Control Systems Laboratory (EE 3321) — Experiment 7

Motor Transfer Function, Second Order System and PD Controller Design

I. Introduction

This experiment uses the model created in the previous experiment to create a positional transfer function for the QUBE-Servo. Additionally, a controller will be designed and implemented that allows for the precise position control of the motor.

II. First Principles Modeling

The Quanser QUBE-Servo is a direct-drive rotary servo system. Its motor armature circuit schematic is shown in Figure 7.1 and the electrical and mechanical parameters are given in Table 7.1. The DC motor shaft is connected to the *load hub*. The hub is a metal disk used to mount the disk or rotary pendulum and has a moment of inertia of J_h. A disk load is attached to the output shaft with a moment of inertia of J_d .

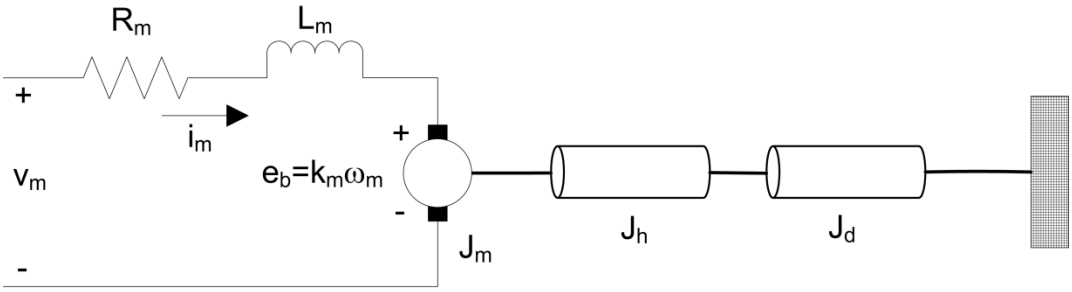


FIGURE 7.1 QUBE-SERVO DC MOTOR AND LOAD

The back-emf (electromotive) voltage $e_b(t)$ depends on the speed of the motor shaft, ω_m , and the back-emf constant of the motor, k_m . It opposes the current flow. The back emf voltage is given by:

$$e_b(t) = k_m \omega(t)$$

Symbol	Description	Value
DC Motor		
R_m	Terminal resistance	6.3 Ω
k_t	Torque constant	0.036 N-m/A
k_m	Motor back-emf constant	0.036 V/(rad/s)
J_m	Rotor inertia	4.0×10^{-6} kg-m^2
L_m	Rotor inductance	0.85 mH
m_h	Load hub mass	0.0087 kg
r_h	Load hub mass	0.0111 m
J_h	Load hub inertia	1.07×10^{-6} kg-m^2
Load Disk		
m_d	Mass of disk load	0.054 kg
r_d	Radius of disk load	0.0248 m

TABLE 7.1: QUBE-SERVO SYSTEM PARAMETERS

Using Kirchoff's Voltage Law, we can write the following equation:

$$v_m(t) - R_m i_m(t) - L_m \frac{di_m(t)}{dt} - k_m \omega_m(t) = 0.$$

Since the motor inductance L_m is much less than its resistance, it can be ignored. Then, the equation becomes

$$v_m(t) - R_m i_m(t) - k_m \omega_m(t) = 0.$$

Solving for $i_m(t)$ the motor current can be found as:

$$i_m(t) = \frac{v_m(t) - k_m \omega_m(t)}{R_m}. \tag{7.1}$$

The motor shaft equation is expressed as

$$J_{eq} \dot{\omega}_m(t) = \tau_m(t), \tag{7.2}$$

where J_{eq} is total moment of inertia acting on the motor shaft and τ_m is the applied torque from the DC motor. Based on the current applied, the torque is

$$\tau_m = k_m i_m(t)$$

The moment of inertia of a disk about its pivot, with mass m and radius r, is

$$J = \frac{1}{2} m r^2 \ . \tag{7.3}$$

III. Second-Order Systems

A. Second-Order Step Response

The *standard second-order* transfer function has the form

$$\frac{Y(s)}{R(s)} = \frac{\omega_n^{\ 2}}{s^2 + 2\zeta\omega_n s + \omega_n^{\ 2}} \ , \tag{7.4}$$

where ω_n is the natural frequency and ζ is the damping ratio. The properties of its response depend on the values of the parameters ω_n and ζ.
Consider a second-order system as shown in Equation 7.4 subjected to a step input given by

$$R(s) = \frac{R_0}{s},$$

with a step amplitude of $R_0 = 1.5$. The system response to this input is shown in Figure 7.2, where the red trace is the output response $y(t)$ and the blue trace is the step input $r(t)$.

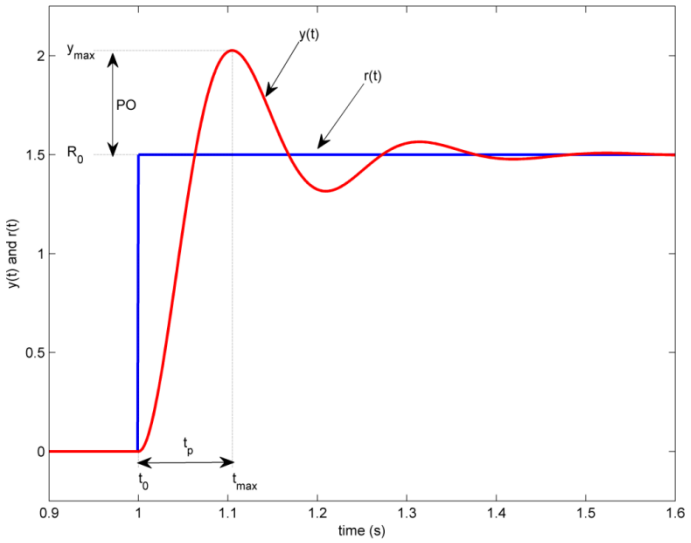


FIGURE 7.2: STANDARD SECOND-ORDER STEP RESPONSE

B. Peak Time and Overshoot

The maximum value of the response is denoted by the variable $y_{\max}$ and it occurs at a time $t_{\max}$. For a response similar to Figure 7.2, the percent overshoot is found using

$$PO = \frac{100(y_{\max} - R_0)}{R_0} \tag{7.5}$$

From the initial step time, t_0, the time it takes for the response to reach its maximum value is

$$t_p = t_{\max} - t_0. \tag{7.6}$$

This is called the peak time of the system.

In a second-order system, the amount of overshoot depends solely on the damping ratio parameter and it can be calculated using the equation

$$PO = 100e^{-\pi\zeta/\sqrt{1-\zeta^2}} \tag{7.7}$$

The peak time depends on both the damping ratio and natural frequency of the system and it can be derived as:

$$t_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}} \tag{7.8}$$

Generally speaking, the damping ratio affects the shape of the response while the natural frequency affects the speed of the response.

C. Unity Feedback

The unity-feedback control loop shown in Figure 7.3 will be used to control the position of the QUBE-Servo.

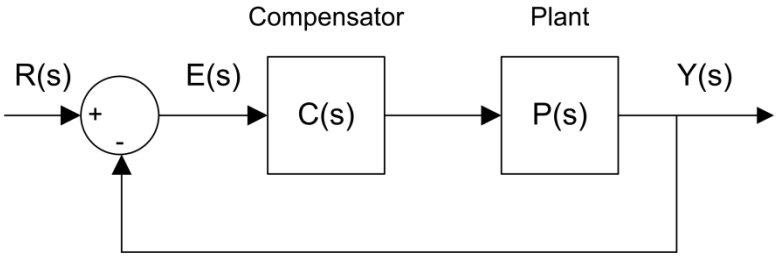


FIGURE 7.3 UNITY FEEDBACK LOOP

The QUBE-Servo voltage-to-position transfer function is

$$P(s) = \frac{\Theta_m(s)}{V_m(s)} = \frac{K}{s(\tau s + 1)}$$

where $K = 23.0$ rad/(V-s) is the model steady-state gain, $\tau = 0.13$ s is the model time constant, $\Theta_m(s) = L\{\theta_m(t)\}$ is the motor / disk position, and $V_m(s) = L\{v_m(t)\}$ is the applied motor voltage. If desired, you can conduct an experiment to find more precise model parameters, K and τ, for your particular servo (e.g. performing the Bump Test Modeling lab).

The controller is denoted by $C(s)$. In this lab, we are only going to use unity feedback therefore

$$C(s) = 1$$

The closed-loop transfer function of the QUBE-Servo position control from the reference input $R(s) = \Theta_d(s)$ to the output $Y(s) = \Theta_m$ using unity feedback as shown in Figure 7.3 is

$$\frac{\Theta_d(s)}{V_m(s)} = \frac{K/\tau}{s^2 + s/\tau + K/\tau} \tag{7.9}$$

IV. PD Control

A. Servo Model

The QUBE-Servo voltage-to-position transfer function is

$$P(s) = \frac{\Theta_m(s)}{V_m(s)} = \frac{K}{s(\tau s + 1)} \tag{7.10}$$

where K = 23.2 rad/(V-s) is the model steady-state gain, τ = 0.13 s is the model time constant, $\Theta_m(s) = L\{\theta_m(t)\}$ is the motor / disk position, and $V_m(s) = L\{v_m(t)\}$ is the applied motor voltage. If desired, you can conduct an experiment to find more precise model parameters, K and τ, for your particular servo (e.g. performing the Bump Test Modeling lab).

B. PID Control

The proportional, integral, and derivative control can be expressed mathematically as follows

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau) d\tau + k_d \frac{de(t)}{dt}. \tag{7.11}$$

The corresponding block diagram is given in Figure 7.4. The control action is a sum of three terms referred to as proportional (P), integral (I) and derivative (D) control gain. The controller Equation 7.11 can also be described by the transfer function

$$C(s) = k_p + \frac{k_i}{s} + k_d s. \tag{7.12}$$

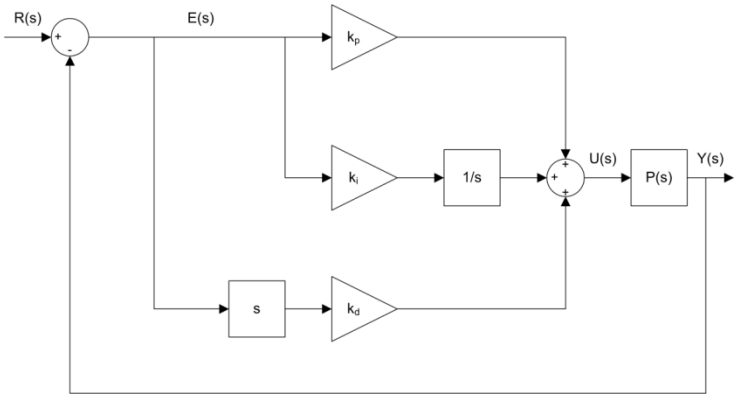


FIGURE 7.4: BLOCK DIAGRAM OF PID CONTROL

The functionality of the PID controller can be summarized as follows. The proportional term is based on the present error, the integral term depends on past errors, and the derivative term is a prediction of future errors.

The PID controller described by Equation 7.11 or Equation 7.12 is an ideal PID controller. However, attempts to implement such a controller may not lead to a good system response for real-world system. The main reason for this

is that measured signals always include measurement noise. Therefore, differentiating a measured (noisy) signal will result in large fluctuations, thus will result in large fluctuations in the control signal.

C. PV Position Control

The integral term will not be used to control the servo position. A variation of the classic PD control will be used: the proportional-velocity control illustrated in Figure 7.5. Unlike the standard PD, only the negative velocity is fed back (i.e. not the velocity of the error) and a low-pass filter will be used in-line with the derivative term to suppress measurement noise. The combination of a first order low-pass filter and the derivative term results in a high-pass filter, H (s), which will be used instead of a direct derivative.

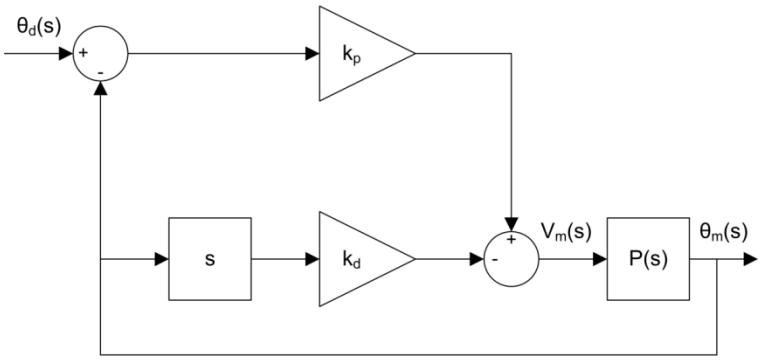


FIGURE 7.5: BLOCK DIAGRAM OF PV CONTROL

The proportional-velocity (PV) control has the following structure

$$u = k_p(r(t) - y(t)) - k_d \dot{y}(t) \tag{7.13}$$

where k_p is the proportional gain, k_d is the derivative (velocity) gain, $r = \theta_d(t)$ is the setpoint or reference motor/load angle, $y = \theta_m(t)$ is the measured load shaft angle, and $u = V_m(t)$ is the control input (applied motor voltage).

The closed-loop transfer function of the QUBE-Servo is denoted $Y(s)/R(s) = \Theta_m(s)/\Theta_d(s)$. Assume all initial conditions are zero, i.e., $\theta_m(0^-) = 0$ and $\dot{\theta}_m(0^-) = 0$, taking the Laplace transform of Equation 1.4 yields

$$U(s) = k_p(R(s) - Y(s)) - k_d s Y(s),$$

which can be substituted into Equation 7.10 to result in

$$Y(s) = \frac{K}{s(\tau s + 1)} k_p(R(s) - Y(s)) - k_d s Y(s)$$

Solving for $Y(s)/R(s)$, we obtain the closed-loop expression

$$\frac{Y(s)}{R(s)} = \frac{Kk_p}{\tau s^2 + (1 + Kk_d)s + Kk_p} \tag{7.14}$$

This is a second-order transfer function. Recall the standard second-order transfer function

$$\frac{Y(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{7.15}$$

V. Experimental Procedure

1. Based on the models already designed in QUBE-Servo Integration and Filtering labs, design a model that applies a 1-3 V 0.4 Hz square wave to the motor and reads the servo velocity using the encoder as shown in Figure 7.6.

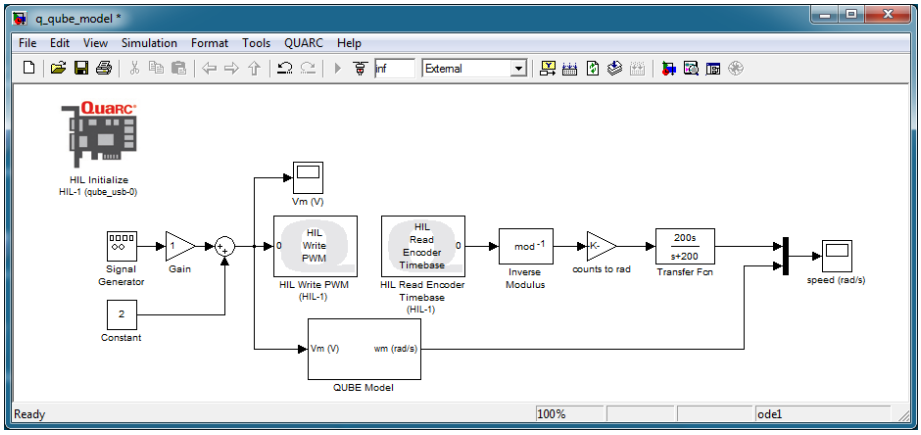


FIGURE 7.6: APPLYING A STEP VOLTAGE AND DISPLAYING MEASURED AND SIMULATED QUBE-SERVO SPEED

Create subsystem called *QUBE-Servo Model*, as shown in Figure 7.6, that contains blocks to model the QUBE-Servo system. Thus using the equations given above, assemble a simple block diagram in Simulink to model the system. You'll need a few Gain blocks, a Subtract block, and an Integrator block (to go from acceleration to speed). Part of the solution is shown in Figure 7.7.

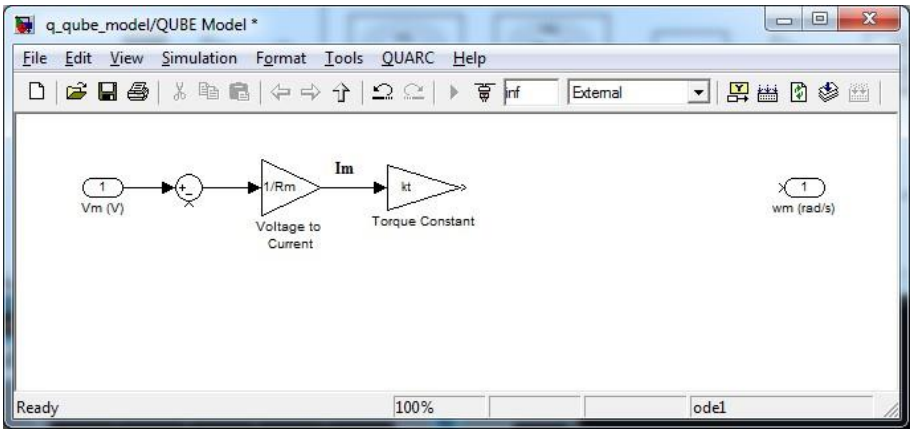


FIGURE 7.7: INCOMPLETE *QUBE-SERVO MODEL* SUBSYSTEM

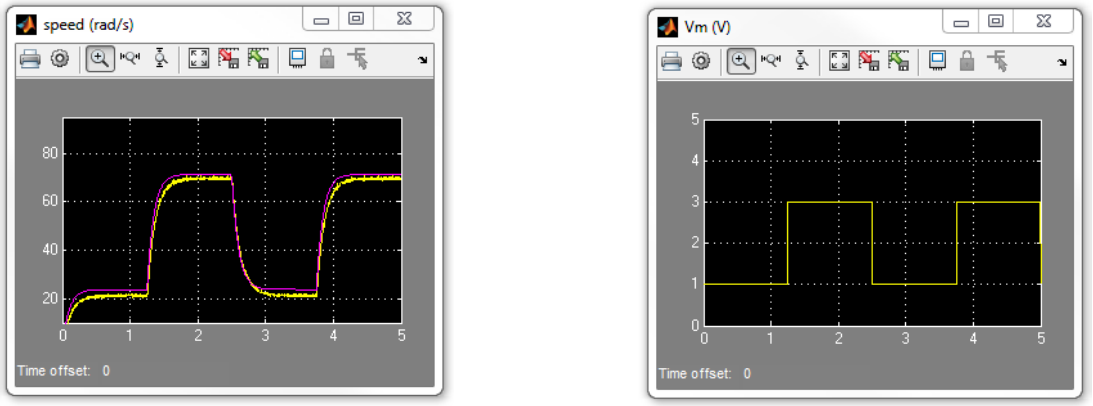
It may also help to write a short Matlab script that sets the various system parameters in Matlab, so you can use the symbol instead of entering the value numerically in the Gain blocks. In the example shown in Figure 7.7, we are using Rm for motor resistance and kt for the current-torque constant. To define these, write a script like:

```
% Resistance

Rm = 8.4;

% Current-torque (N-m/A)

kt = 0.042;
```

1.1 The motor shaft of the QUBE-Servo is attached to a *load hub* and a disk load. Based on the parameters given in Table 7.1, calculate the equivalent moment of inertia that is acting on the motor shaft.

1.2 Design the *QUBE-Servo Model* subsystem as described above. Attach a screen capture of your model and the Matlab script (if you used one).

1.3 Build and run the QUARC controller with your QUBE-Servo model. The scope response should be similar to Figure 7.8. Attach a screen capture of your scopes. Does your model represent the QUBE-Servo well? Explain.



(a) Motor Speed

(b) Motor Voltage

FIGURE 7.8: QUBE RESPONSE

2. Design the Simulink model shown in Figure 7.9. This implements the unity feedback control given in Figure 7.5 in Simulink. A step reference (i.e., desired position or setpoint) of 1 rad is applied at 1 second and the controller runs for 2.5 seconds.

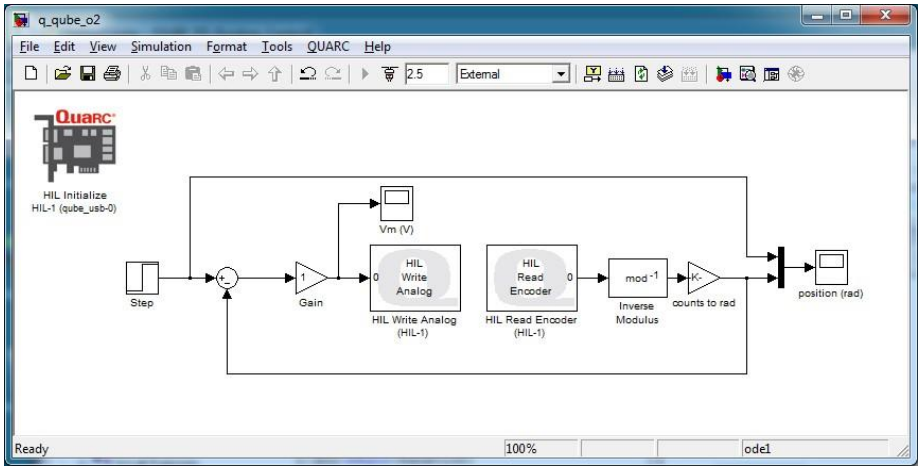
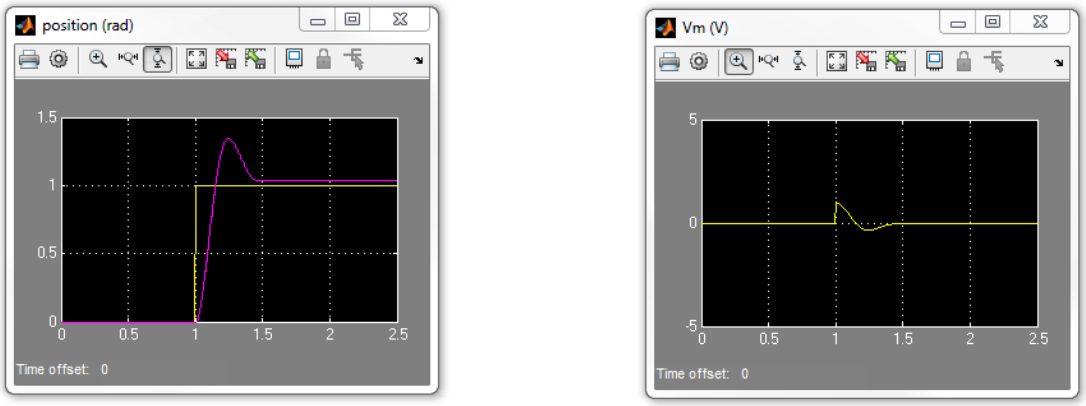


FIGURE 7.9: UNITY FEEDBACK POSITION CONTROL OF QUBE-SERVO

2.1 Given the QUBE-Servo closed-loop equation under unity feedback in Equation 7.15 and the model parameters above, find the natural frequency and damping ratio of the system.

2.2 Based on your obtained ω_n and ζ, what is the expected peak time and percent overshoot?

2.3 Build and run the QUARC controller. The scopes should look similar to Figure 7.10.



(a) Position (b) Voltage

FIGURE 7.10: UNITY FEEDBACK QUBE-SERVO STEP RESPONSE

2.4 Attach the QUBE-Servo position response - showing both the setpoint and measured positions in one scope - as well as the motor voltage.

Hint: For information on saving data to Matlab for offline analysis, see the QUARC help documentation (under QUARC Targets | User's Guide | QUARC Basics | Data Collection). You can then use the Matlab plot command to generate the necessary Matlab figure.

2.5 Measure the peak time and percent overshoot from the response and compare that with your expect results. **Hint:** Use the Matlab *ginput* command to measure points off the plot.

3. Design the Simulink model shown in Figure 7.11. This implements the PV controller with a high-pass filter of 50s/(s + 50). Set the Signal Generator block such that the servo command (i.e., reference angle) is a square wave with an amplitude of 0.5 rad and at a frequency of 0.4 Hz.

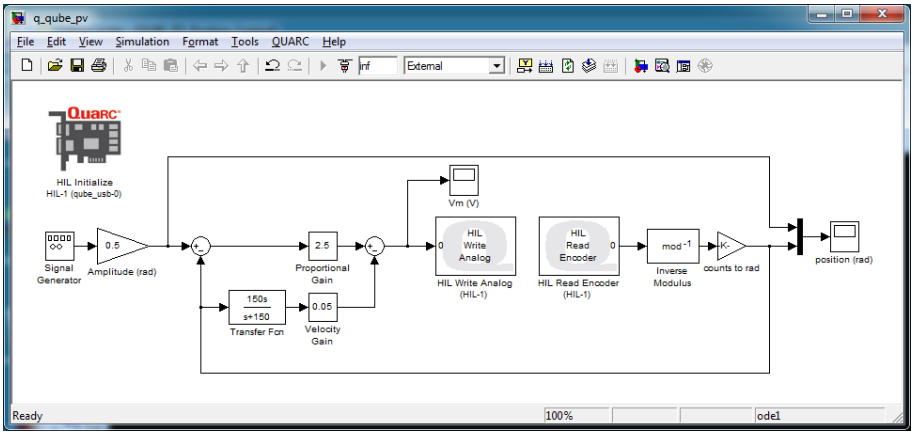
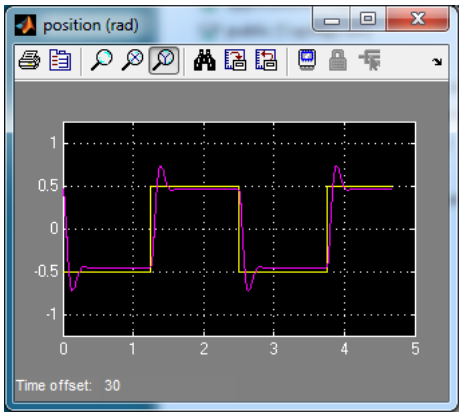
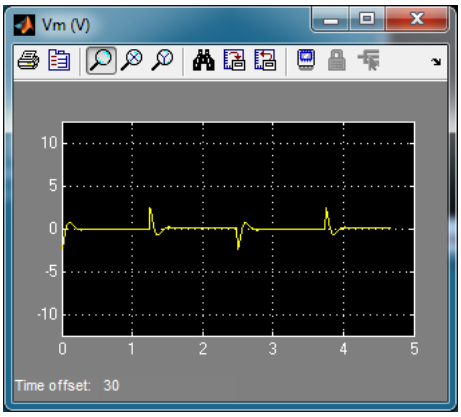


FIGURE 7.11: PV CONTROL ON QUBE-SERVO

3.1 Build and run the QUARC controller. The response should look similarly as shown in Figure 7.12.



(a) Position



(b) Voltage

FIGURE 7.12: QUBE-SERVO PV CONTROL WITH KP = 2.5 AND KD = 0.05

3.2 Set $k_p = 2.5$ V/rad and $k_d = 0$. Keep the derivative gain at 0 and vary k_p etween 1 and 4. What does the proportional gain do when controlling servo position?

3.3 Set $k_p = 2.5$ V/rad and vary the derivative gain k_d between 0 and 0.15 V/(rad/s). What is its effect on the position response?

3.4 Stop the QUARC controller.

3.5 Find the proportional and derivative gains required for the QUBE-Servo closed-loop transfer function given in Equation 7.14 to match the standard second-order system in Equation 7.15. Your gain equations will be a function of ω_n and ζ.

3.6 For the response to have a peak time of 0.15 s and a percentage overshoot of 2.5%, the natural frequency and damping ratio needed are $\omega_n = 32.3$ rad/s and $\zeta = 0.76$. Using the QUBE-Servo model parameters, K and τ, given above in the Background section of this lab (or those you found previously through a modeling lab), calculate the control gains needed to satisfy these requirements.

3.7 Run the PV controller with the newly designed gains on the QUBE-Servo. Attach the position response as well as the motor voltage used.

3.8 Measure the percent overshoot and peak time of the response. Do they match the desired percent overshoot and peak time specifications given in Step 6 without saturating the motor, i.e., going beyond ± 10V?

Hint: Use the Matlab ginput command to measure points off the plot and the equations from the Second-Order Systems Lab.

3.9 If your response did not match the above overshoot and peak time specification, try tuning your control gains until your response does satisfy them. Attach the resulting Matlab figure, resulting measurements, and comment on how you modified your controller to arrive at those results.