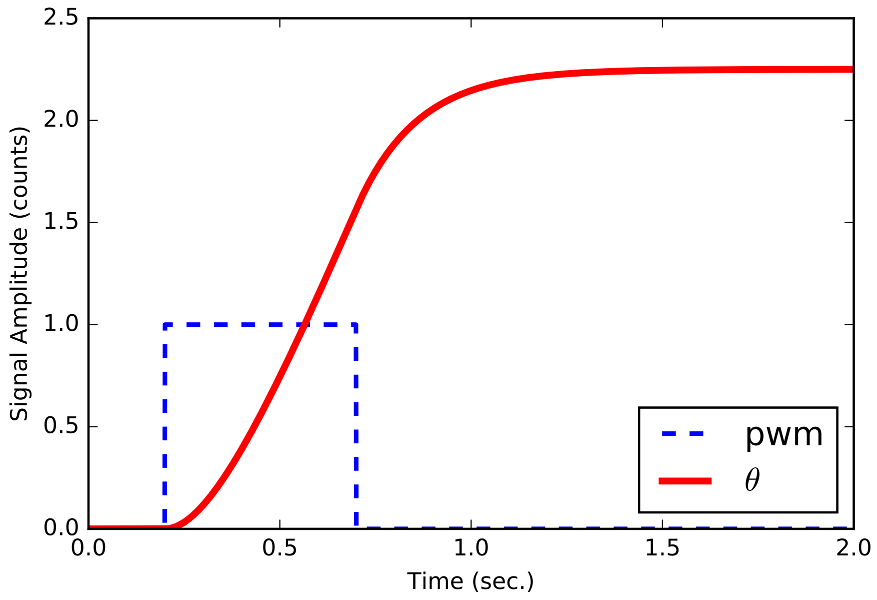

Figure 4: Example pulse response for a DC motor

System Identification (Curve Fitting)

Once you have saved several pulse response tests to csv files, pick at least one response to curve fit using `scipy.optimize.fmin`. Remeber that this is best done by creating two helper functions: `mymodel` and `mycost`. The `mymodel` function takes a list of unknown transfer function coefficients as its input and returns the $\theta(t)$ pulse response. Within the `mymodel` function, you will want to create a transfer function and call the function `control.forced_response` to find the pulse response.

The cost function `mycost` must take the same list of unknown coefficients as its input and return the scalar cost to be minimized (the sum of the squared errors).

Verification: Modeling Proportional Control

Once you have the transfer function for the motor with the coefficients estimated using `fmin`, find the closed-loop transfer function for proportional control using `control.feedback`. Then find the closed-loop step response. Compare this closed-loop simulation to experimental results. In order to do this, you will need to modify your open-loop pulse test code to perform proportional control step responses. Assuming proportional control with the plant transfer function $G(s)$ and the proportional gain K_p as shown in Figure 5, the closed-loop transfer function `cltf` can be found like so:

```
#assuming num and den are already defined
G = control.TransferFunction(num,den)
Kp = x.xxx
cltf  = control.feedback(Kp*G)
```

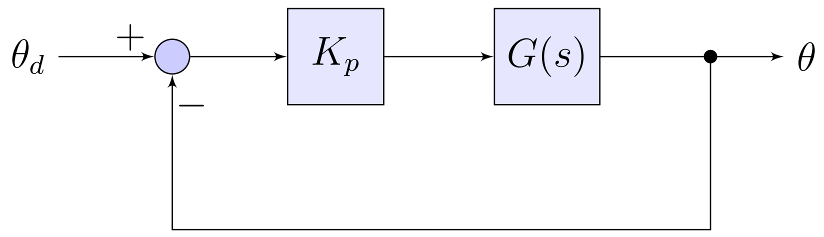

Figure 5: Block diagram of proportional control

Report Requirements

- include the transfer function you derived for the DC motor
  - discuss whatever assumptions are needed to arrive at the transfer function
- include your Arduino code for the open-loop pulse tests; discuss how your code performs real-time dynamic systems experiments
- show a graph overlaying an experimental open-loop pulse test with the prediction from your transfer function model after curve fitting (i.e. the model prediction should be based on transfer function coefficents from `fmin`)
- give the final form of your transfer function, including the estimates of all coefficents
- overlay step response graphs from model and experiment for several different values of K_p
  - do your simulations point toward the right values for "good" choices of K_p?
- answer the following comprehension question:

Comprehension Question

- Describe the open-loop step response of a DC motor system where voltage is the input and θ is the output.