Control Pendulum Mini Project

EE 132 Department of Electrical Engineering, UC Riverside

Project Team Member(s)	Alberto Arriaga Felix Various Other Team Members
Final Revision Date	3/ 15/2019
Revision	Version 1.1
Permanent Emails	albertoarriagafelix@gmail.com (Alberto)

Revisions

Version	Description of Version	Author(s)	Date Completed
1.0	First Version	Aberto Arriaga Felix,	6/01/2016
1.1	Last Version	Aberto Arriaga Felix	3/15/2019

Table of Contents

1 Summary of Project

In this project we are working with a Quanser Qube Servo, and. MATLAB Simulink, figure 1 and 2. The Qube Servo is a modular servomotor lab experiment designed for teaching mechatronics and control concepts.

The goal of the project is to develop a model using Simulink and methods based on control theory, for a feedback controller that balances the pendulum in the upright position.

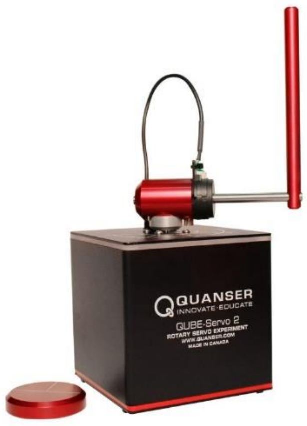


Figure 1 Quanser Qube Servo, the red arm is free to rotate 360 degrees around the motor arm, the motor arm is balance by a dc motor attached to the box that moves horizontally.

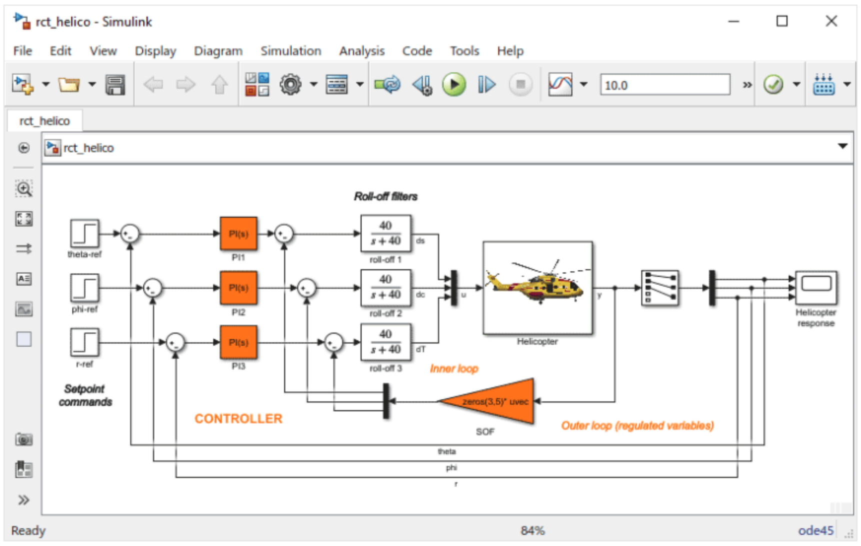


Figure 2 MATLAB Simulink is used to model plant dynamics and design and tune feedback loops and supervisory controllers. Uses simulation models to verify control design and to automatically generate code for rapid prototyping and production.

2 LTI First Order System

In this part we work with an LTI first order system with transfer function of the type;

$$T(s) = \frac{A}{s+A}$$

Where A is a parameter that can be changed experimentally to determine the behavior of our controller. The expression 1/A is the time constant of the system. This system is a low pass filter, as 1/A increases the filter attenuates the signal that passes through it, removing high frequency noise.

Our experimental system controller:
The input in Figure 3 is a square wave of amplitude 1 produced by the Signal Generator. The output is recorded by the scope named "speed".

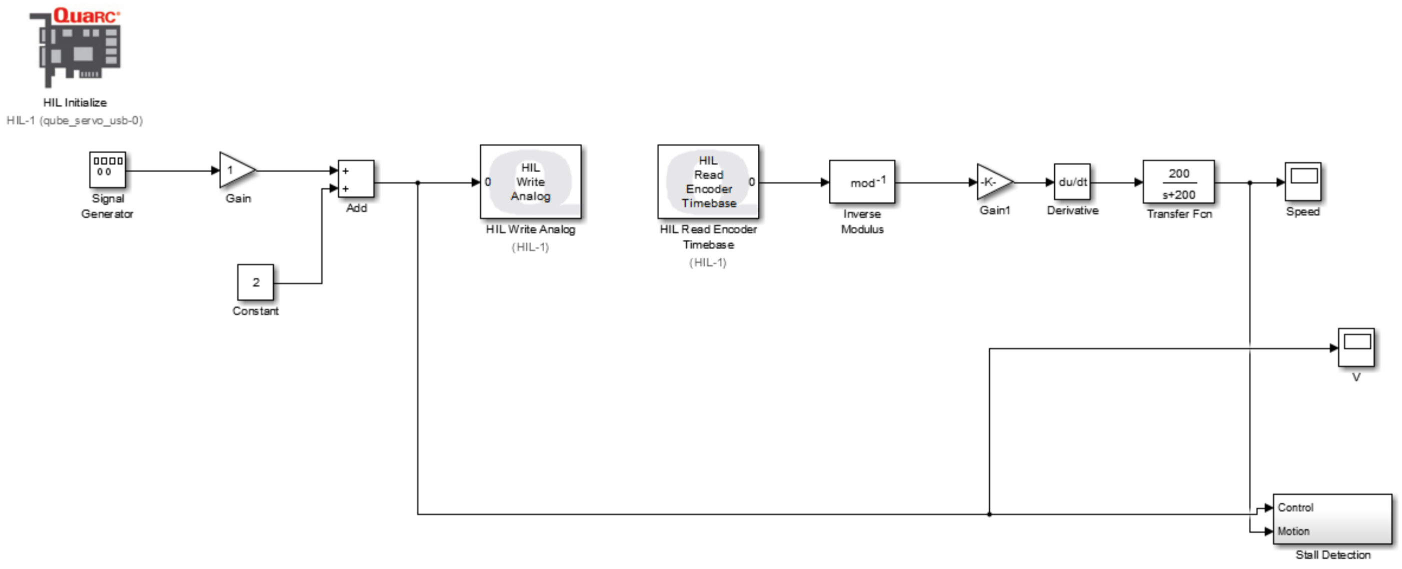


Figure 3 Controller to measure speed.

1. Our first result was the curve obtained without the filter present in the controller shown in figure 2.

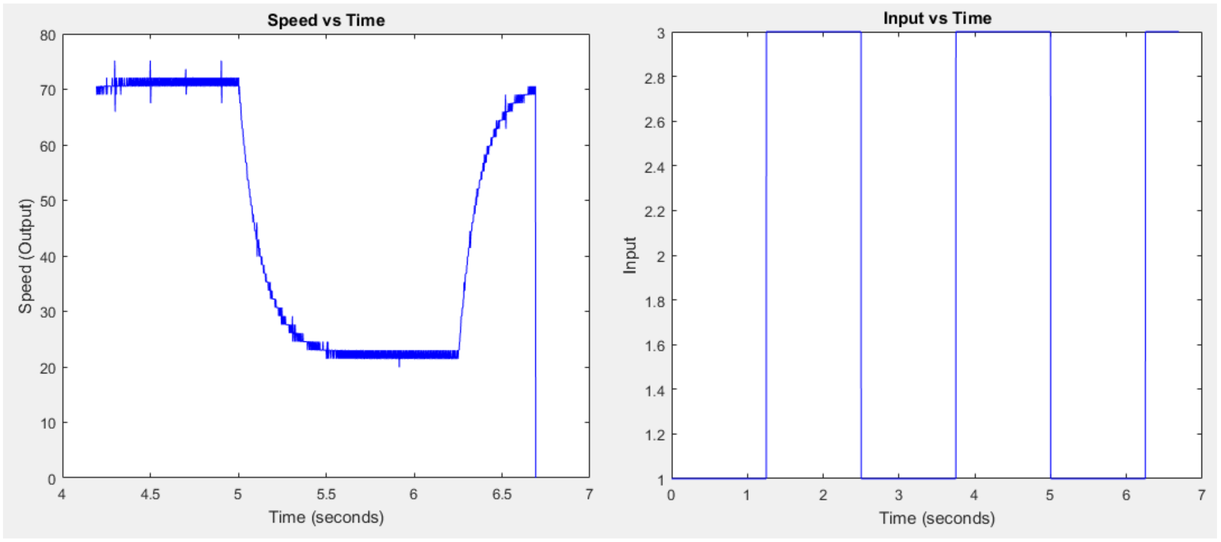


Figure 4 Without Filter

2. The second result was with the filter attached and the A parameter set to 50 as shown in figure 5.

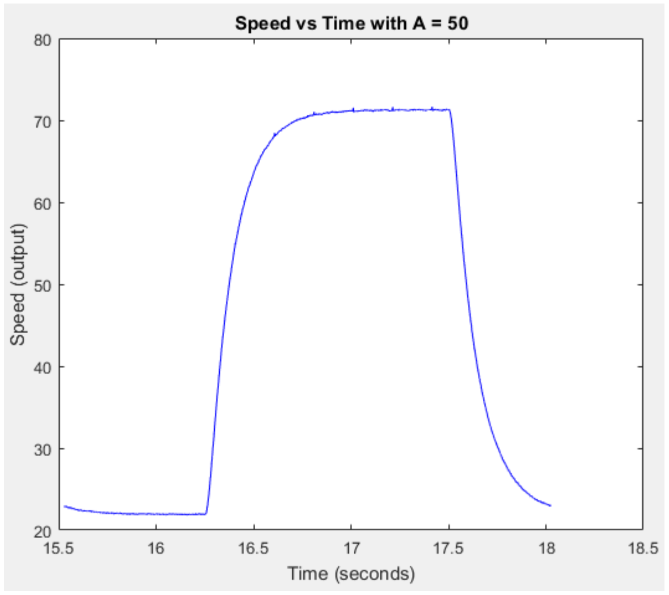


Figure 5 With Filter and A=50

3. The last result we set the filter parameter A to 10 and 200 respectively in figure 6.

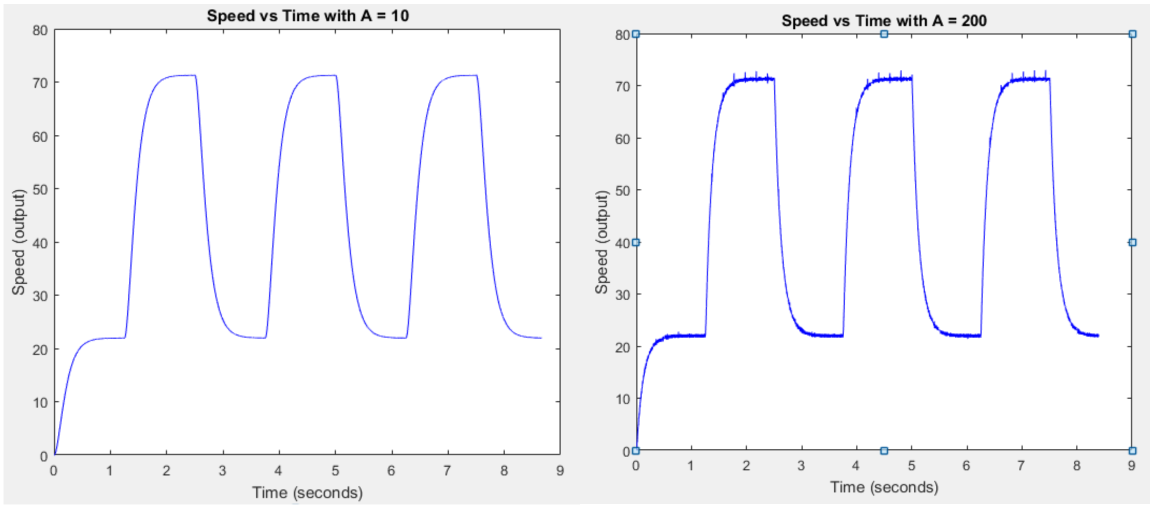


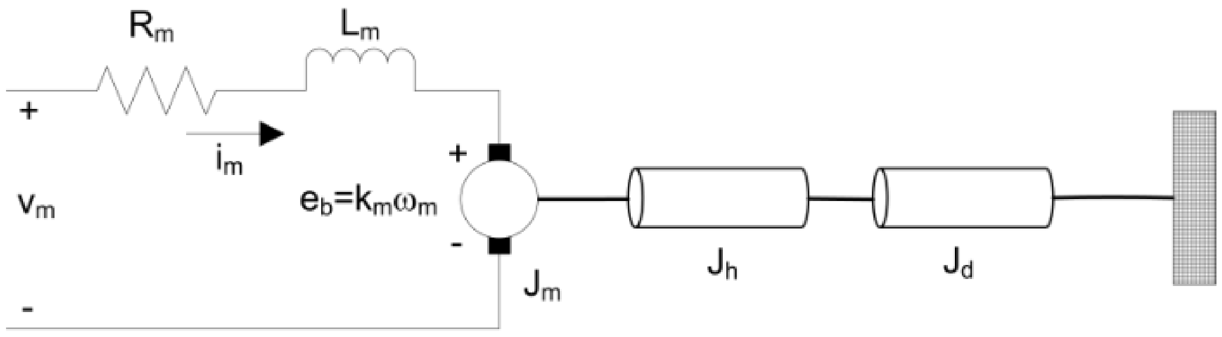
Figure 6 With Filter and A=10, A=200

Analysis:

From the experimental data we can conclude that as the expression $\frac{1}{A}$ increases, that means as A is smaller, the filter attenuates high frequency noise in the output. This can be seen in figure 6. When A is set to 10, the output speed signal is cleaner and without disturbances. When A is set to 200, the output signal shows small ripples as it reaches its peaks.

3 Motor and Metal Disk Modeling

Now we look at the differential equations that models a dc motor connected to a metal disk.



The disk has a moment of inertia J_d and is a mechanical device, the DC motor is an electrical component. The differential equation of the system is;

Eq. 1 $$v_m - R_m i_m - L_m \frac{di_m}{dt} - k_m \omega_m = 0$$

Equation 1 can be simplified;

Eq. 1 $$i_m = \frac{v_m - k_m \omega_m}{R_m}$$

Eq.2 $$J_{eq}\omega'_m = \tau_m$$

Eq. 3 $$\tau_m = k_t i_m$$

Adding Equations 2 and 3;

Eq. 4 $$i_m = \frac{J_{eq}\omega'_m}{k_t}$$

Adding Equations 1 and 4;

Eq. 5 $$\frac{v_m k_t}{J_{eq} R_m} - k_m \omega_m = \omega'_m$$

Where the input of the system is v_m , the electrical voltage applied to the DC motor, and the output of the system is ω_m, the speed of the motor shaft.

If you integrate both sides in the s-domain;

$$\frac{1}{s}\left(\frac{v_m k_t}{J_{eq} R_m} - k_m \omega_m\right) = \omega_m$$

Our experimental system controller:
Figure 7 shows the Simulink model of the controller for our system. The input is a square wave of amplitude 1 produced by the Signal Generator. The output is recorded by the scope named “speed”. The bottom left part of the figure shows the subsystem that represents the differential equation 5 of the theory.

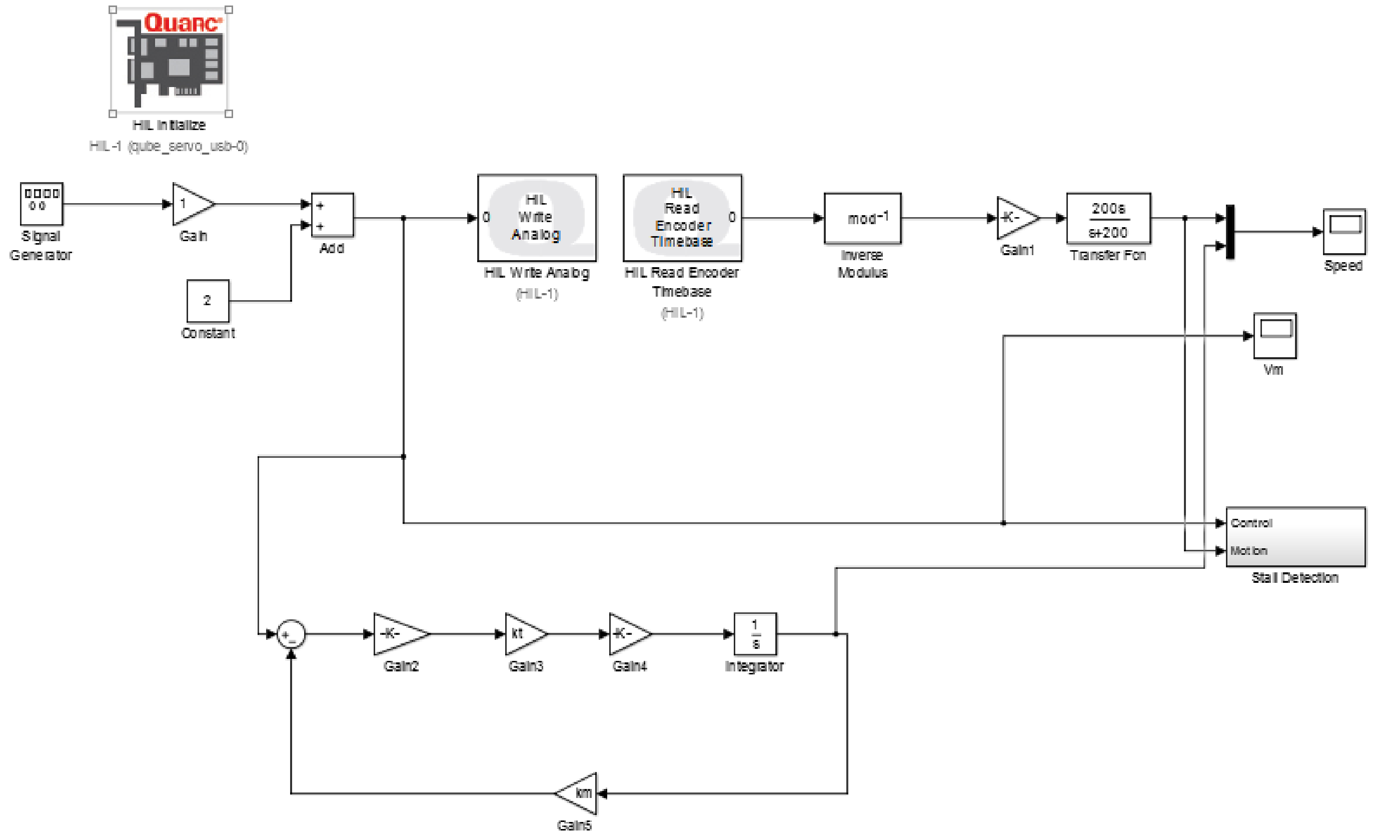


Figure 7 Controller to measure speed.

The Parameter of the subsystem were given by the Matlab code:

```
% Resistance
Rm = 8.4;
% Current-torque (N-m/A)
kt = 0.042;
% Back-emf constant (V-s/rad)
km = 0.042;
% Rotor inertia (kg-m^2)
Jr = 4e-6;
% Hub mass (kg)
mh = 0.0106; % 9 g
% Hub radius (m)
rh = 22.2/1000/2; % diameter 22.2 mm
% Hub inertia (kg-m^2)
Jh = 0.5*mh*rh^2;
% Disc mass (kg)
md = 0.053;
% Disc radius (m)
rd = 49.5/1000/2; % diameter = 49.5 mm
% Disc moment of inertia (kg-m^2)
Jd = 0.5*md*rd^2;
% Equivalent moment of inertia (kg-m^2)
Jeq = Jr + Jh + Jd;
```

1. The result of our Simulink model.

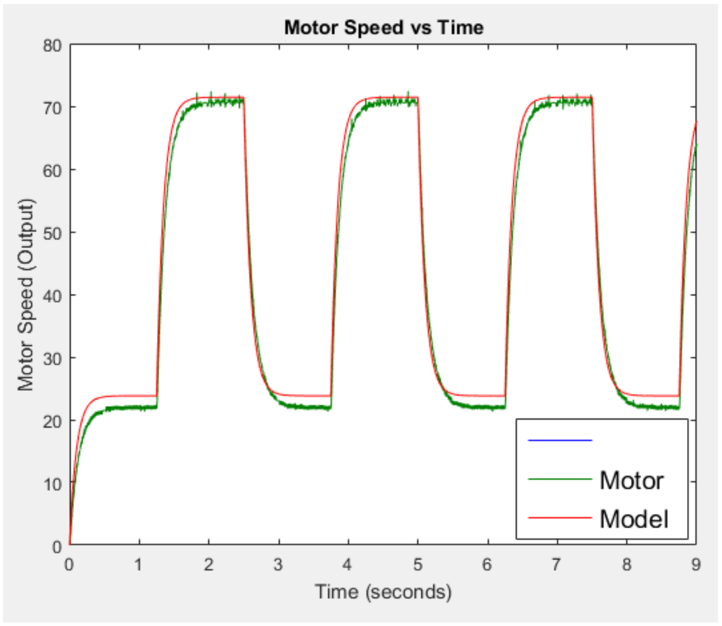


Figure 8 Output of system

Analysis:

In figure 8, the two curves are very close to each other, this is because our model is almost accurate to the response of the motors.

The curves are not the same because in theory we chose to ignore the inductance of the motor, and possibly other parameters of the physical object that we did not model appropriately.

Why is the derivative or $G\,s = s$ not causal?

Because a derivative would depend on a time t + Δt, this would make the system depend on a future time Δt, which makes it non-causal.

Why do we use $\frac{s}{1+\frac{s}{w}} = \frac{sw}{s+w}$ to replace s?

When w is sufficiently large, the response will go to s as $\frac{s}{w}$ goes to 0.

4 Bump Test Modeling

The bump test is a simple test based on the step response of a stable system. A step input is given to the system and its response is recorded. As an example, consider a system given by the following transfer function:

Eq. 1
$$\frac{Y(s)}{U(s)} = \frac{K}{\tau s + 1}$$

The step response is shown in figure 9.

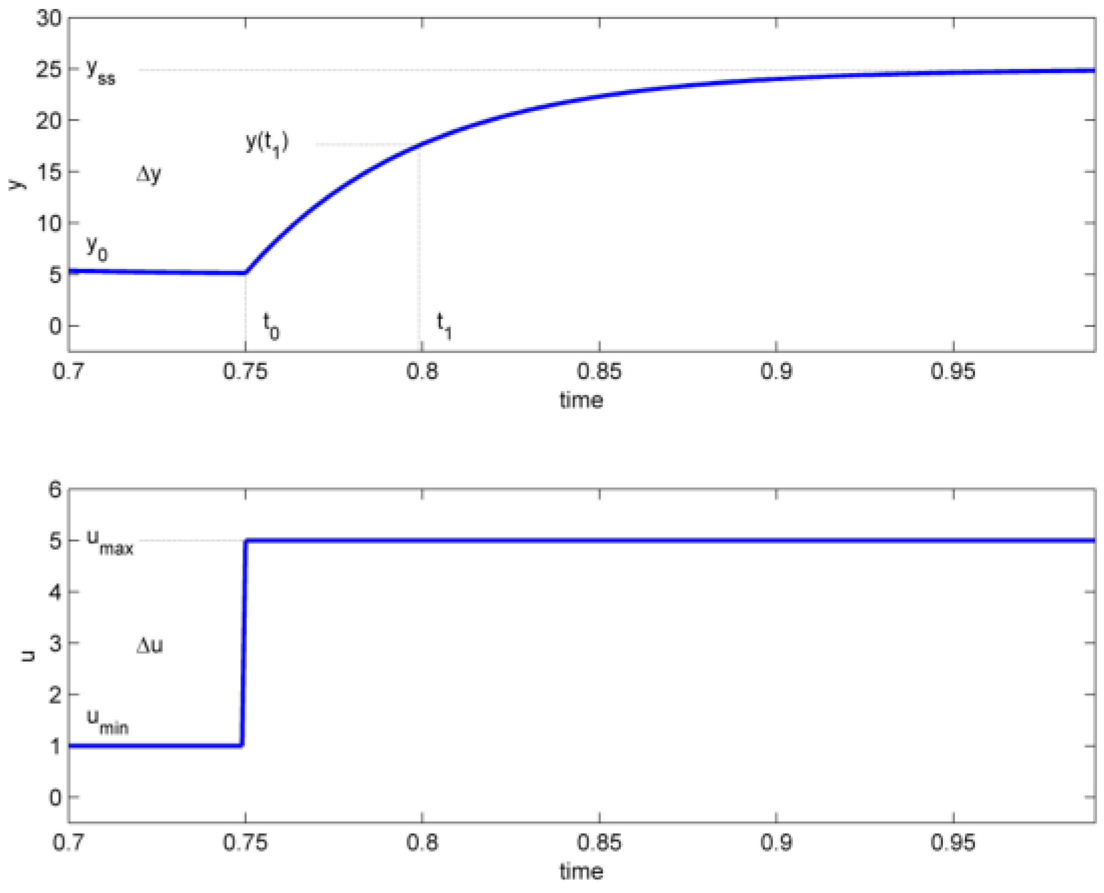


Figure 9 Input and output signal used in a bump test method.

We measured the step response of this system by applying a step input U(s) and recording the output Y(s). K is the steady state gain, and is given by;

$$K = \frac{\Delta Y}{\Delta U} = \frac{y_{ss} - y_0}{u_{max} - u_{min}} \quad \text{Eq. 2}$$

The step input has minimum value is u_{min} and a maximum value of u_{max}. The resulting output signal has initial value y_0 and final steady state value y_{ss}. The time constant τ of the system is given by;

$$\tau = t_1 - t_0 \quad \text{Eq. 3}$$

Where t_0 is the beginning of the step input, and t_1 is the time for Y(s) to reach 63.2% of y_{ss} according to the equation;

$$y(t_1) = 0.632\Delta y + y_0 \quad \text{Eq. 4}$$

Our experimental system controller that we build with Simulink. The lower left part of figure 10 shows the Transfer function for the QUBE servo model of Eq. 1 above. The experimentally found values of K and τ are already plugged into this transfer function. Both the response from model in the lower left part and the response from the motor in the top part are mux together to the output in scope Speed. The other output scope Vm simply measures the input to both systems.

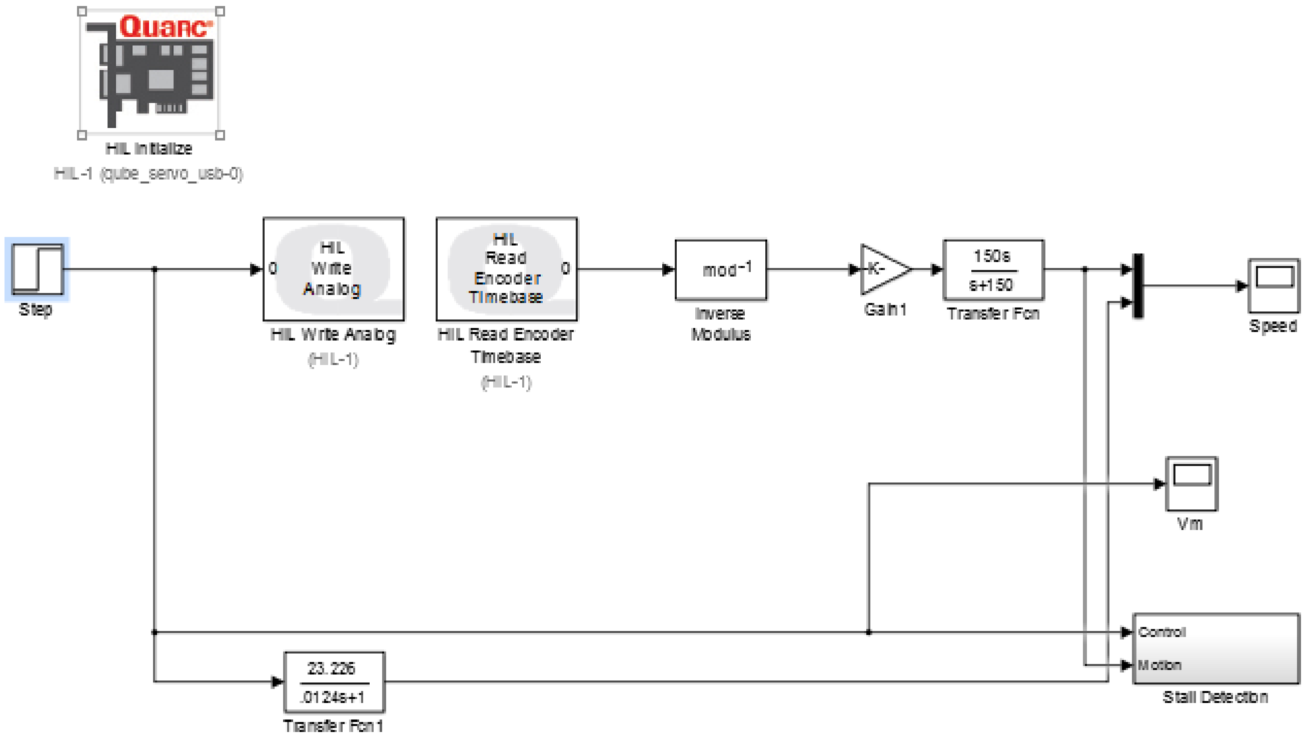


Figure 10 Controller to measure speed.

We used the following MatLab code to plot the data obtained from the simulation:

```
t = data_wm(:,1);%time series
wm_meas = data_wm(:,2);%output to motor
wm_sim = data_wm(:,3);%output of the model built
plot(t,wm_meas, 'LineWidth',3)
hold on
plot(t, wm_sim, 'r--', 'LineWidth', 3) %plot the output of the model
```

Analysis:

Why to use the bump test to model the motor?

The unit step response of a first order system allows us to find the time constant τ of the system, and to use the initial value and final value theorems to characterize the behavior of the system completely.

Why to use first order model?

The unit step response of anything higher that a first order system is more complex as the system will have real and complex roots. We would have to consider different damp cases and pole approximation would be more difficult.

We calculated the parameters for our model from;

$y_{ss} = 46.4527$ - Measured from the mean of points on the graph.

$$K = \frac{y_{ss} - y_0}{u_{max} - u_{min}} = \frac{46.4527 - 0}{2 - 0} = 23.226$$

$$y(t_1) = 0.632\Delta y + y_0 = 0.632(46.4527) = 29.3581$$

$t_1 = 1.1240$ - Taken from data points where $y(t_1)$ that was close to 29.3581

$$\tau = t_1 - t_0 = 1.1240 - 1 = 0.1240$$

Graph of response from the system model and motor:

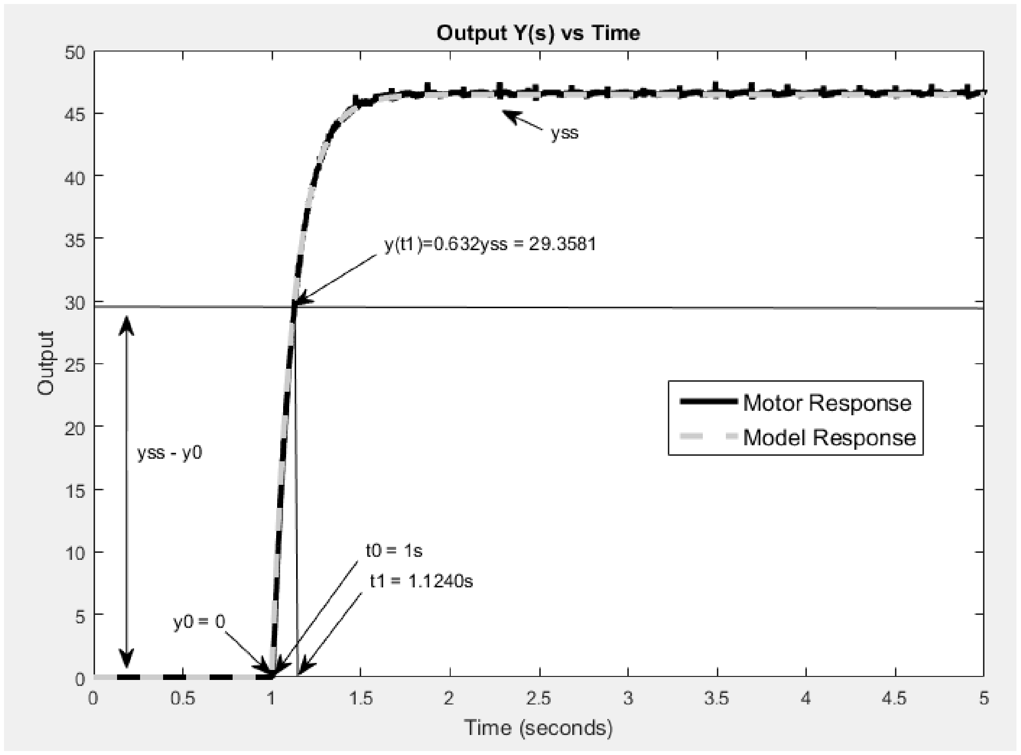


Figure 11 Output of system

The modeling approach was successful in reproducing a unit step response that was very similar to the actual response of the motor. The graph from figure 11 shoes how close we got to the actual response. The methods we used to calculate the parameters of the first order system gave us an overall understanding of the system behavior, and we were able to create a suitable model.

5 Second Order Systems

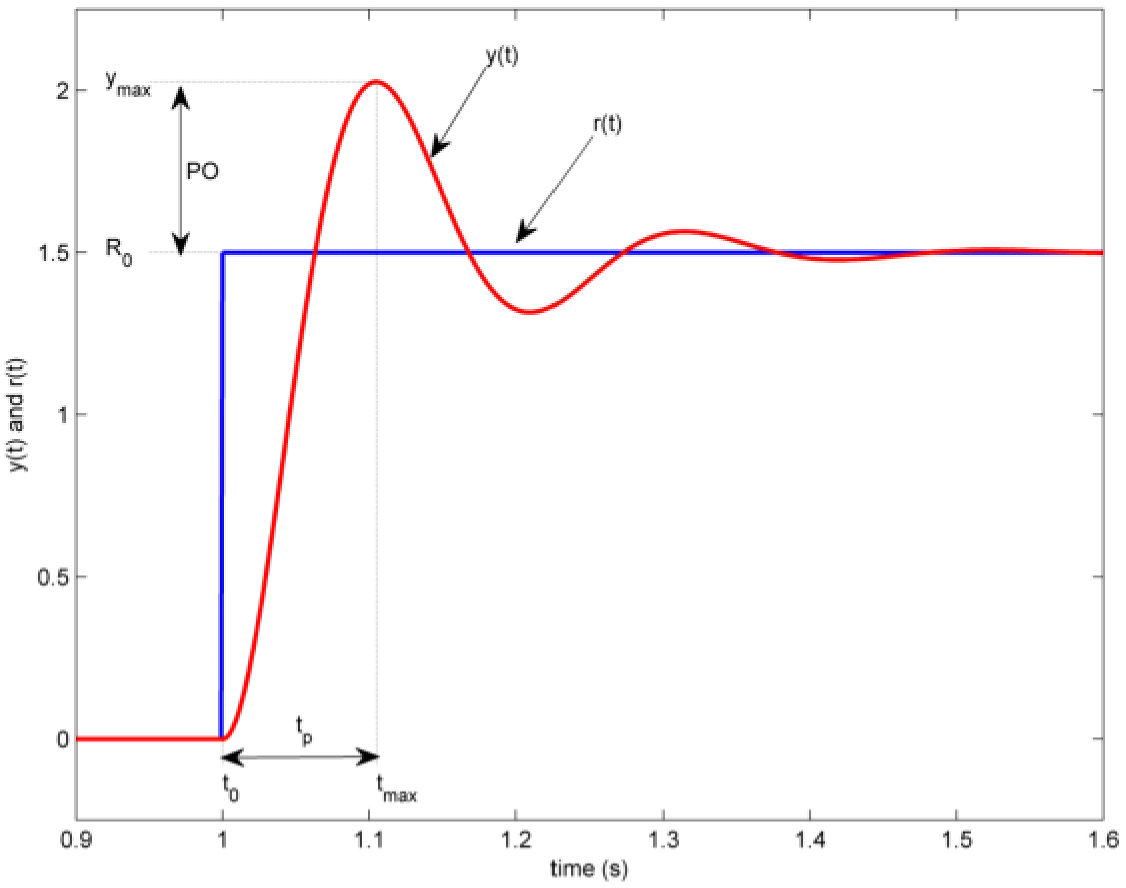


Figure 12 Standard second-order step response

In this part we have a second order system with a transfer function of the form;

$$\text{Eq. 1} \qquad \frac{Y(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

Where R(s) = $\frac{R_0}{s}$ is a step input, ω_n is the natural frequency, and ζ is the damping ratio. The maximum value of the response is given by y_{ss}, and it occurs at a time t_{max}. The percent overshoot is given by;

$$\text{Eq. 2} \qquad PO = \frac{100(y_{ss} - R_0)}{R_0}$$

Or by;

$$\text{Eq. 3} \qquad PO = 100(-\frac{\pi\zeta}{\sqrt{1-\zeta^2}})$$

t_0 is the time where the step input begins, and $t_p$is peak time or the time where Y(s) is max.

Eq. 4 $$t_p = t_{max} - t_0$$

Eq. 5 $$t_p = \left(\frac{\pi}{\omega_n\sqrt{1-\zeta^2}}\right)$$

The Qube-Servo will have a transfer function for speed given by the 1st order system.

Eq. 6 $$G_1(s) = \frac{K}{\tau s + 1} = \frac{23}{0.13s + 1}$$

Where the values for K and τ were found in the previous lab.

The Qube-Servo will have a transfer function for position given by the 2nd order system.

Eq. 7 $$G_2(s) = \frac{K}{s(\tau s + 1)} = \frac{23}{s(0.13s + 1)}$$

Which gives us the closed loop transfer function of the 2nd order system.

Eq. 8 $$T(s) = \frac{K/\tau}{s^2 + \frac{1}{\tau}s + K/\tau} = \frac{23/0.13}{s^2 + \frac{1}{0.13}s + 23/0.13} = \frac{176.92}{s^2 + 7.692s + 176.92}$$

Combining the given values, we can solve for ζ and ω_n;

$$\omega_n^2 = 176.92$$

$$\omega_n = 13.30$$

$$2\zeta\omega_n = 7.692$$

$$\zeta = 0.2892$$

Then using Eq. 3 the theoretical PO = 38.71%.

Using Eq. 5 the theoretical $t_p = 0.2468$.

Our experimental system controller that we build with Simulink. The model shows unity feedback and a step input on the left side. The output of the controller is the position of the disk on the motor.

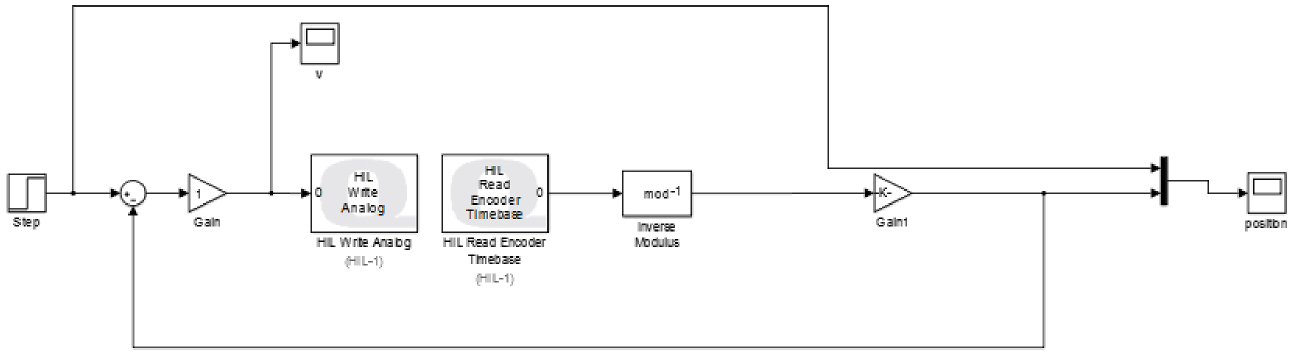


Figure 13 Controller to measure position.

We used the following MatLab code to plot the data obtained from the simulation:

```
t=data_wm(:,1);
wm_input=data_wm(:,2);
wm_motor=data_wm(:,3);
plot(t,wm_input,'LineWidth',3);
hold on
plot(t,wm_motor,'r--','LineWidth',3);
```

Analysis:

From the simulation data taken from figure 13 we obtained the following results;

The percent overshoot is

$$\text{PO} = \left(\frac{1.270-1.00}{1.00}\right) 100 = 27.9\%.$$

The time to peak is

$$t_p = t_{max} - t_0 = 1.258 - 1 = 0.258\ seconds$$

Compare to our theoretical data we get an error for t_p of

$$t_{p-error} = \frac{0.258 - 0.2468}{0.2468} x100 = 4.5\%$$

Graph of response from the motor:

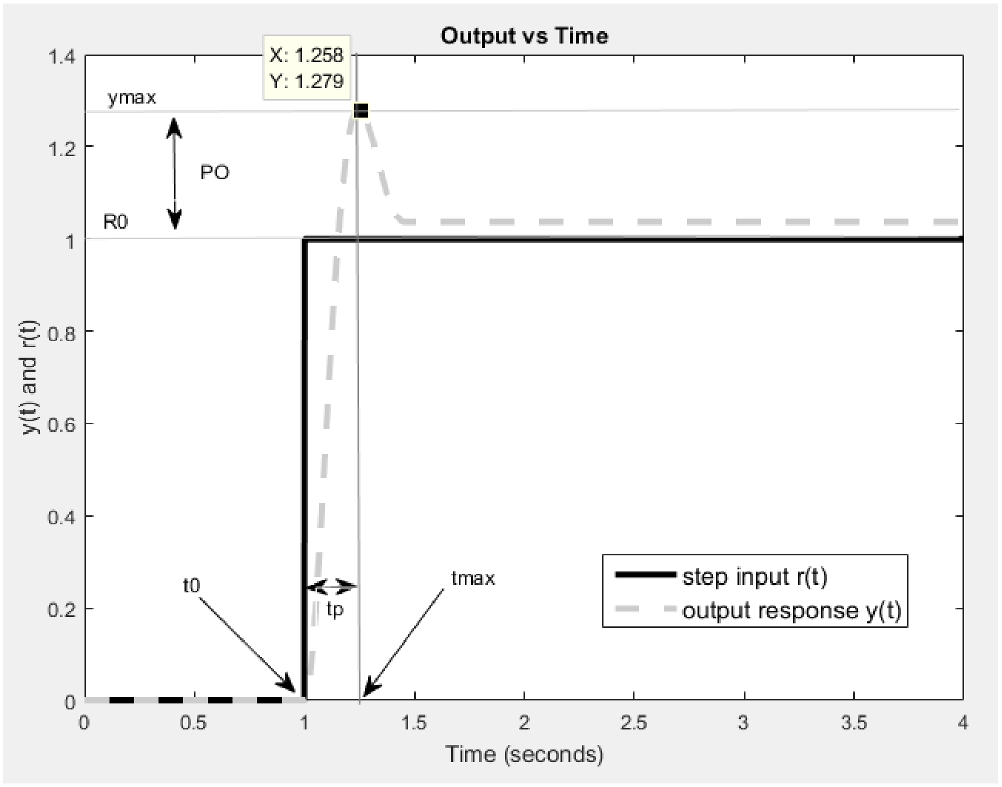


Figure 14 Output of system

Overall, I think we accomplished the objective. Our theoretical values were very close to the actual values obtained from the lab. The only reason for deviation I can think of is that we initially considered the values for K and τ to be the given ones, but in our previous work we had found that those values would change depending on the motor we were working with. The slight deviation in the end is due to the non-precision in our chosen K and τ parameters.

6 PD Control

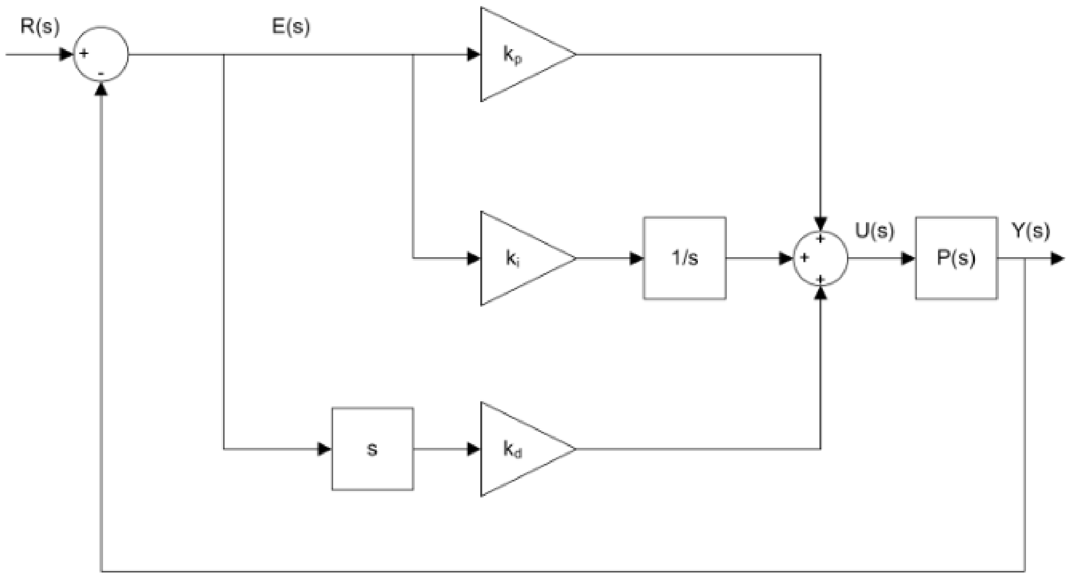


Figure 15 Block diagram of PID control.

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.

In this part we will try to implement a PD controller as a proportional-velocity controller with the following structure;

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

Where k_p the proportional is gain, and k_d is the velocity gain. The closed loop transfer function for the system is;

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

This is a standard second order transfer function, that has parameters;

K = 23.2

τ = 0.13 seconds

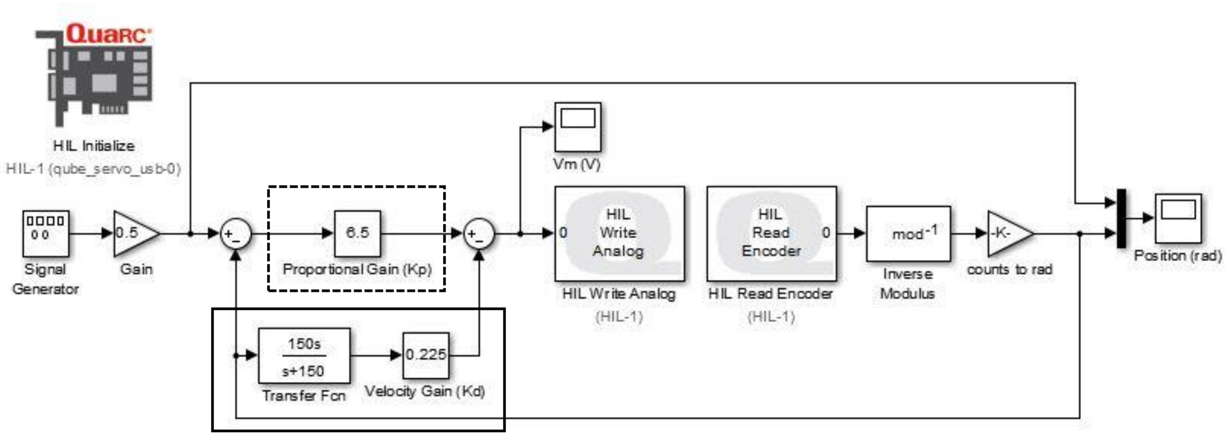
Eq.1 $$w_n{}^2 = (Kk_p)/\tau$$

Eq.2 $$2\varsigma w_n = ((1 + Kk_d)/\tau)$$

The parameters K and τ are taken from voltage-to-position transfer function of the plant;

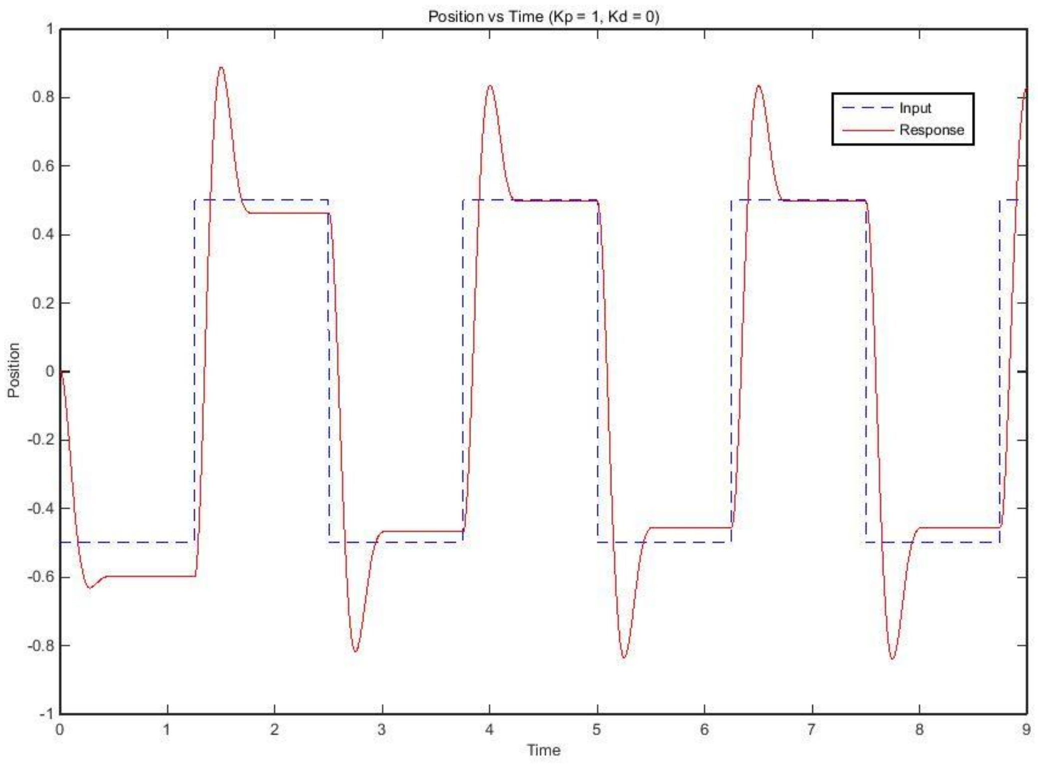
$$G(s) = \frac{K}{s(\tau s + 1)} = \frac{23.2}{s(0.13s + 1)}$$

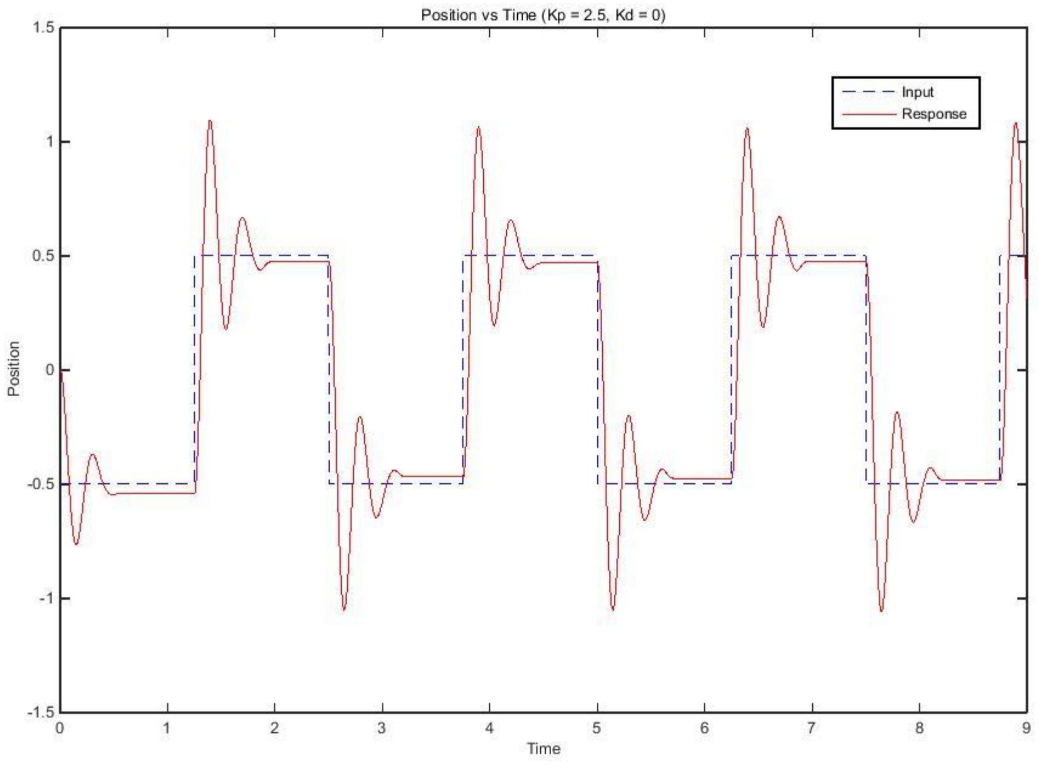
This is our implemented PD controller Simulink schematic. The dashed outline shows the proportional controller, and the solid outline shows the derivative controller.



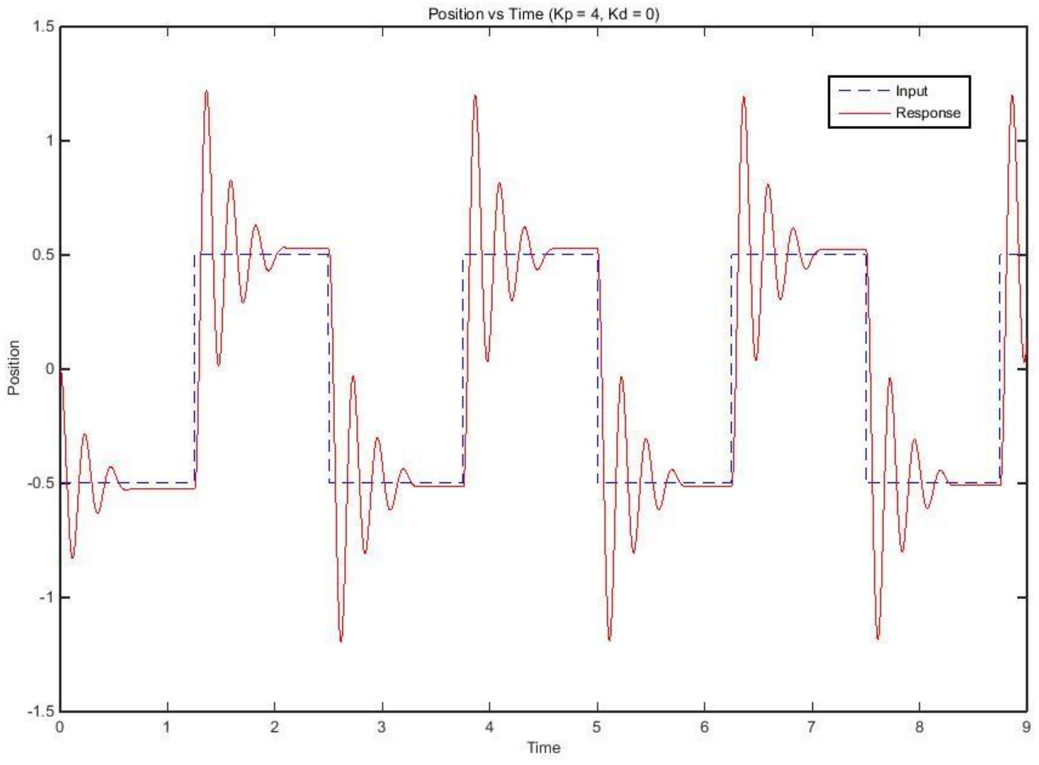
Function of the P controller

We set $k_d = 0$ and vary $k_p = 1, 2.5, 4$.





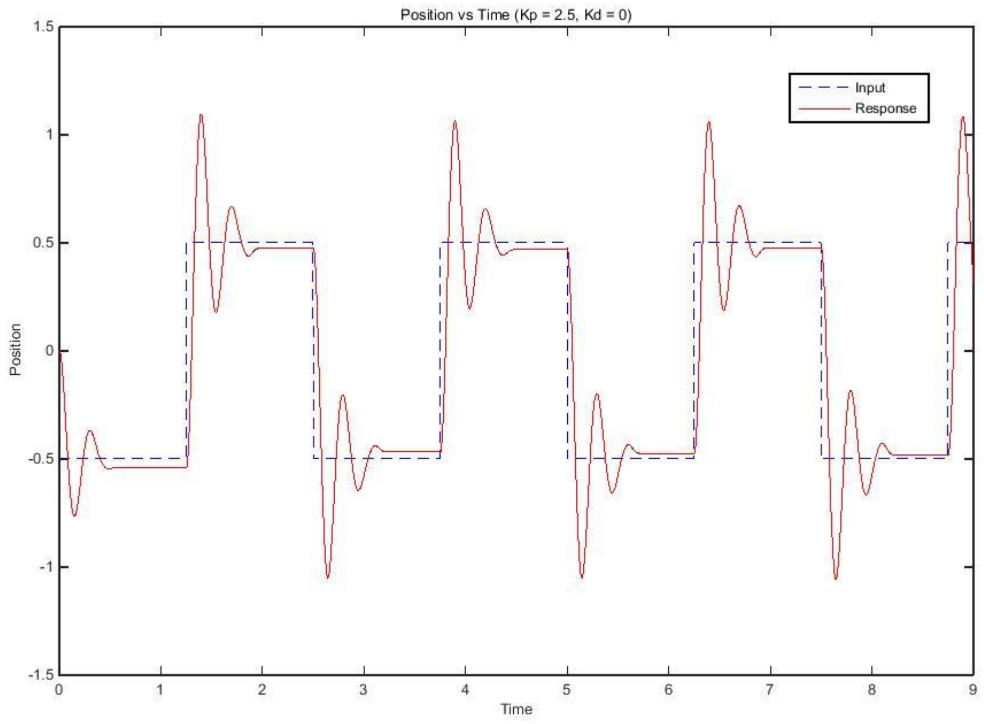
Position vs Time (Kp = 2.5, Kd = 0)
Input
Response
Position
Time

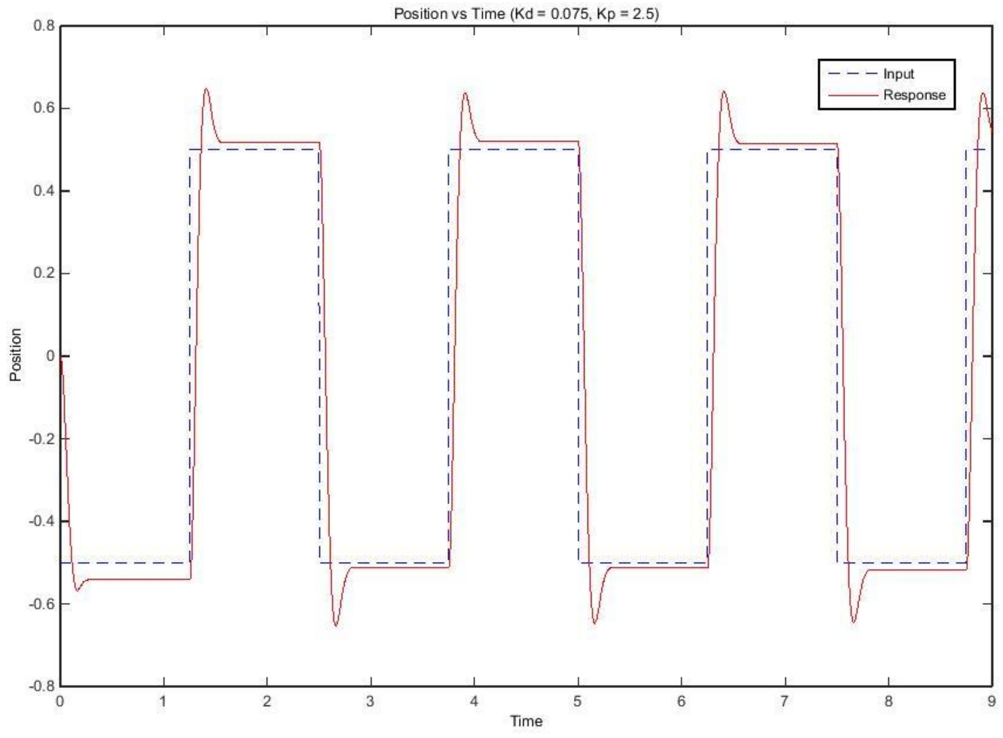


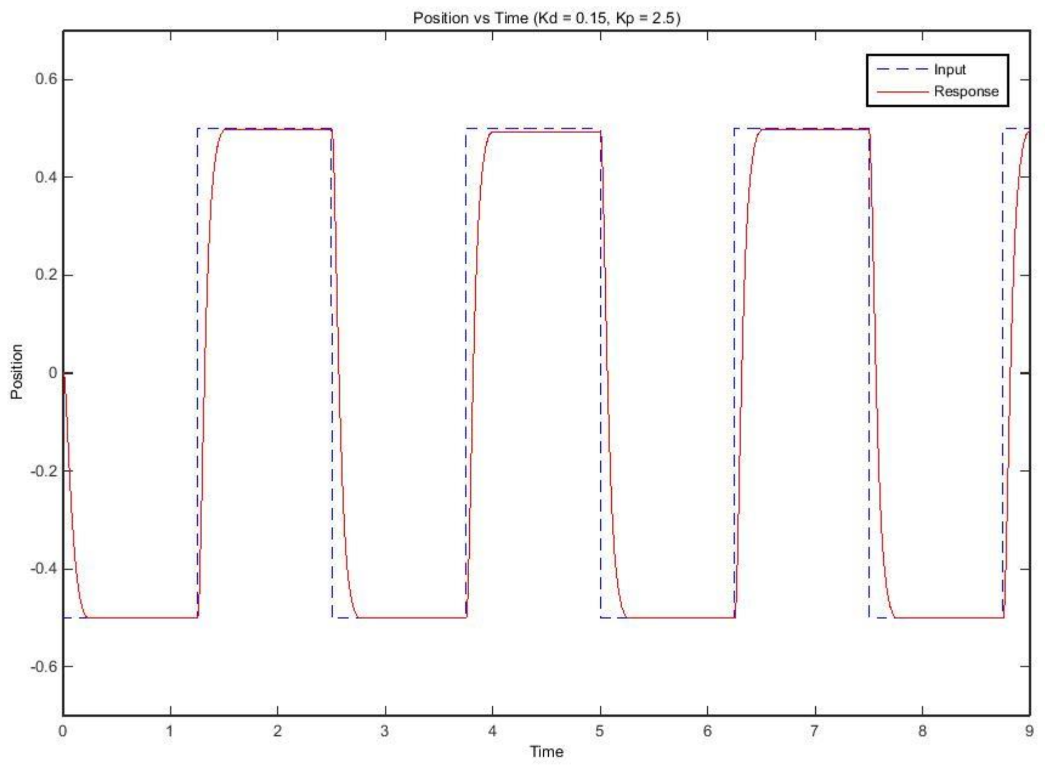
Position vs Time (Kp = 4, Kd = 0)
Input
Response
Position
Time

Function of the D controller

We set $k_p = 2.5$ and vary $k_d = 0, 0.075, 0.15$.






Position vs Time (Kd = 0.15, Kp = 2.5)
Input
Response
Position
Time
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
1
2
3
4
5
6
7
8
9

Finding the parameters for given specifications of:

Peak Time: 0.15s

Overshoot: 2.5%

$\zeta = 0.76$

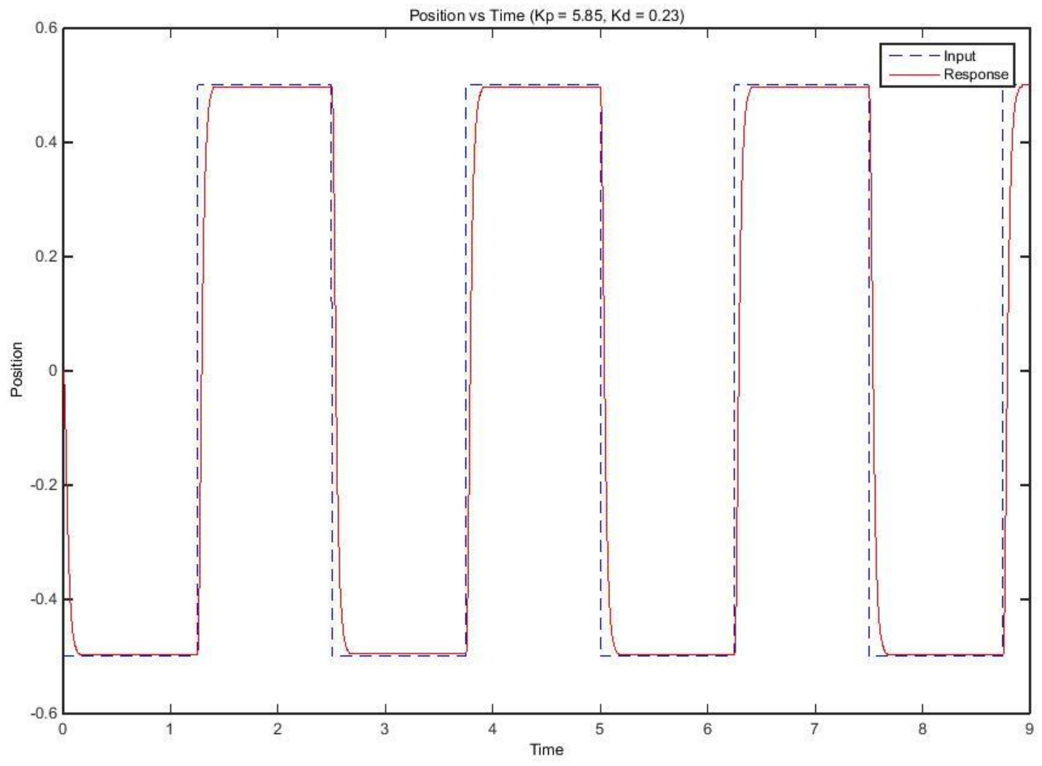
$w_n = 32.3$

Using Eq.1 and Eq. 2 from the theory section we find;

$$k_p = 5.85$$

$$k_d = 0.23$$

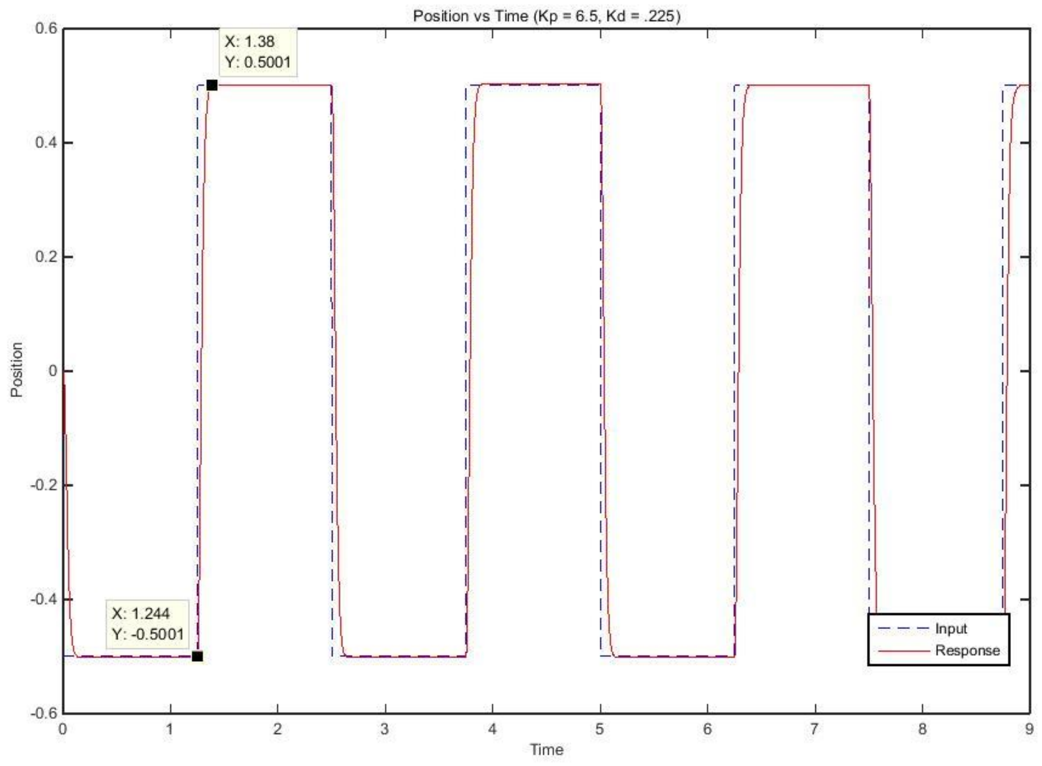
For these parameters we got the following response;



Not all specifications were met, so we tuned the PD controller. We obtained the following response for;

$$k_p = 6.5$$

$$k_d = 0.225$$



The above response meets the specifications;

Peak Time: $t_{max} - t_0 = t_p = 0.1361\ seconds$

Overshoot: $\frac{100(y_{max} - r_0)}{r_0} = .02\%$

Analysis:

If we increase kp, what will happen to the response?

The time to peak and time to rise decreases, but the percent overshoot and time to settle increases.

If we increase kd, what will happen to the response?

The percent overshoot and time to settle decreases, but the time to rise increases.

7 Modeling and Balance Control of Pendulum

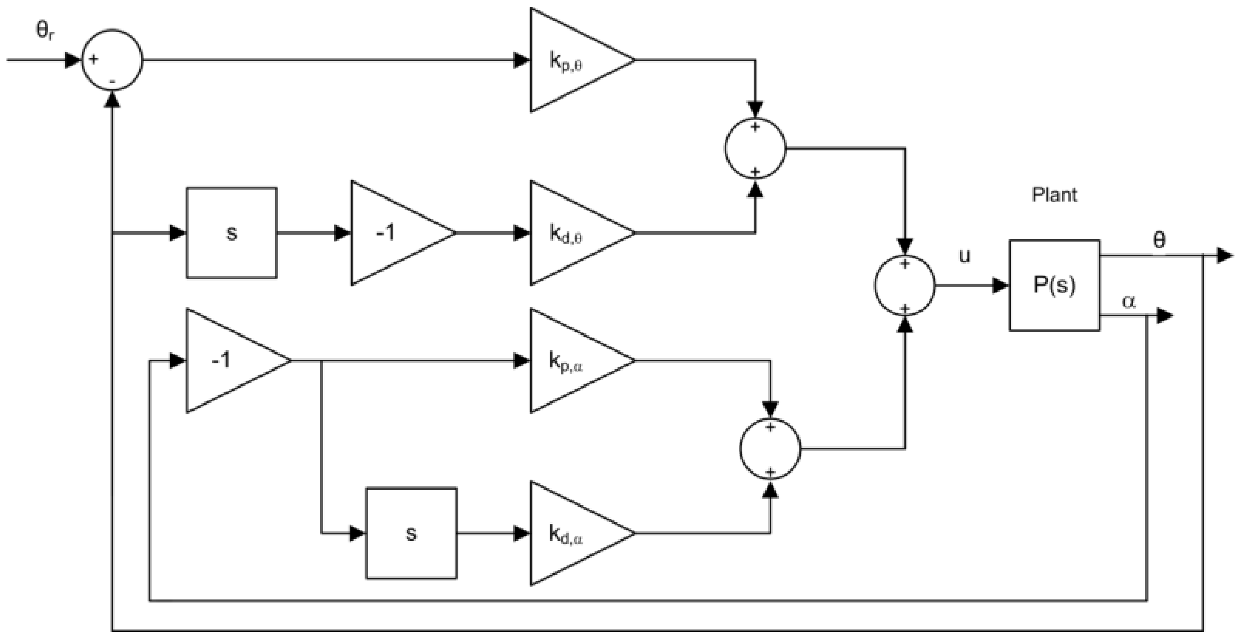


Figure 16 Block diagram of balance PD control for rotary pendulum.

In this part we developed a feedback controller to balance a pendulum in the upright position. The control law can be described by the following equation

$$u(t) = k_{p,\theta}(\theta_r - \theta) - k_{p,\alpha}\alpha - k_{d,\theta}\dot{\theta} - k_{d,\alpha}\dot{\alpha}$$

Where $k_{p,\theta}$ is the proportional gain of the lower rotating arm, and $k_{p,\alpha}$ is the proportional gain of the pendulum arm. $k_{d,\theta}$ and $k_{d,\alpha}$ are the derivative gains of the rotating arm and pendulum arm. The angle θ_r is the desired angle of the arm. The angle α is the pendulum angle which is zero when the pendulum is in the upright position. The controller will only enable the balance control mode when the angle α is less than 10 degrees from the upright position. The angle α_{full} is the pendulum angle measured by the encoder. The relationship of the pendulum angles is;

$$\alpha = \alpha_{full} \, mod \, 2\pi - \pi$$

Both angles are defined as positive when rotated counter clock wise.

In essence we are using two PD controllers, one for each arm. The parameters for the PD controllers are;

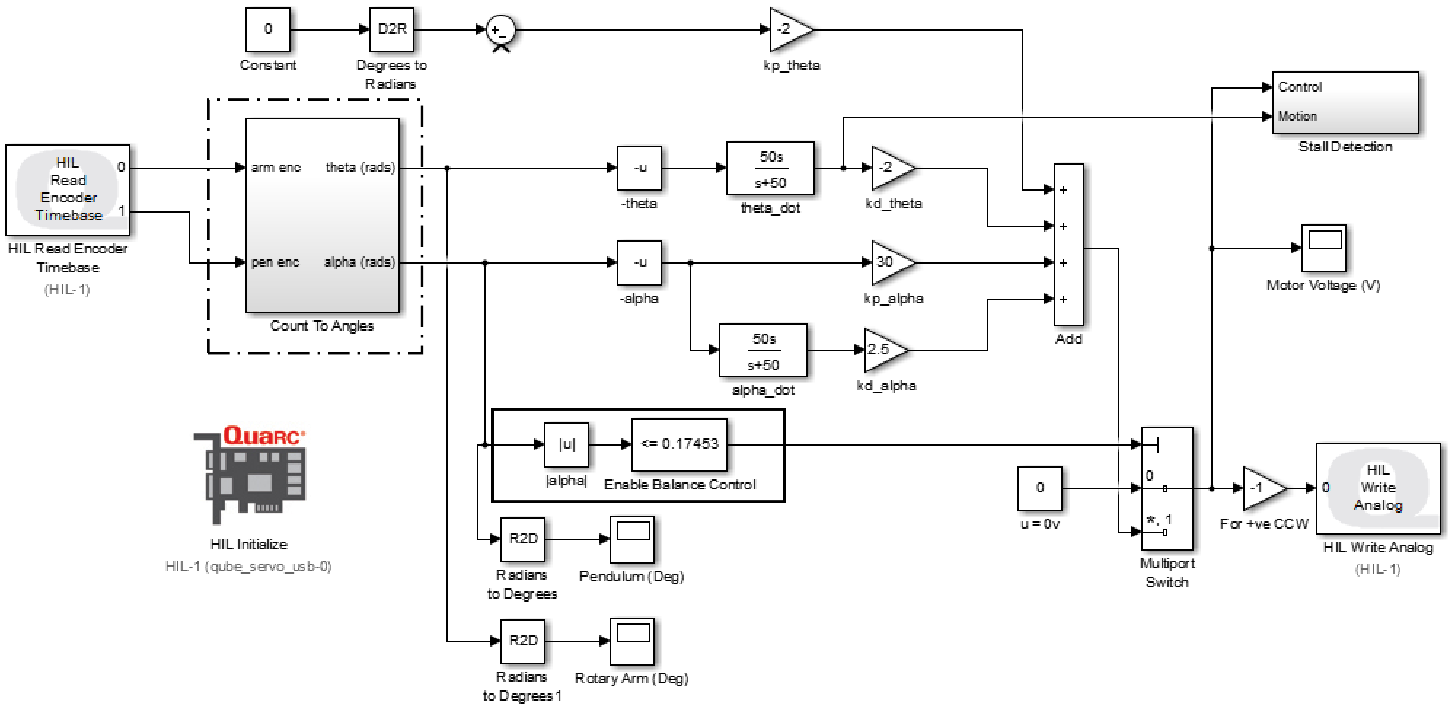
$$k_{p,\theta} = -2$$

$$k_{p,\alpha} = 30$$

$$k_{d,\theta} = -2$$

$$k_{d,\alpha} = 2.5$$

This is our implemented PD controller Simulink schematic. The dash-line box shows the part of the controller that measures the pendulum and rotary arm angles from the encoder using rad/s. The dashed box shows the 2 PD controllers that control the arm positions. The line box shows the angle at which the balance controller is engaged.



Analysis:

In the balance control part, why should α be small?

Because we linearized the original equation for the motion of the pendulum. The nonlinear equation gives a moment of inertia J_p of;

$$J_p\ddot{\alpha} = \frac{M_p g L_p}{2}\sin(\alpha(t))$$

Compared to the linearized equation we used:

$$J_p = \frac{M_p g L_p}{(2\pi f)^2}$$

Which doesn't consider the sine of the angle of α, and we instead chose a small range for α to be at. The larger the deviation from optimal settings, the small angle α, the larger the error in our linearized system.

Figure 17 shows the position of the pendulum as I introduced disturbances. The pendulum would receive a disturbance of <10 degrees and the controller would correct the position back to the 0 degrees in the upright position. Figure 18 shows how the rotary arm reacted to the disturbances, in order to correct the pendulums position, in this case, the rotary arm would rotate CCW. Figure 19 shows the voltage applied to the controller, when disturbances to the system occurred the voltage applied increased as shown.

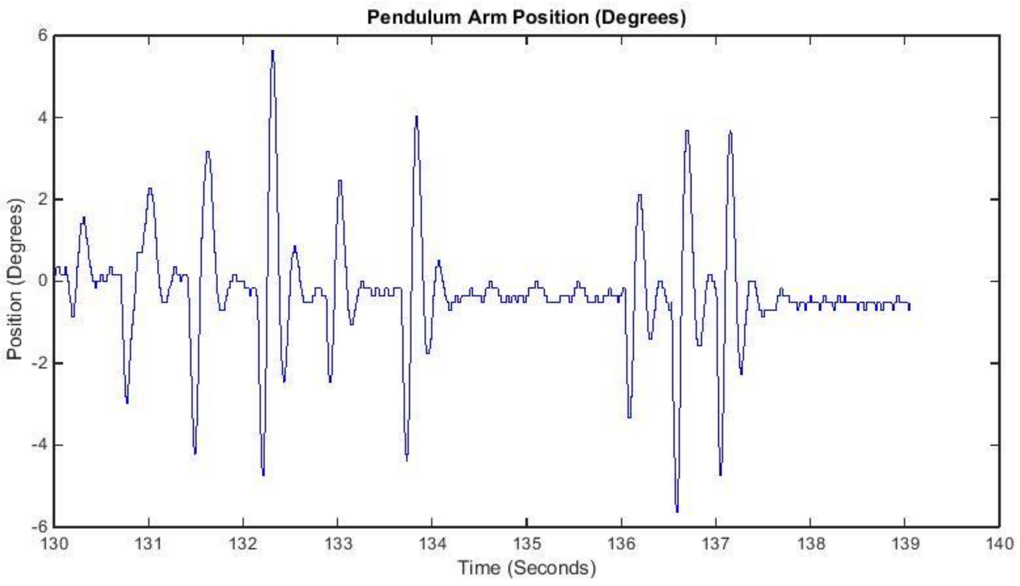


Figure 17

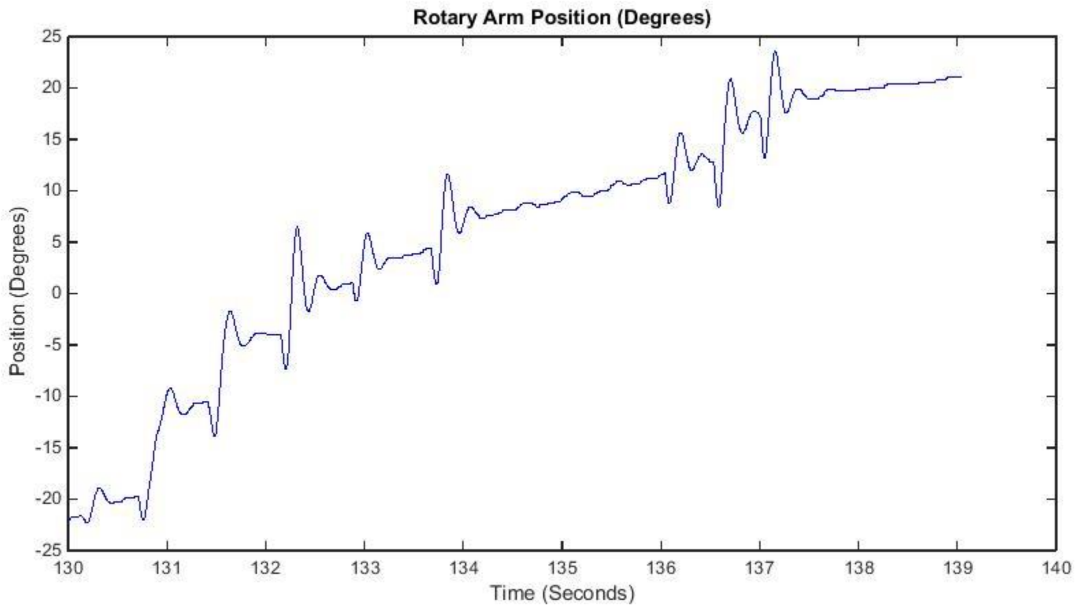


Figure 18

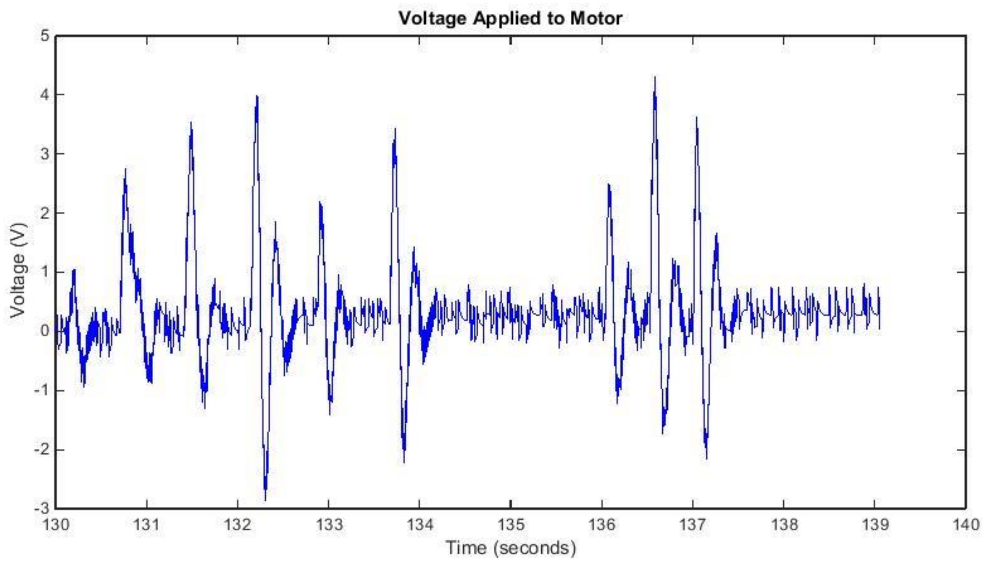


Figure 19

When the angle of the pendulum with the upright position, it suffered a disturbance that was a small angle α, the controller was then able to effectively balance the pendulum by correcting the angle α back to its original position. If the angle of disturbance α was too large, the controller would disengage, and the pendulum would free fall. Overall, the controller was effective when the angle of disturbance was small and shows that our linearization was effective given a small range of disturbance for angle α.

8 Appendices

Appendix A: Parts and Software List

- Quanser Qube Servo (https://www.quanser.com/products/qube-servo-2/)
- MATLAB Simulink (https://www.mathworks.com/products/simulink.html)