Prelab	Participation	Lab
√+ √ √− 0	√+ √ √− 0	√+ √ √− 0

Name: ______________________

8 Lab: Motor Control

The purpose of control system design is to determine an appropriate input to an ***actuator*** (e.g., voltage to a ***motor***) to obtain a desired (or nominal) output (e.g., motor speed).

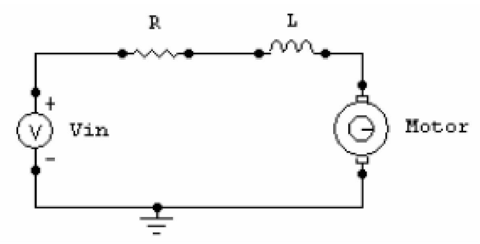
Control systems help satellites to track distant stars, airplanes to follow a desired trajectory, cars to travel at a designated speed, disk-drives to spin at desired angular speeds, and humans to walk, hear, and regulate body temperature.

This lab uses system identification techniques to determine a motor's relevant physical parameters. You will implement a control law to control a motor's orientation and angular speed. You will look at a motor's step response and its transient behavior for various inputs.

8.1 PreLab: Working Model and brainstorming

1. Download the following Working Model simulations from the class website:
   MotorControlWithOnOffAndDeadBand.wm2d
   MotorControlWithKpKi.wm2d
2. Run the Working Model simulations.
   Record results on the <u>Working Model PreLab</u> (feedback control of motor in back of the book).

8.2 Experimental



The system to be controlled is a motor whose rotor (shaft) is attached to a rod. Using $\vec{\mathbf{F}} = m\vec{\mathbf{a}}$ and circuit analysis, the equation relating v_i to motor orientation is as follows (this should be familiar from the homework):

$$\frac{L\,(J_{\text{rod}} + J_m)}{k_m}\,\dddot{\tilde{\theta}} \;+\; \frac{L\,b_m \;+\; R\,(J_{\text{rod}} + J_m)}{k_m}\,\ddot{\tilde{\theta}} \;+\; \left(\frac{R\,b_m}{k_m} \;+\; k_v\right)\dot{\tilde{\theta}} \;=\; \tilde{v}_i$$

In the previous governing ODE, moments of inertia always appear combined as $J_{eff} \triangleq J_{\text{rod}} + J_m$. Knowing $J_{rod} \gg J_m$, approximate the value of J_{eff}.

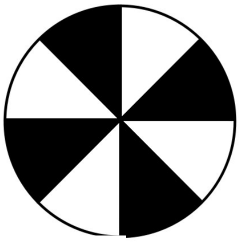
mass of rod $m_{rod} \approx$ ______ kg length of rod $L_{rod} \approx$ ______ m

$$J_{eff} \;\approx\; J_{rod} \;=\; \frac{1}{12}\,m_{rod}\,{L_{rod}}^2 \;\approx\; \rule{5cm}{0.4pt}\ \text{kg m}^2$$

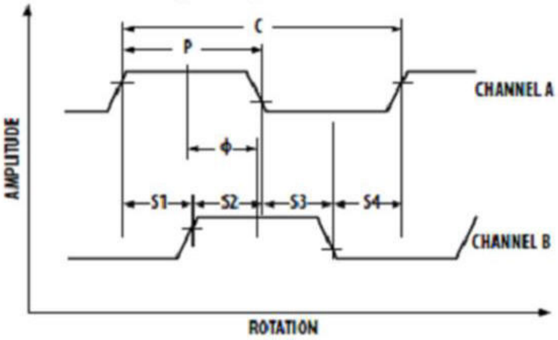
8.3 Equipment

We use several pieces of equipment to measure and control the motor's angular speed, namely, we use an encoder, an Arduino microprocessor, a transceiver, and a computer.[13]

- **Encoder:**
  Our optical quadrature encoder determines our motor's rotational speed by detecting alternating light and dark patterns on a disk. For example, the encoder on the right shows 8 transitions (from light to dark or vice-versa). A quadrature encoder has the ability to detect both angular speed **and** direction. Our encoder has 1000 transitions (500 black sections and 500 white sections) and counts 1000 $\frac{\text{tics}}{\text{rev}}$.



  Shown to the right is the output signal from the encoder to the Arduino microprocessor.



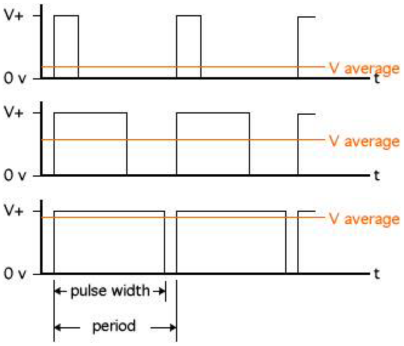
- **Computer:**
  The computer connects to the Arduino microprocessor via a USB cable (Serial Communication). Computer bits (ones and zeros) are transferred between the computer and Arduino microprocessor. You will use the compiled Arduino-specific executable file (Lab8.ino) to communicate between the computer and motor. When you run Lab8.ino, you will be prompted to select proportional or velocity control (choose the appropriate selection for the associated lab question).

- **Arduino UNO microprocessor (the interface between the computer and motor):**
  In this lab, the Arduino will use PWM (Pulse Width Modulation)[14] to vary the average voltage delivered to the motor, which indirectly controlling the motor's speed.

$$\text{PWM} \Rightarrow \text{Average Voltage} \Rightarrow \text{Motor current} \Rightarrow \text{Motor torque} \Rightarrow \alpha \Rightarrow \omega$$

  The frequency of this PWM signal is 30 KHZ.
  The Arduino receives the digital signal from the encoder and counts the transitions from "high" (5 Volts) to "low" (0 Volts) of the signal in 2 milliSecond intervals.
  The on-screen data (i.e., x in the following equation) is in units of $\frac{1\text{ tic}}{2\text{ ms}}$. The equation converting units displayed on the computer screen to the motor's angular speed (in RPM) is

$$\frac{x\text{ tics}}{2\text{ ms}} * \frac{1000\text{ ms}}{1\text{ second}} * \frac{60\text{ sec}}{1\text{ minute}} * \frac{1\text{ rev}}{1000\text{ tics}} = \frac{30\text{ rev}}{1\text{ minute}}$$



- **Motor driver/Arduino Interface Board:**
  The motor driver circuit receives the PWM signal from the Arduino microprocessor and controls the voltage delivered to the motor via the 12 Volt wall adapter.

---

[13] Most motors do not come attached to a rotary encoder and assembled with a encoder, microprocessor, transceiver, and computer.

[14] A PWM signal is on (designated by a "high-bit" 1) or off (designated by a "low-bit" 0) for different intervals.

8.3.1 Data collection

1. If necessary, login to the lab computer. Username: me161student
   Password: 1euler1. Ensure the domain is ENGR
2. Power the Arduino by plugging-in (in order):
   (a). 12 Volt adaptor (between the board and wall socket)
   (b). USB cable (between the board and the computer)
3. From the desktop, navigate to the Lab8 folder and open Lab8.ino
4. Under Tools → ports, select something other than COM1, COM2, or COM3
   (the USB port can be enumerated to anything other than these)
5. Click the magnifying glass button (or type Ctrl+Shift+m) to open the serial monitor
6. On the serial monitor screen, a menu should appear. Select ***position*** or ***velocity*** control.
7. Enter values for k_p, k_i, and k_d and enter a desired (nominal) value (see below).
8. Enter "r" to stop recording data (when sufficient data has been collected).
9. Plot the data (e.g., using Excel, MATLAB®, or PlotGenesis) and determine ζ, ω_n, etc.
   For the yAxis, convert encoder counts to radians by multiplying by $\frac{2\,\pi}{1000}$
   For the xAxis, convert to seconds by dividing by 500 (i.e., each tick is 2 ms)
   Note: Use the oscilloscope to ensure $Vcc = 12$ volts.
10. Email the data files and/or graphs to yourself and your group members.
11. Ensure the power to the board is off and the setup is neat for the next lab.

8.3.2 Experimental determination of ζ and ω_n

Using an input step response of $\theta_{\text{desired}} = 10$ rad and a position PD control law of

$$\widetilde{v}_i \;=\; -k_p\,\widetilde{\theta} \;-\; k_d\,\dot{\widetilde{\theta}} \qquad \text{with } k_p = 1 \text{ and } k_d = 0.01$$

experimentally determine values for ω_n and ζ (assume $L \approx 0$ from here on).

Result:

$\omega_n =$ $\frac{\text{rad}}{\text{sec}}$ $\zeta =$ noUnits

8.3.3 Analytical expressions for ζ and ω_n in terms of k_p, k_d, ...

Using the governing ODE and previous PD control law, write the equations for ω_n and ζ as functions of k_d and k_p (do this symbolically - without numbers).

Result:

$\omega_n =$ $\zeta = \frac{1}{2}$

8.3.4 System identification for motor armature resistance and motor damping

Determine the value for the motor resistance R and motor's damping b_m for the motor with a rod attached. (Note: The last lab showed $k_m = k_v \approx 0.041 \, \frac{\text{N m}}{\text{Amp}}$.)

Result:

$R =$ Ω $\qquad b_m =$ N m sec

8.3.5 Position Control

Analytically calculate values for k_p and k_d corresponding to an overshoot of 0.257 and settling time of 1.75 sec. How confident are you that this will work? [1=doubtful 2 3 4 5=very confident]

Result:

$k_p =$ $\qquad k_d =$

Try these values for k_p and k_d on the actual system. Graph the data and show your lab TA your graph. Calculate the experimental overshoot M_p and settling time t_{settling}.

Result:

$M_p =$ $\qquad t_{\text{settling}} =$

How do the actual values of M_p and t_{settling} compare to the desired maximum overshoot of 0.257 and settling time of 1.75 sec. Discuss potential sources of error, e.g., in the governing ODE or experimental data acquisition.

Result:

What is the fastest settling time you can get with critical damping and what are the corresponding values for k_p and k_d? [Hint: Pick a k_p value (5-10) and solve for k_d so the system is critically damped. Then keep trying!]

Result:

$t_{\text{settling}} =$ $\qquad$ $k_p =$ $\qquad$ $k_d =$

Why can't you get a faster settling time?

Do you get any steady-state error? If so, how much and what do you think causes it?

Write out a new PID control law (with integral control) and then try various values for k_i on the actual system. What happens? How does integral control eliminate steady-state error? Do you have to be careful with your value for k_i?

$$\widetilde{v}_i =$$

8.3.6 Velocity Control

Note: The velocity control on the microcontroller rounds the target velocity you give it to increments of $\frac{2pi}{5} \approx 1.26\frac{rad}{sec}$. This number arises from the combination of running the control software at 500 Hz and having 500 gaps in the encoder.

Implement velocity PI control. What is the governing equation when using PI control?

$$= 0$$

Find k_p and k_i to give a peak of 30 $\frac{\text{rad}}{\text{sec}}$ at $t = 2$ sec to a 24 $\frac{\text{rad}}{\text{sec}}$ step. Try to get the motor to spin at the desired rate of 24 $\frac{\text{rad}}{\text{sec}}$.

$k_p =$ $\qquad$ $k_d =$