GEORGIA INSTITUTE OF TECHNOLOGY

SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 4550 — Control System Design — Fall 2017

Lab #6: DC Motor Position Control

Contents

1 Background Material

1.1 Introductory Comments

The objective of this lab is to implement a state-space integral control algorithm on a microcontroller, in order to perform position control. We will leverage the reduced-order physics-based model of the DC motor plant; the coefficient matrices of this model will be needed for controller design and implementation, so we will use approximate values inferred from measured input-output data. We will utilize plant model knowledge and the computational capability of the microcontroller to perform the numerical processing required by the state-space integral control algorithm, including the real-time approximate solution of differential equations governing the estimator and regulator subsystems of the controller; integrator anti-windup compensation will be used to avoid the poor transient response that can arise when actuator saturation occurs.

1.2 Plant Modeling

We typically require two different models of the same DC motor plant. The *simulation model* has three state variables (position, speed and current) whereas the *design model* has only two state variables (position and speed). We use the simulation model to represent the plant for simulation purposes, but we use the design model to represent the plant for controller design purposes. We have this option since the dynamics being neglected by the design model—the voltage to current transient response—is both stable and fast.

You should review both the simulation model and the design model from posted lecture notes. The simulation model will be explicitly used in Lab 6 simulation code, whereas the design model will be implicitly used in Lab 6 for controller design. In order to prepare for the controller design calculations that follow, recall that the design model may be expressed in the form

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & -\alpha \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ \beta \end{bmatrix} (u(t) - w(t))$$
$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t)$$

where α and β are load-dependent constant parameters to be determined using a parameter identification procedure. The coefficient matrices A, B and C are the point of departure for design.

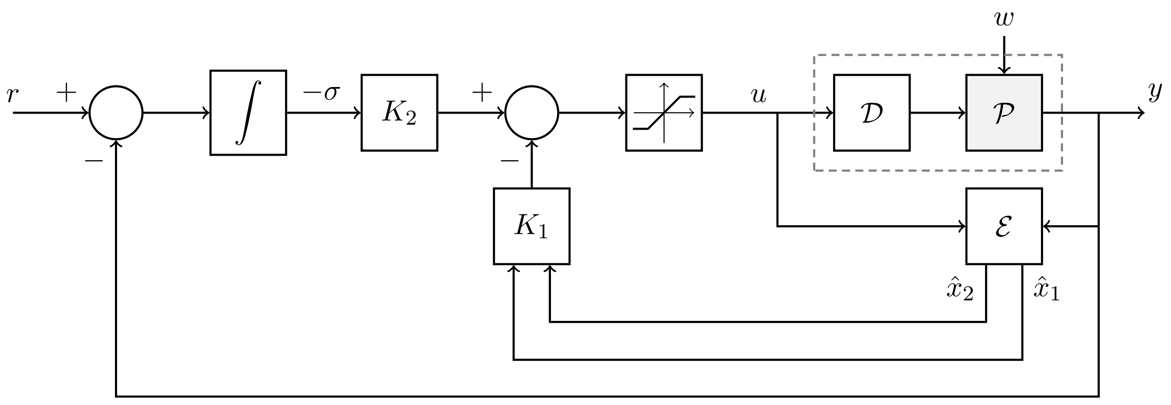
1.3 Controller Design

The general theory of state-space integral control has been summarized in some notes posted to tsquare, so *you should have read through those notes prior to attempting this lab*. Critical steps in applying those notes to the control system design problem at hand are highlighted below. Following the design-by-emulation concept, often referred to as "indirect" digital design, we will first design a suitable continuous-time position controller, and then discretize that controller in a final step in order to obtain a corresponding discrete-time position controller.

The controller is implemented in C with GPIO enable signals (Labs 1–2), it operates periodically with period established by a timer interrupt (Lab 3), the input signal of the plant is a pulse-width modulated voltage and the output signal of the plant consists of quadrature encoder pulses (Lab 5). Hence, this lab integrates almost all that we have learned so far this semester.

1.3.1 Controller Structure

The structure of the controller we will be using in this lab is shown in the block diagram below. The shaded block labeled $\mathcal{P}$ represents our DC motor plant, which consists of an actuator (the drive circuit), a motion system (the rotor) and a sensor (the optical encoder); all other unshaded blocks represent embedded code. The block labeled $\mathcal{D}$ represents the duty cycle generator; it receives a voltage command u and translates that into a duty cycle such that the average voltage applied by the drive circuit will be equal to u. Together, $\mathcal{P}$ and $\mathcal{D}$ constitute a generalized plant. The saturation block accounts for the limits on applied motor voltage inherited as a consequence of the fixed power supply voltage. The external torque disturbance, interpreted as an equivalent external voltage disturbance, is denoted by w. The measured position is denoted by y.



In the inner loop of the controller, the estimator—the block labeled $\mathcal{E}$—generates the scalar signals $\hat{x}_1$ and $\hat{x}_2$, which are estimates of the measured position y and the unmeasured speed $\dot{y}$, respectively. The estimator uses the pre-saturated command voltage as a proxy for the average voltage actually applied to the motor, in order to eliminate the need for any direct measurement of motor voltage. In the outer loop of the controller, the scalar signal σ is generated by integrating the error between the measured position y and its commanded value r. The gain matrices of the estimator (L) and both regulator loops (K_1 and K_2) are selected so as to ensure stabilization of the overall system, and the error integrator guarantees effectively zero steady-state error between the measured position and its commanded value.

1.3.2 Analog Controller Design

From the design model shown above, it is clear that

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -\alpha \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \beta \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

The stability of the estimator subsystem is determined by the matrix

$$A - LC = \begin{bmatrix} 0 & 1 \\ 0 & -\alpha \end{bmatrix} - \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} -L_1 & 1 \\ -L_2 & -\alpha \end{bmatrix}.$$

If we want the actual characteristic polynomial

$$\det\left(sI - (A - LC)\right) = s^2 + (L_1 + \alpha)\, s + (L_1\alpha + L_2)$$

to match the desired characteristic polynomial

$$(s + \lambda_e)^2 = s^2 + 2\lambda_e s + \lambda_e^2$$

having two roots at $s = -\lambda_e$, then the estimator gains must be

$$L_1 = 2\lambda_e - \alpha, \quad L_2 = \lambda_e^2 - 2\alpha\lambda_e + \alpha^2.$$

It was possible to solve the estimator eigenvalue assignment problem in this case because our sensor choice results in an *observable* system. As expected, L_1 and L_2 depend on plant parameters (α, β) and design parameter λ_e. A large λ_e would lead to a fast estimator, but also to large L_1 and L_2 which could result in problems relating to sensor noise amplification; therefore, selection of λ_e involves a trade-off between rate of response and signal/noise ratio.

The regulator subsystem is modeled by the matrices

$$\mathcal{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\alpha & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad \mathcal{B} = \begin{bmatrix} 0 \\ \beta \\ 0 \end{bmatrix}, \quad \mathcal{K} = \begin{bmatrix} K_{11} & K_{12} & K_2 \end{bmatrix}$$

and its stability is determined by the matrix

$$\mathcal{A} - \mathcal{B}\mathcal{K} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\alpha & 0 \\ 1 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 \\ \beta \\ 0 \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} & K_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -\beta K_{11} & -\alpha - \beta K_{12} & -\beta K_2 \\ 1 & 0 & 0 \end{bmatrix}.$$

If we want the actual characteristic polynomial

$$\det\left(sI - (\mathcal{A} - \mathcal{B}\mathcal{K})\right) = s^3 + (\alpha + \beta K_{12})\, s^2 + (\beta K_{11})\, s + (\beta K_2)$$

to match the desired characteristic polynomial

$$(s + \lambda_r)^3 = s^3 + 3\lambda_r s^2 + 3\lambda_r^2 s + \lambda_r^3$$

having three roots at $s = -\lambda_r$, then the regulator gains must be

$$K_{11} = \frac{1}{\beta} 3\lambda_r^2, \quad K_{12} = \frac{1}{\beta}(3\lambda_r - \alpha), \quad K_2 = \frac{1}{\beta}\lambda_r^3.$$

It was possible to solve the regulator eigenvalue assignment problem in this case because our actuator choice results in a *controllable* system. As expected, K_{11}, K_{12} and K_2 depend on plant parameters (α, β) and design parameter λ_r. A large λ_r would lead to a fast regulator, but also to large K_{11}, K_{12} and K_2 which could result in problems relating to actuator saturation limits; therefore, selection of λ_r involves a trade-off between rate of response and control effort.[1]

Once the gain matrices have been determined, all that would remain for analog control implementation would be to construct an op-amp circuit capable of implementing the feedback law

$$\begin{aligned} u(t) &= -K_1 \hat{x}(t) - K_2 \sigma(t) \\ \dot{\hat{x}}(t) &= A\hat{x}(t) + Bu(t) - L\left(C\hat{x}(t) - y(t)\right) \\ \dot{\sigma}(t) &= y(t) - r(t). \end{aligned}$$

Instead, for many reasons, we prefer to approximate this control algorithm on a microcontroller.

1.3.3 Digital Controller Design

By definition, analog controllers are implemented by analog circuits and digital controllers are implemented by digital circuits. Analog controllers provide corrective actions on a continuous-time basis whereas digital controllers provide corrective actions on a discrete-time basis. If we want to implement an existing analog controller on an embedded microcontroller chip, the analog controller must first be discretized to generate a corresponding digital controller. It is not possible to perform *controller discretization* in an exact way, since a discrete-time system response cannot match a continuous-time system response on a continuous-time basis.[2]

For this lab, we will be content to employ the simplest of all discretization schemes, the forward Euler method. This simple method relies on the approximation

$$\dot{x}(kT) \approx \frac{x(kT + T) - x(kT)}{T}$$

where x denotes an arbitrary signal, T denotes a fixed sampling period and k represents a discrete-time index. Using this method to approximate derivatives by differences, the previously designed analog controller gets replaced by the corresponding digital controller[3]

$$\begin{aligned} u[k] &= -K_1 \hat{x}[k] - K_2 \sigma[k] \\ \hat{x}[k+1] &= \hat{x}[k] + T\left(A\hat{x}[k] + Bu[k] - L\left(C\hat{x}[k] - y[k]\right)\right) \\ \sigma[k+1] &= \sigma[k] + T\left(y[k] - r[k]\right). \end{aligned}$$

[1]A large λ_r could also cause instability, since our controller design neglects current as a state variable.

[2]It is possible to perform *plant discretization* in an exact way, since a continuous-time system response can match a discrete-time system response at sampling instants. This concept is exploited in so-called “direct” digital design.

[3]The square bracket notation is used to distinguish discrete-time signals from continuous-time signals; the index inside the brackets indicates the control cycle during which the specified signal value is valid.

This digital controller is not equivalent to the analog controller from which it was derived, for the reasons stated above. However, this digital controller will do a good job of emulating the desired analog controller behavior if T is sufficiently small.

To complete the design of our digital controller, we must substitute our plant model matrices (A, B, C) and our selected gain matrices (L, K_1, K_2) into the above equations, we must add the controller saturation effect, and we must exhibit the one-cycle delay introduced by our ISR programming philosophy. The result of these final steps is the computed controller output signal

$$u^*[k] = -K_{11}\hat{x}_1[k-1] - K_{12}\hat{x}_2[k-1] - K_2\sigma[k-1],$$

the saturated controller output signal

$$u[k] = \begin{cases} +U_{\max} & \text{, if } u^*[k] > +U_{\max} \\ -U_{\max} & \text{, if } u^*[k] < -U_{\max} \\ u^*[k] & \text{, otherwise} \end{cases}$$

and the controller state variable update equations

$$\begin{aligned} \hat{x}_1[k+1] &= \hat{x}_1[k] + T\hat{x}_2[k] - TL_1\left(\hat{x}_1[k] - y[k]\right) \\ \hat{x}_2[k+1] &= \hat{x}_2[k] - T\alpha\hat{x}_2[k] + T\beta u[k] - TL_2\left(\hat{x}_1[k] - y[k]\right) \\ \sigma[k+1] &= \sigma[k] + T\left(y[k] - r[k]\right). \end{aligned}$$

These equations have been presented in scalar form to simplify their programming in C.

1.3.4 Anti-Windup Compensation

The implementation of state-space integral control as previously described would not perform well if actuator saturation were to occur. The problem, known as integrator windup, has been described in some notes posted to tsquare; generally speaking, integration of output error is not appropriate during time intervals on which the actuator is saturated, since that would lead to an increasing integrator output without resulting in a larger corrective influence.

One approach to anti-windup compensation is to use conditional integration. With this approach, pure integration is performed whenever saturation is not occurring whereas no integration at all is performed whenever saturation is occurring. Assuming that the control signal will be pre-saturated in code so as to avoid triggering saturation within the actuator itself, the synthesis of the control signal involves three steps: (i) compute the desired control signal $u^*(t)$ in the usual way; (ii) pre-saturate $u^*(t)$ to obtain a value for $u(t)$ that is compatible with actuator saturation limits; (iii) compute the integrator output $\sigma(t)$ by assigning either a zero value or the output error value to the integrator input according to the presence or absence of saturation. Mathematically, this three-step procedure is summarized by

$$\begin{aligned} u^*(t) &= -K_1\hat{x}(t) - K_2\sigma(t) \\ u(t) &= \begin{cases} +U_{\max} & \text{, if } u^*(t) > +U_{\max} \\ -U_{\max} & \text{, if } u^*(t) < -U_{\max} \\ u^*(t) & \text{, otherwise} \end{cases} \end{aligned}$$

where

$$\dot{\sigma}(t) = \begin{cases} 0 & \text{, if } u^*(t) > +U_{\max} \\ 0 & \text{, if } u^*(t) < -U_{\max} \\ y(t) - r(t) & \text{, otherwise} \end{cases}.$$

The recommended digital implementation, including the one-cycle delay, is defined by

$$u^*[k] = -K_1\hat{x}[k-1] - K_2\sigma[k-1]$$

$$u[k] = \begin{cases} +U_{\max} & , \text{ if } u^*[k] > +U_{\max} \\ -U_{\max} & , \text{ if } u^*[k] < -U_{\max} \\ u^*[k] & , \text{ otherwise} \end{cases}$$

where

$$\sigma[k+1] = \begin{cases} \sigma[k] & , \text{ if } u^*[k] > +U_{\max} \\ \sigma[k] & , \text{ if } u^*[k] < -U_{\max} \\ \sigma[k] + T\,(y[k] - r[k]) & , \text{ otherwise} \end{cases} .$$

For a more complete discussion of integrator windup and its compensation, including a detailed example with illustrative simulations, please consult the notes posted to tsquare.

1.4 Simulation of the Positioning System

Control engineers follow a systematic process when working on a development project. The process begins with an understanding of the plant physics, including the derivation of its time-domain dynamic model. Selection of an actuator and sensor come next, where the options under consideration should be limited to those that guarantee existence of feedback gains providing internal stability; such guarantees are established from controllability and observability analyses. The next step is to obtain numerical values for all actuator-plant-sensor parameters appearing in the design model, either by deriving these values from physical principles, by finding these values on data sheets, or by evaluating these parameters using experimental measurements. Once the design model parameters are known, the next step is to compute feedback gains. These feedback gains depend on eigenvalue locations typically selected with transient response specifications in mind, but over-ambitious design choices may lead to large feedback gains and consequent problems due to actuator saturation and sensor noise. Hence, *simulation* is an important next step in the design process, especially for microcontroller-based implementations. Simulation provides a convenient way to explore the influence of eigenvalue specification, parameter uncertainty, sampling frequency, and actuator-sensor quantization. We are now at the simulation stage in this project.

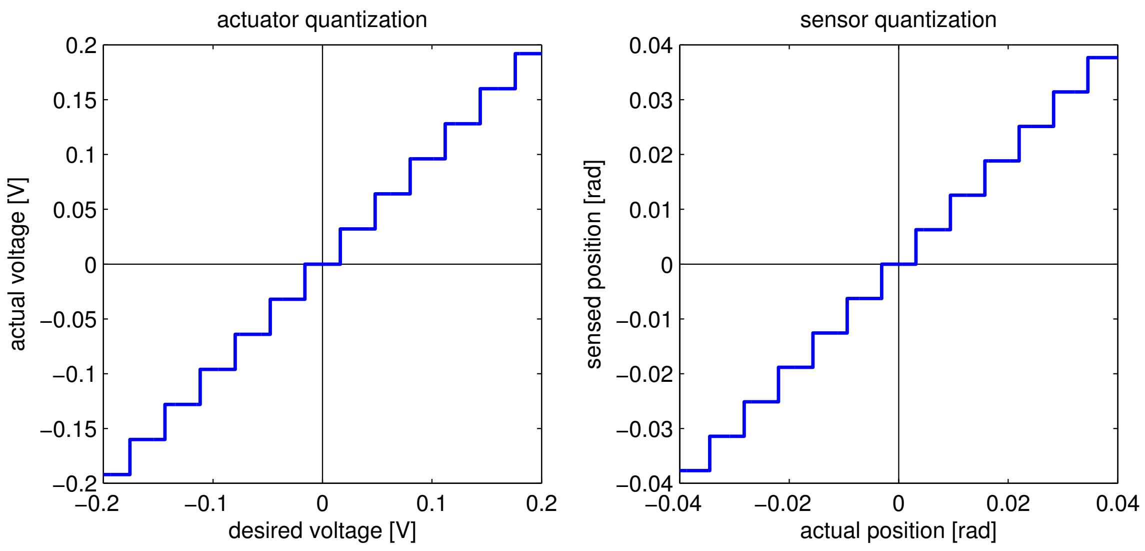
1.4.1 Actuator-Sensor Quantization

Before considering the task of simulating the complete control system, let's first learn how to simulate the effect of signal quantization. Our actuator supplies a voltage using 1500 duty cycles over a ± 24 V range. Our sensor measures a position, using a 250-slot encoder wheel and $4\times$ decoding logic. The resulting discrepancies between the actuator input signal and actuator output signal, and between the sensor input signal and sensor output signal, are computed in the following Matlab code; both input-output relations are graphically displayed in the figure below.

```
% actuator quantization (1500 duty cycles for +/- 24 V)
Qa = 48/1500;
Xa = 0.2;
xa = Xa*(-1:1/1000:1);
ya = Qa*round(xa/Qa);
subplot(121), plot([-Xa Xa],[0 0],'k',[0 0],[-Xa Xa],'k')
hold on, stairs(xa,ya,'LineWidth',1.5), hold off
```

```
axis([-Xa Xa -Xa Xa]), axis square
xlabel('desired voltage [V]')
ylabel('actual voltage [V]')
title('actuator quantization')

% sensor quantization (250 slots per revolution)
Qs = 2*pi/1000;
Xs = 0.04;
xs = Xs*(-1:1/1000:1);
ys = Qs*round(xs/Qs);
subplot(122), plot([-Xs Xs],[0 0],'k',[0 0],[-Xs Xs],'k')
hold on, stairs(xs,ys,'LineWidth',1.5), hold off
axis([-Xs Xs -Xs Xs]), axis square
xlabel('actual position [rad]')
ylabel('sensed position [rad]')
title('sensor quantization')
```



1.4.2 Description of Simulation Code

Now let's consider our main task, which is to simulate the overall system consisting of our continuous-time DC motor plant, with quantized actuator and quantized sensor, under the influence of a discrete-time state-space integral controller commanded by a reference signal. The supplied Matlab code is intended to assist with this task. First you must enter values for the plant parameters. Nominal values from the datasheet of the unloaded motor are provided, but in reality there is always a mismatch between the characteristics of the true plant and those assumed for sake of controller implementation; this phenomenon is captured by adopting a third-order simulation model and a second-order design model, and by assuming that parameters `J` and `F` represent the true plant, but that `Jhat` and `Fhat` are the parameters actually used for controller implementation; you can vary the `*1` factors to explore the robustness of your control system to parameter mismatch. Next you must enter values for the regulator subsystem bandwidth, `lambda_r`, and the estimator subsystem bandwidth, `lambda_e`; typically, we make the estimator eigenvalues faster than the regulator eigenvalues by a factor of four, but you can change the `4*` ratio if desired. Finally, you must enter a value for the sampling period, `T`; 1 ms is a good choice to begin with.

Additional features of this code are as follows: a 24 V power supply is assumed, so the software-generated actuator saturation limits are set to 95% of ±24 V (to avoid current sensing problems, not considered here); the external disturbance torque is assumed to be zero; the overall system is initialized at the zero equilibrium, with external inputs and state variables all equal to zero; the reference input `r` is preset to command one full revolution; the feedback gain matrices `L`, `K1` and `K2` are computed according to the procedure shown earlier; the forward Euler approximation is used to discretize the controller with period `T` and to simulate the plant with time-step `h`; and a one-cycle delay in application of `u` is assumed, consistent with our standard interrupt-based code flow choices made in previous labs.

2 Lab Assignment

2.1 Pre-Lab Preparation

Each individual student must work through the pre-lab activity and prepare a pre-lab deliverable to be submitted *by the beginning of the lab session*. The pre-lab deliverable consists of a brief typed statement, no longer than one page, in response to the following pre-lab activity specification:

1. Read through this entire document, and describe the overall purpose of this week's project.
2. Simulate the control system using the supplied code, in order to provide informed responses to the following questions; use words to convey your observations (don't submit plots).
   (a) How were the formulas for `alpha` and `beta` derived (see code lines 16 and 17)?
   (b) What happens when `Jhat` and `Fhat` are not equal to `J` and `F`?
   (c) What happens when `lambda_r` and `lambda_e` are varied?
   (d) What happens when `T` is varied?
   (e) What happens when `r` is varied?
   (f) How big is the steady-state positioning error?

Please note that it is not essential to write application code prior to the lab session; the point of the pre-lab preparation is for you to arrive at the lab session with a clear understanding of how the plant should respond to the controller we will implement during the lab session.

2.2 Specification of the Assigned Tasks

A preliminary task of any model-based design is to approximate the parameters of the design model, in this case α and β. Use the data files you produced in Lab 5, along with the procedure described in the posted notes on Least-Squares Parameter Identification, to approximate numerical values for α and β. These will be used to calculate the feedback gains required for the position controller you will implement in the tasks specified below.

Once you have determined suitable values for α and β, develop code that implements the integral controller of §1.3.3, including the refinement of §1.3.4; incorporate the possibility to disable or enable the anti-windup compensation, in order to easily make comparisons. An abruptly changing squarewave position reference signal should be generated in code to request periodic transitions between two programmable positions; for context, imagine that your motor is connected to a belt-pulley apparatus that will locate a robotic pick-and-place tool on a linear motion axis in order to automate the assembly of a printed circuit board. The PWM and QEP modules should be configured as in Lab 5. Perform data logging of u (in V) and y (in rad).

2.2.1 Position Control with Small Motion

Set the programmable positions to 0 rad and 2π rad, and request transitions between these positions once per half second; the complete forward-reverse cycle will take 1 second. Assign the regulator parameter λ_r so as to provide the fastest transient response possible without saturating the actuator. Since actuator saturation should not occur in this case, anti-windup compensation need not be enabled. Display u and y in graph windows to demonstrate the system's behavior.

Instructor Verification (separate page)

2.2.2 Position Control with Large Motion

Set the programmable positions to 0 rad and 10π rad, and request transitions between these positions once per second; the complete forward-reverse cycle will take 2 seconds. Use the value of λ_r obtained from the previous case; this choice will induce significant actuator saturation due to the larger motions involved. Observe the system response in graph windows, with and without anti-windup compensation enabled. Generate response plots in Matlab, illustrating how the presence or absence of anti-windup compensation influences the system's response.

Instructor Verification (separate page)

GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 4550 — Control System Design — Fall 2017
Lab #6: DC Motor Position Control

Instructor Verification Page

Lab Section	Begin Date	End Date
L01, L02	October 17	October 24
L03, L04	October 19	October 26

To be eligible for full credit, do the following:

1. Submissions required by each student (one per student)
   (a) Upload your pre-lab deliverable to tsquare before lab session begins on begin date.
   (b) Upload your `main.c` file for §2.2.2 to tsquare before lab session ends on end date.
2. Submissions required by each group (one per group)
   (a) Submit a hard-copy of this verification page before lab session ends on end date.
   (b) Attach to this page the hard-copy plots requested in §2.2.2.

Name #1: ____________________________________

Name #2: ____________________________________

Checkpoint: Verify completion of the task assigned in §2.2.1.

Verified: ________________________ **Date/Time:** ________________________

Checkpoint: Verify completion of the task assigned in §2.2.2.

Verified: ________________________ **Date/Time:** ________________________