School of Engineering
University of California, Merced

ME141 - Control Engineering

Experiment No. Four

Rotary Servo Base - Integration and Position Control

Objectives

1. Familiarize with the DC motor servo system using the Rotary Servo Base Unit.
2. Design of a proportional-derivative (PD) controller for position control of a servo load shaft.
3. Design of a proportional-integral-derivative (PID) controller to track a ramp reference signal.

Background

Integration

The Rotary Servo Base unit utilizes an incremental optical encoder which typically consists of a coded disk, and LED, and two photo sensors. The disk is codedwith an alternating light and dark radial pattern causing it to act as a shutter seen below.

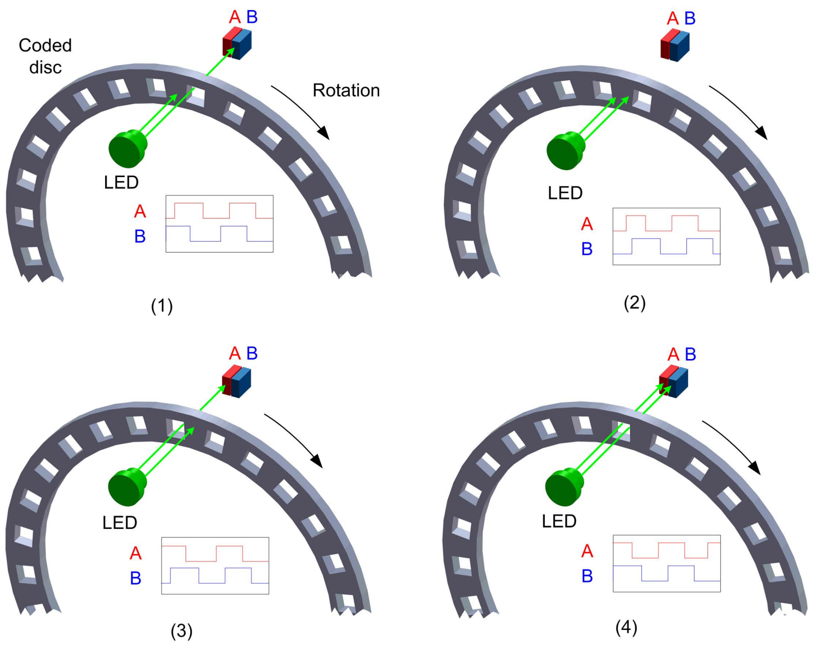

Figure 1: Output of an incremental encoder showing signals A and B when rotating in a clockwise manner

The light emitted by the LED is interrupted by the coding as the disk rotates around its axis. The two photo sensors (A and B) are positioned behind the coded disk sense the light emitted by the LED resulting in A and B signals in four distinct states as seen from the image. The encoder outputs are often referred to as quadrature encoders since the signals are separated in phase by 90^o. The resolution of an encoder corresponds to the number of light and dark patterns on the disk and is given in terms of pulses per revolution (PPR).
In order to make encoder measurements, the encoder must be connected to a counter to count the A and B signals. Then use a decoder algorithm to determine the number of counts and direction of rotation. Three decoding algorithms are used: X1, X2, X4.

- **X1 Decoder**: When an X1 decoder is used, only the rising or falling edge of signal A is counted as the shaft rotates. When signal A leads signal B, the counter is incremented on the rising edge of signal A. When signal B leads signal A, the counter is decremented on the falling edge of signal A. Using an X1 decoder, a 1,024 PPR encoder will result in a total of 1,024 counts for every rotation of the encoder shaft

- **X2 Decoder**: When an X2 decoder is used, both the rising and falling edges of signal A are counted as the shaft rotates. When signal A leads signal B, the counter is incremented on both the rising and falling edge of signal A. When signal B leads signal A, the counter is decremented on both the rising and falling edges of signal A. Using an X2 decoder, a 1,024 PPR encoder will generate a total of 2,048 counts for every rotation of the encoder shaft.

- **X4 Decoder**: When an X4 algorithm is used, both the rising and falling edges of both signals A and B are counted. Depending on which signal leads, the counter will either increment or decrement. An X4 decoder generates four times the number of counts generated by an X1 decoder resulting in the highest resolution among the three types of decoders. Using an X4 decoder, a 1,024 PPR encoder will generate a total of 4,096 counts for every rotation of the encoder shaft.

The angular resolution of an encoder depends on the encoder's PPR and the encoding algorithm used

$$\Delta\theta = \frac{2\pi}{N \times PPR} \tag{1}$$

where N = 1, 2, or 4 corresponds to X1, X2, and X4 decoders respectively. The figure below compares the number of counts generated by each of the X1, X2, and X4 decoders.

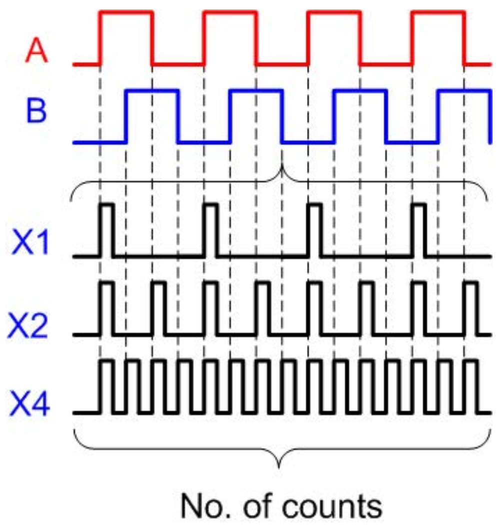

Figure 2: Comparison of number of counts generated by X1, X2, and X4 decoding algorithms

Rotational speeds can also be measured with the encoders to help calculate the velocity by taking the difference between consecutive angle measurements:

$$\omega(k) = \frac{\theta(k) - \theta(k-1)}{h} \tag{2}$$

where $\theta(k)$ represents the k^{th} position measurement sample and h is the sampling interval of the software. The resolution, or ripple, in the velocity measurement is given by

$$\Delta\omega = \frac{\Delta\theta}{h} \tag{3}$$

where ω is rotational speed (rad/s) and $\Delta\theta$ is the resolution of the encoder, and h is the sampling interval.

Position Control

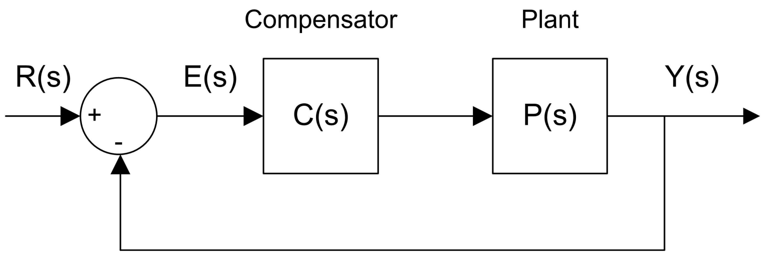

Figure 3: Unity Feedback System

The block diagram above is a general unity feedback system with compensator (controller) $C(s)$ and a transfer function representing the plant, $P(s)$. The measured output, $Y(s)$, is supposed to track the reference signal, $R(s)$, and the tracking has to match to certain desired specifications. The output of this system can be solved for $Y(s)$ to find the closed-loop transfer function:

$$Y(s) = C(s)P(s)(R(s) - Y(s)) \quad \rightarrow \quad \frac{Y(s)}{R(s)} = \frac{C(s)P(s)}{1 + C(s)P(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{4}$$

Using the voltage-to-position transfer function, we can put an integrator $(1/s)$ in series with the speed transfer function to obtain the resulting open-loop voltage-to-load gear position transfer function shown below:

$$P(s) = \frac{K}{s(\tau s + 1)} \tag{5}$$

PD Control

The proportional-derivative (PD) compensator to control the position of the Rotary Servo Base Unit has the following structure:

$$V_m(t) = k_p(\theta_d(t) - \theta_l(t)) - k_d(\frac{d}{dt}\theta_l(t)) \tag{6}$$

where k_p is the proportional control gain, k_d is the derivative control gain, $\theta_d(t)$ is the setpoint or reference load shaft angle, $\theta_l(t)$ is the measured load shaft angle, and $V_m(t)$ is the Rotary Servo Base Unit motor input voltage; the diagram below shows the PD control:

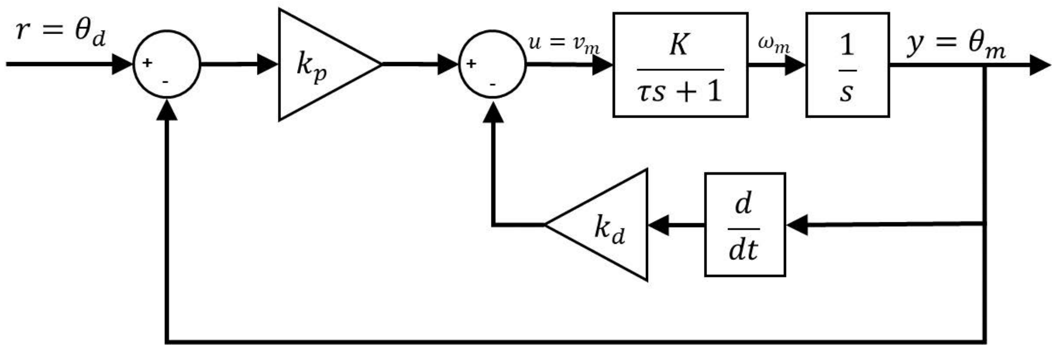

Figure 4: Block diagram of Rotary Servo Base Unit PD position control

We need to find the closed-loop transfer function $\Theta_l(s)/\Theta_d(s)$ By taking the Laplace transform of the former equation, we get

$$V_m(s) = k_p(\Theta_d(s) - \Theta_l(s)) - k_d s\Theta_l(s) \quad \longrightarrow \quad \frac{\Theta_l(s)}{V_m(s)} = \frac{K}{s(s\tau + 1)} \tag{7}$$

Through substitution, we can rewrite the equation as

$$\frac{\Theta_l(s)}{\Theta_d(s)} = \frac{Kk_p}{\tau s^2 + (1 + Kk_d)s + Kk_p} \tag{8}$$

For a step reference input $R(s) = \Theta_d(s) = \frac{R_0}{s}$, the error for the PD control gives:

$$E(s) = \frac{R_0(\tau s + 1 + Kk_d)}{s(\tau s^2 + s + Kk_p + Kk_d s)} \tag{9}$$

PID Control

Adding integral control can help eliminate steady-state error; the proportional-integral-derivative (PID) algorithm is shown below

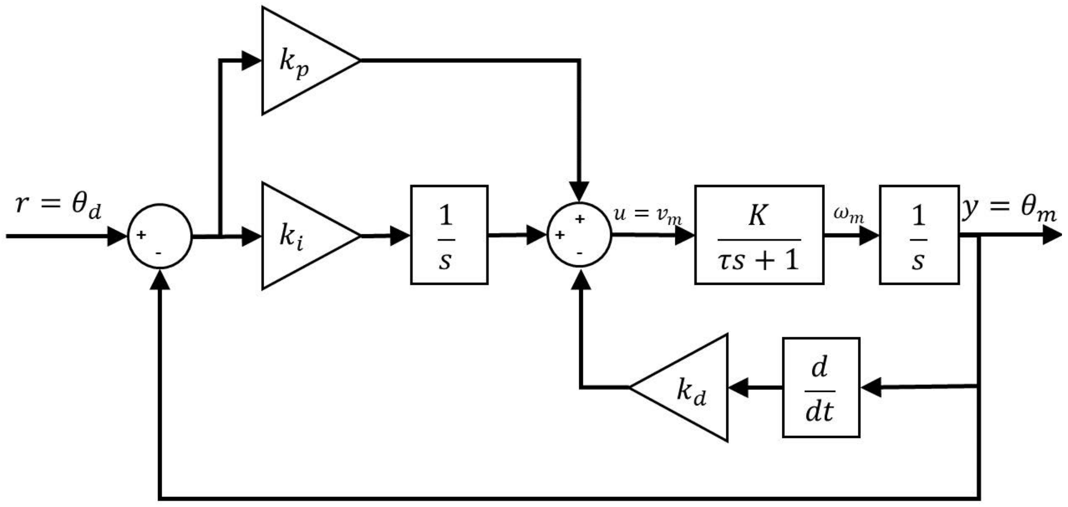

Figure 5: Block diagram of Rotary Servo Base Unit PID position control

Aside from the previous variables in the PD control, k_i is the integral gain. In order to find the closed-loop transfer function $\Theta_l(s)/\Theta_d(s)$ for the closed-loop position of the Rotary Servo Base Unit. Taking Laplace transform (left) and writing the Plant block gives us

$$V_m(s) = (k_p + \frac{k_i}{s}(\Theta_d(s) - \Theta_l(s)) - k_d s \Theta_l(s) \quad \& \quad \frac{\Theta_l(s)}{V_m(s)} = \frac{K}{(\tau s + 1)s} \tag{10}$$

Through substitution, we can rewrite the closed-loop transfer function as:

$$\frac{\Theta_l(s)}{\Theta_d(s)} = \frac{K(k_p s + k_i)}{s^3\tau + (1 + Kk_d)s^2 + Kk_p s + Kk_i} \tag{11}$$

For a ramp reference input (steadily increasing) $R(s) = \Theta_d(s) = \frac{R_0}{s^2}$, the error for the PID control gives:

$$E(s) = \frac{R_0(\tau s + 1 + Kk_d)}{s^3\tau + s^2 + Kk_p s + Kk_i + Kk_d s^2} \tag{12}$$

It takes a certain amount of time for the output response to track the ramp reference with zero steady-state error. This is called the *settling time* and it is determined by the value for the integral gain. In steady-state, the ramp response error is constant, therefore, to design an integral gain the velocity compensation can be neglected. This leaves us with a PI controller:

$$V_m(t) = k + p(\theta_d(t) - \theta_l(t)) + k_i \int (\theta_d(t) - \theta_l(t))dt \tag{13}$$

When in steady-state, the expression can be simplified to

$$V_m(t) = k_p e_{ss} + k_i \int_0^{t_i} e_{ss} dt \tag{14}$$

where the variable t_i is the integration time.
Consider the following time-domain specifications for controlling the position of the Rotary Servo Base Unit load shaft:

- e_{ss} *= 0*
- t_p *= 0.20 s*
- *PO = 5.0 %*

***Refer to the Appendix for additional formulas used in previous labs.*

Procedure

Change the workspace directory to the folder on the Desktop labeled "ME141 Lab #4" > "Rotary Servo Base". We will be exploring the encoder response and position control of the Rotary Servo Base Unit.

Integration

For the Integration section, you will be using $q\_servo\_pos.slx$ file which opens the Simulink block diagram as seen below:

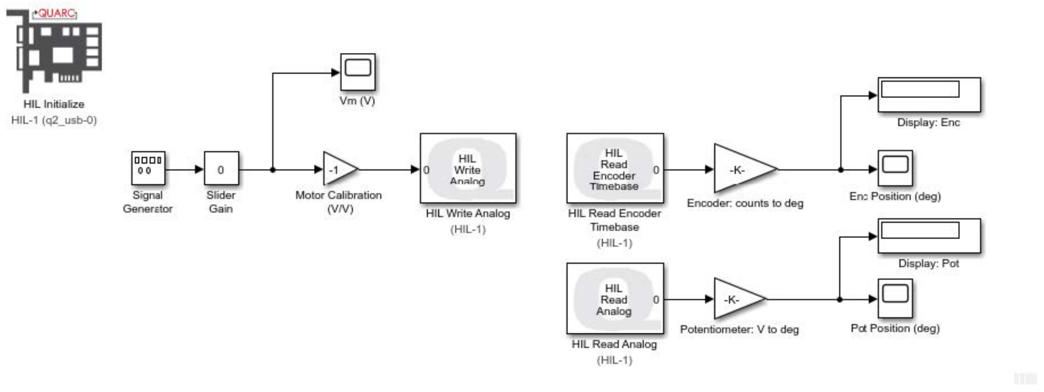

Figure 6: q_servo_pos Simulink Block Diagram

1. Open the file named $q\_servo\_pos.slx$
2. Build the controller and Run
3. Open the "Display" block and begin rotating the disc back and forth.
   – What happens to the count each time you begin the run?
4. Measure the amount of counts the encoder outputs for a full rotation.

Simulation

For the Simulation section, you will be using $s\_servo\_pos\_cntrl$ file which opens the Simulink block diagram as seen below:

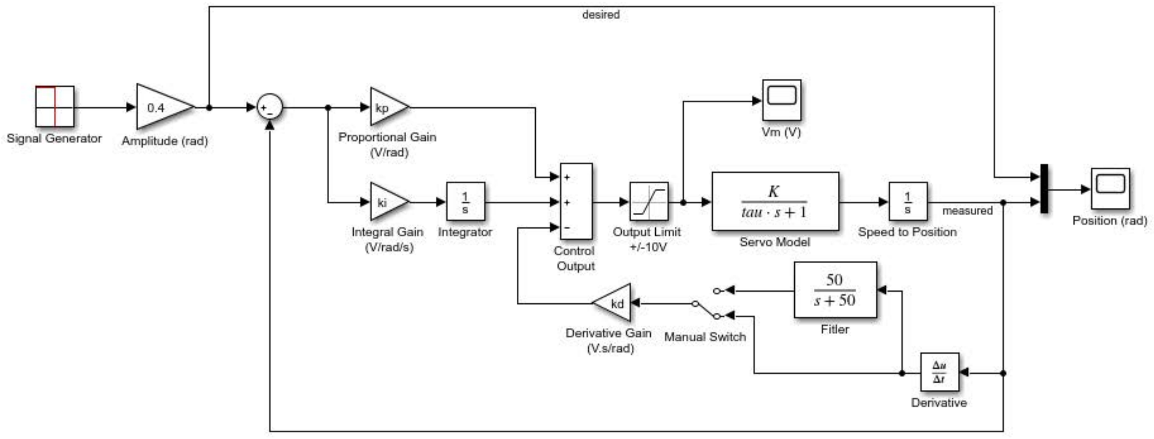

Figure 7: s_servo_pos Simulink Block Diagram

1. Open the script $setup\_servo\_pos\_cntrl$ file and click run
   - Verify that the K and τ values are set to 1.53 rad/s and 0.0217 s, respectively
2. Enter the proportional (k_p) and derivative (k_d) control gains
   - If (k_i) has any value, set it to 0
3. Open $s\_servo\_pos\_cntrl$ file
4. Set the following parameters in the blocks:
   - *Integral* - Integral Gain = 0
   - *Signal Generator* - Signal Type = *square*
   - *Signal Generator* - Amplitude = 1
   - *Signal Generator* - Frequency = 0.4 Hz
5. Set the Amplitude (rad) gain block to 0.4(rad) to generate a step with an amplitude of 45.8^o
6. Set the Manual Switch such that the velocity of the motor load shaft is fed back directly

7. Open the servo position Position (rad) scope and the motor input voltage V$_m$ (V) scope

8. Start the simulation for the PD controller, observe the simulated response and save the following data

   - The scopes will show the ideal PD position response and ideal PD motor input voltage
   - $data\_pos$
     * $data\_pos(:,1)$ - time vector
     * $data\_pos(:,2)$ - set-point
     * $data\_pos(:,3)$ - simulated angle
   - $data\_vm$
     * $data\_vm(:,1)$ - time
     * $data\_vm(:,2)$ - simulated input voltage

9. Enter the integral (k_i) control gain

10. Start the simulation for the PD controller, observe the simulated response and save the $data\_pos$ and $data\_vm$ files

Implementation

For the Implementation section, you will be using $q\_servo\_pos\_cntrl$ file which opens the Simulink block diagram as seen below:

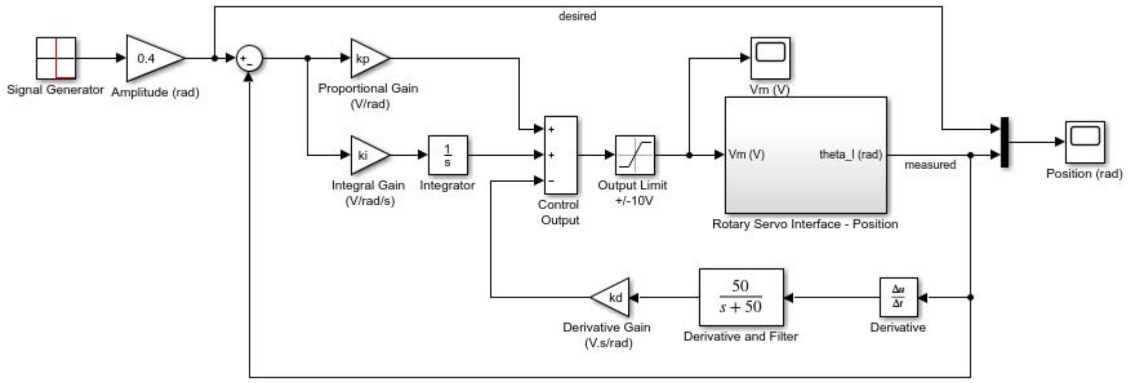

Figure 8: q_servo_pos_cntrl Simulink Block Diagram

1. Open $q\_servo\_pos\_cntrl$ file
2. Set the integral (k_i) control gain to 0 to implement the PD controller
3. Set *Signal Type* in the Signal Generator block to *square* to generate a step reference
4. Set the *Amplitude (rad)* gain block to 0.4 to generate a step with an amplitude of 45.8^o
5. Open the load shaft position scope, *Position (rad)*, and the motor input voltage scope, V_m (V)
6. At the top, open the "$Monitor\ \&\ Tune$" drop-down menu and click on "$Build\ for\ Monitoring$"
   - Turn on the Amplifier and set it to 1x
7. Once the system is built, click *Connect* and then *Run*
8. Stop the run when there is enough data display on the scopes and save the
   - θ_l (rad) - $data\_pos$
     * $data\_pos(:,1)$ - time vector
     * $data\_pos(:,2)$ - set-point
     * $data\_pos(:,3)$ - measured angle

- V_m (V) - $data\_vm$
  - $data\_vm(:,1)$ - time
  - $data\_vm(:,2)$ - simulated input voltage

9. Enter the integral (k_i) control gain to implement the PID controller and repeat steps 5 through 7

Action Items

1. Calculate the resolution of the encoder. (The encoder for the unit had a 1024 PPR and quadrature decoding)
2. Find the velocity estimation for sampling intervals: 0.01s, 0.002s, and 0.001s.
3. For the PD controller, express the control gains k_p and k_d in terms of ω_n and ζ.
4. For the PD controller, calculate the minimum damping ratio and natural frequency required to meet the desired specifications.
5. For the PD controller, calculate the maximum proportional gain that would give the maximum voltage $V_{max} = 10V$ to the motor for the PD controlled system for the reference step of $\theta_d = 45.8°$ starting from $\theta_l = 0$. Ignore the derivative control (k_d=*0*).
6. For the PID controller, find the integral gain k_i value where $V_{max} = 10V$ and steady-state error is eliminated.
7. Calculate the steady-state error, the percent overshoot, and the peak time for the simulation and experimental implementation of the PD controller and the PID controller.
8. Briefly discuss any sources of error, and how they affect your final results.

$FileName$	$Description$
q_servo_pos.slx	Simulink model that applies a voltage to the motor and reads the load gear angle of the Rotary Servo Base Unit
setup_servo_pos_control.m	Run this file only to set up the experiment's Rotary Servo Base Unit's control parameters.
config_servo.m	Returns the configuration-based Rotary Servo Based Unit model specificatio R_m, k_t, k_m, η_g, B_{eq}, J_{eq}, and η_m, the sensor calibration constants K_POT and K_ENC, and the amplifier limits VMAX_AMP and IMAX_AM
d_model_param.m	Calculates the Rotary Servo Base Unit model parameters K and τ based on t device specifications R_m, k_t, k_m, η_g, B_{eq}, J_{eq}, and η_m
calc_conversion_constants.m	Returns various conversion factors
s_servo_pos_cntrl	Simulink file that simlutates a closed-loop PID position controller for the Rotary Servo Base Unit system
q_servo_pos_cntrl	Simulink file that implements a closed-loop PID position controller on the Rotary Servo Base Unit system

Table 1: Matlab files needed for the Rotary Servo Base Unit

Appendix

Percent Overshoot (max): $PO = \frac{100(y_{max} - R_0)}{R_0}$
Peak Time (max): $t_p = t_{max} - t_0$
Percent Overshoot: $PO = 100e^{\left(-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}\right)}$
Error Transfer Function: $E(s) = R(s) - Y(s) = \frac{R(s)}{1+C(s)P(s)}$
Error for Step Response: $e_{ss} = R_0\left(\lim_{s\to 0} \frac{(\tau s+1)s}{\tau s^2+s+K}\right)$