Position Control

1. Proportional controller

We use motor1 as the plant. The transfer function of velocity is:

$$g = \frac{0.01831}{s + 10.79}$$

As discussed in PI controller document, we choose the PI controller as:

$$d = \frac{1245(s + 20)}{s}$$

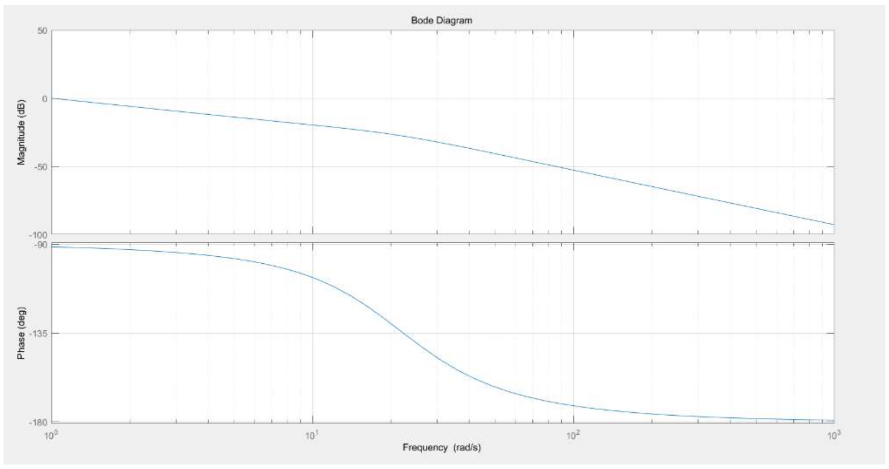
The close-loop transfer function is:

$$Gv = \frac{22.8s + 455.9}{s^2 + 33.59s + 455.9}$$

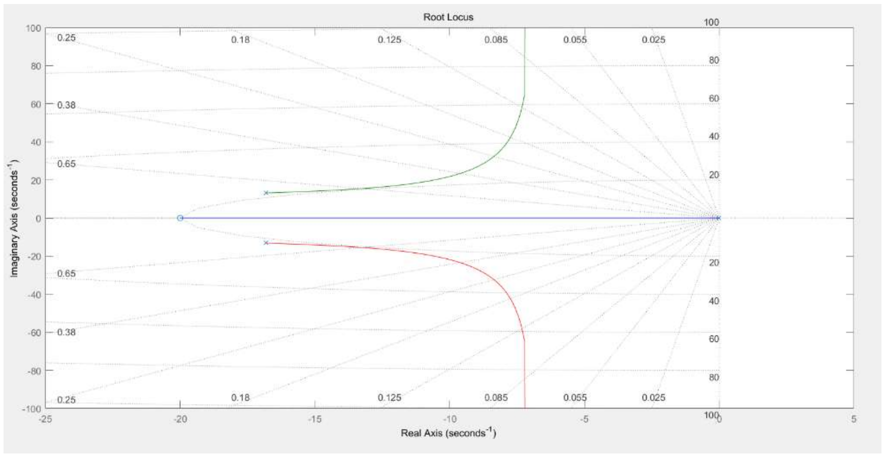
The transfer function of position is:

$$Gp = \frac{22.8s + 455.9}{s(s^2 + 33.59s + 455.9)}$$

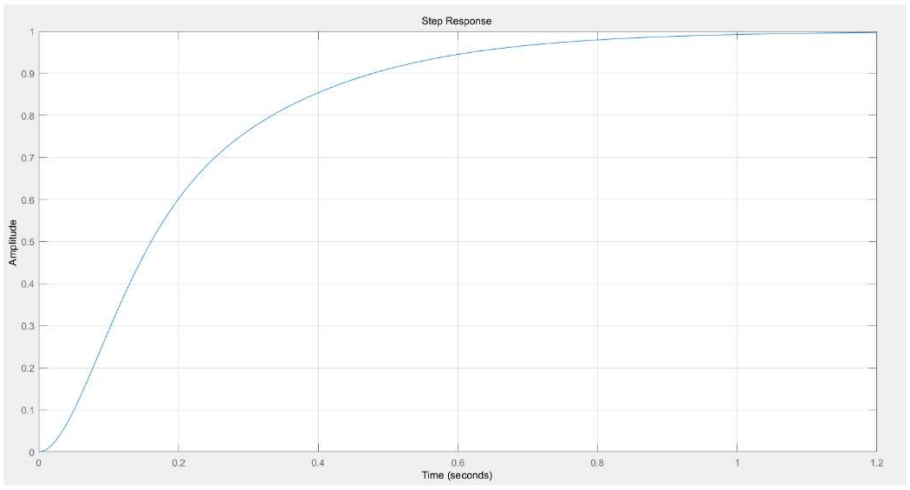
The corresponding bode diagram is:



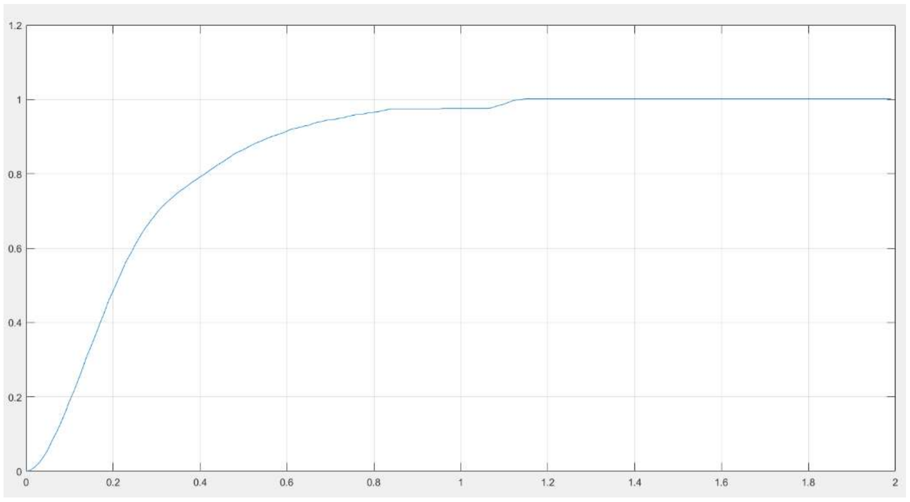
And the root locus is shown below:



Now consider the proportional controller. If we want to choose the damping ratio as 0.707, from the root locus, we find the gain is around 4.5. The step response is simulated as below.

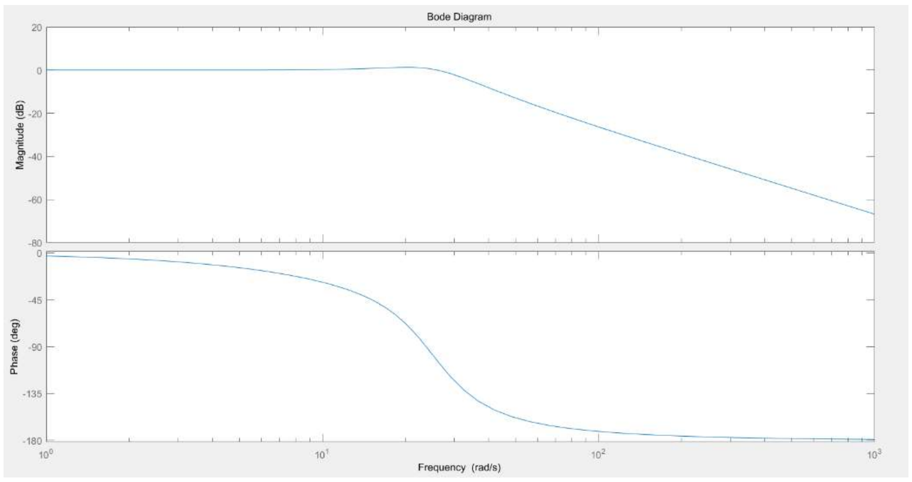


And here is the test result:

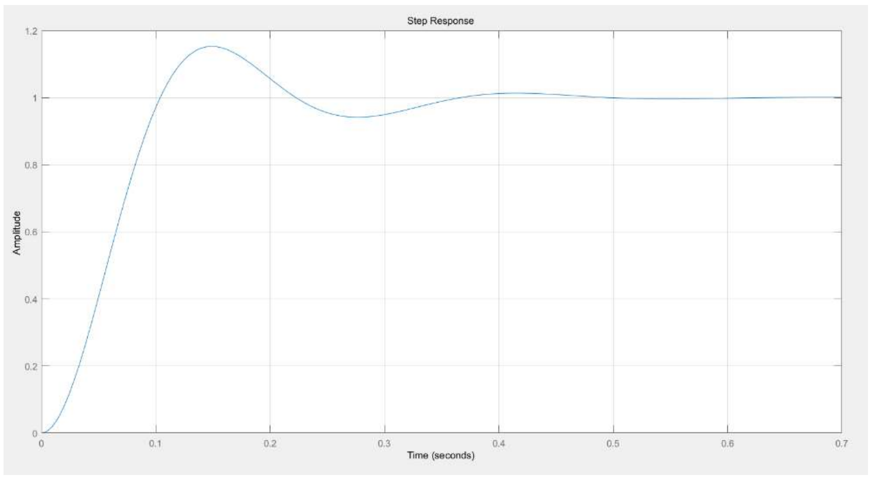


They look similar but this response is too slow.

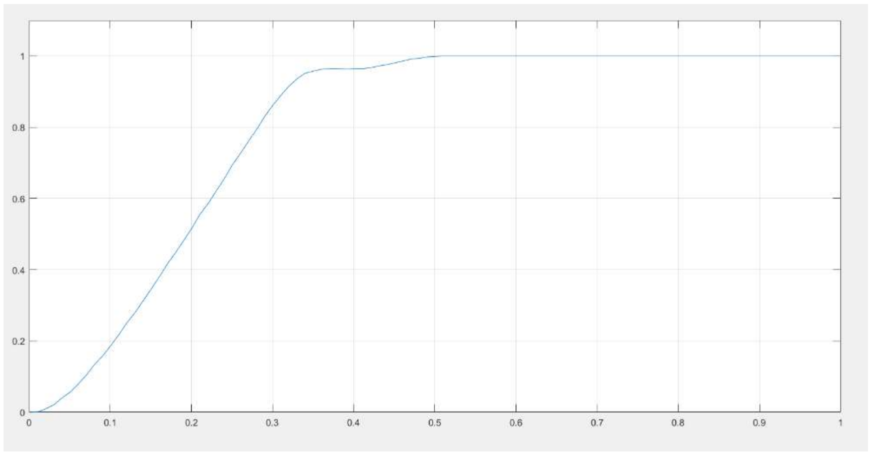
We now choose Kp = 20. From the root locus, we can find the damping ratio is 0.3, Here is the close loop bode diagram:



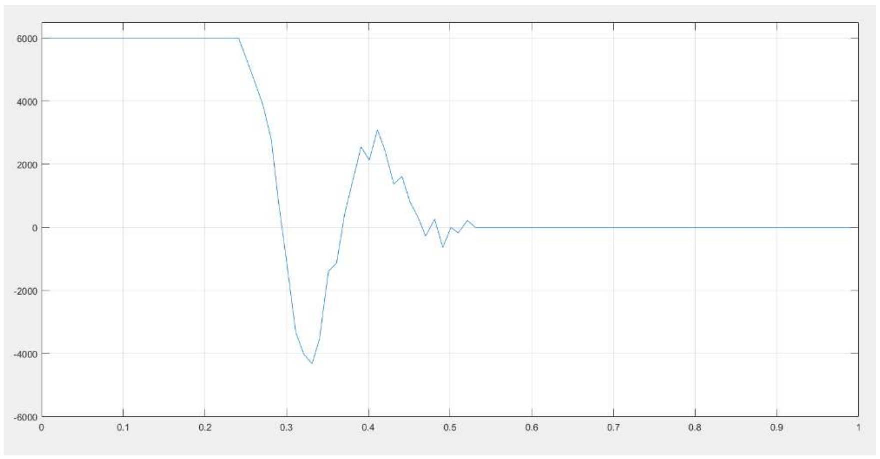
The simulation of step response is:



And the actual test result is shown below:

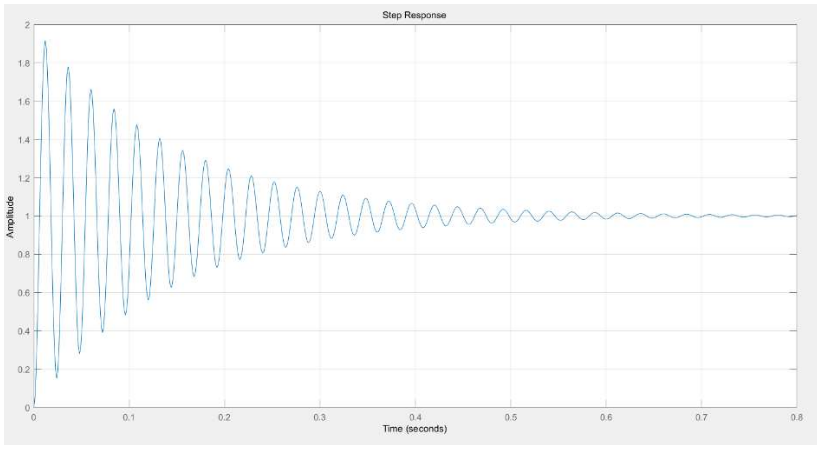


As we can see, the actual result is slower than the simulation. The main reason for this should be the saturation of the motor. We can find this clearly from the control signal below:



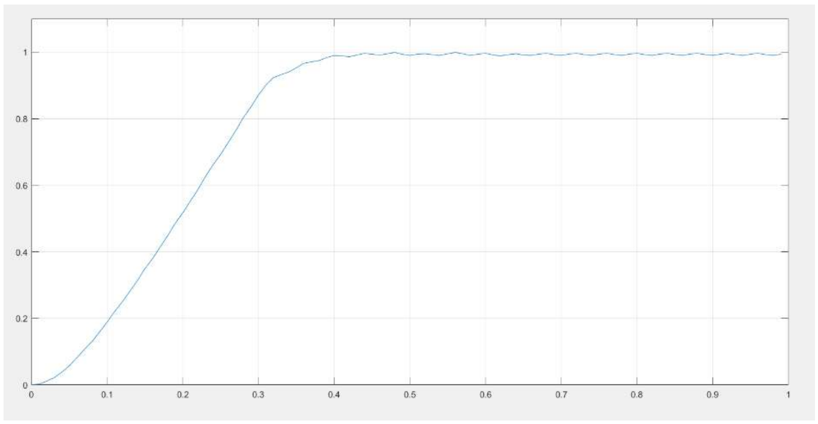
The motor is saturated in the first 0.25s. Therefore, the motor is slower than it supposed to be. Besides, we have dry friction in the motor which is neglected in the simulation, this error might contribute to the actual slow response.

To see the effect of saturation more clearly, we set the Kp = 3000. The simulated step response is shown below:



This response is bad. We observe strong oscillation here.

Here is the actual test result:



We can observe the oscillation clearly. However, there is a major difference between the simulation and real test. The magnitude of oscillation in real test is a lot smaller than the simulation. This is also because the motor is saturated. Since the control sign can't be bigger than 6000, the controller behaves like someone set the Kp to a smaller value when the motor is saturated.

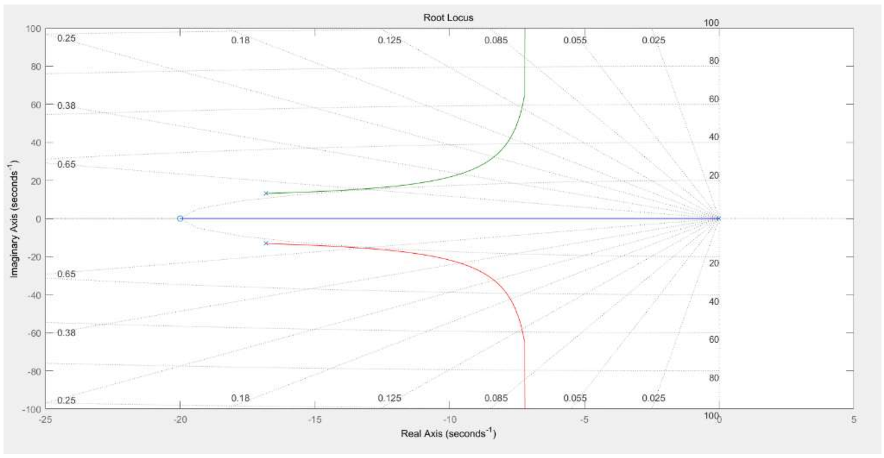
2. The influence of a non-ideal speed controller.

Our speed controller is limited by the sample rate. It requires some time to track the reference speed. This makes the speed loop become a delay in the system. We can observe the effect of it from the root locus.

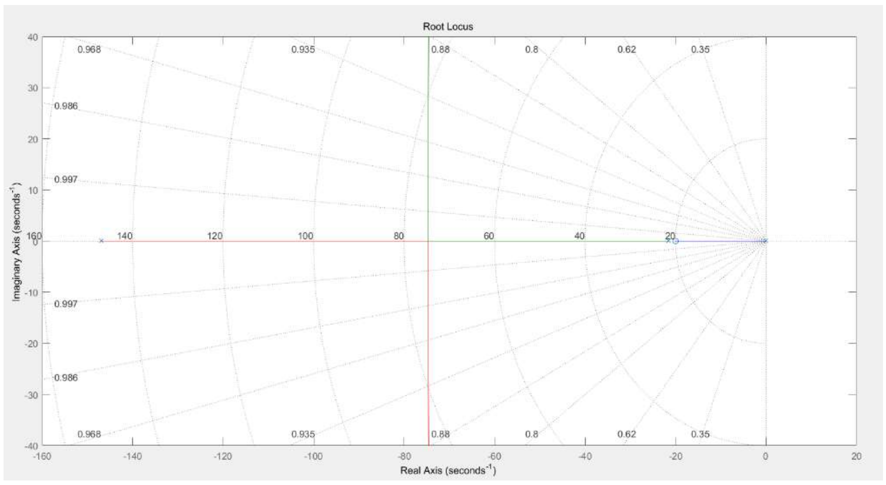
The first figure is the root locus of current speed loop with an integrator, which had been shown before. The bandwidth of this PI controller is 32 rad/s, which is 1/20 of our sample frequency. And the second figure is a PI controller with bandwidth = 320 rad/s, ten time larger than the original one.

Obviously, in the first one, no matter what the Kp is, our system is a 2nd or 3rd order system. And the real part of poles is smaller than 18. We can find the influence on two aspects.

First, the influence of bandwidth on the settling time. From the 2nd order system response, we know that the setting time is $t_s = \frac{4.6}{\sigma}$. Therefore, our position loop's settling time is at least 0.2s



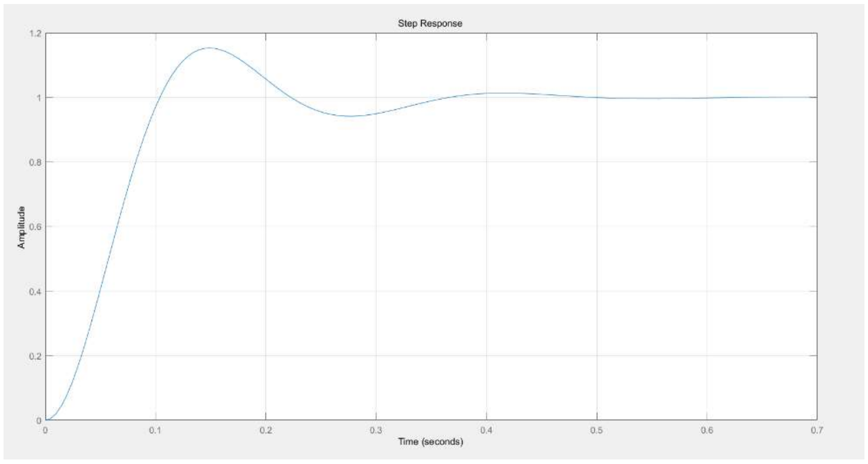
PI bandwidth = 32 rad/s



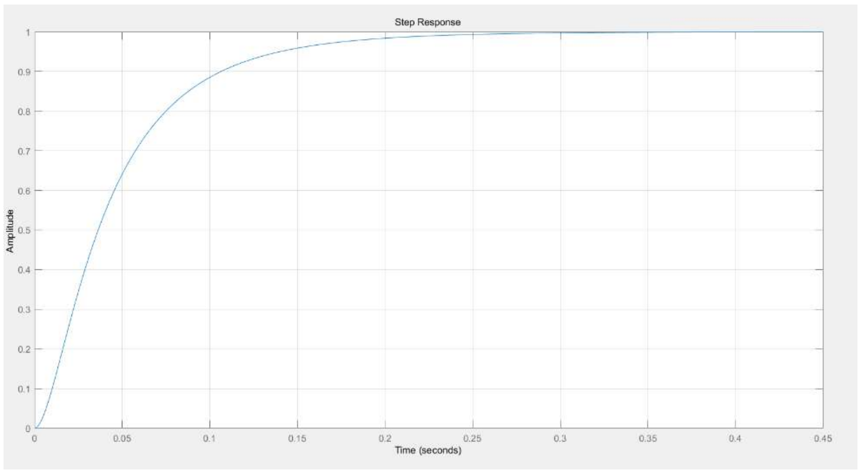
PI bandwidth = 320 rad/s

Second, the damping ratio is also influenced by the PI bandwidth. In the first figure, the largest damping ratio is 0.787 and in the second figure, the damping ration could be 1. This means we have more oscillation in the smaller bandwidth.

Let’s set Kp = 20 and check the simulation of step response of these two systems.



PI bandwidth = 32 rad/s

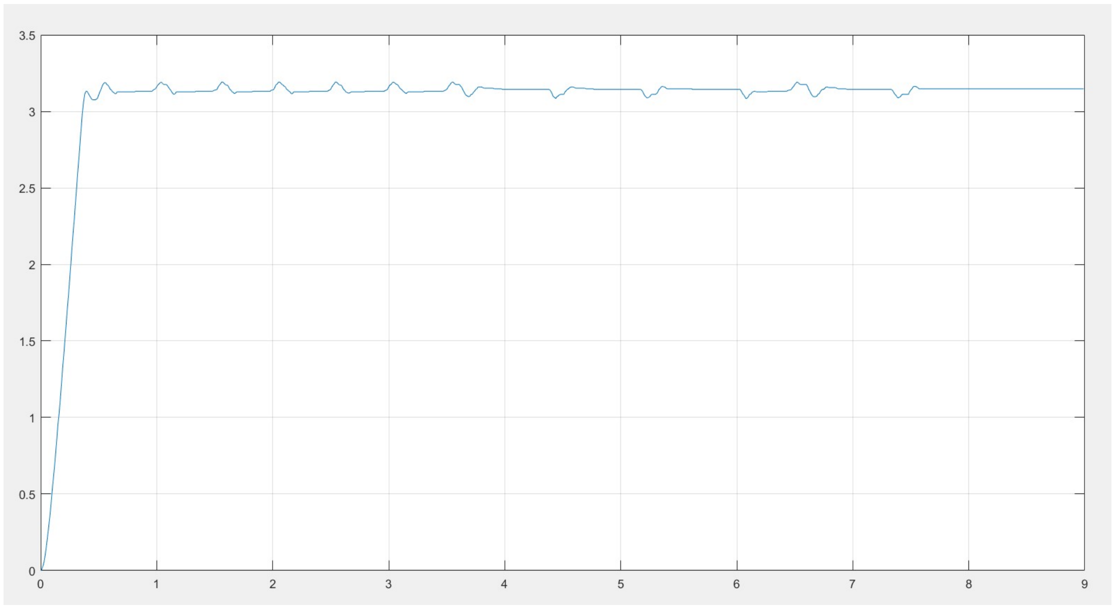


PI bandwidth = 320 rad/s

We can find that with larger bandwidth, the step response is a lot faster and there is no overshoot. Therefore, a larger bandwidth of speed loop is preferred. However, as mentioned before, the bandwidth is limited by the sampling frequency hence can't be chosen freely.

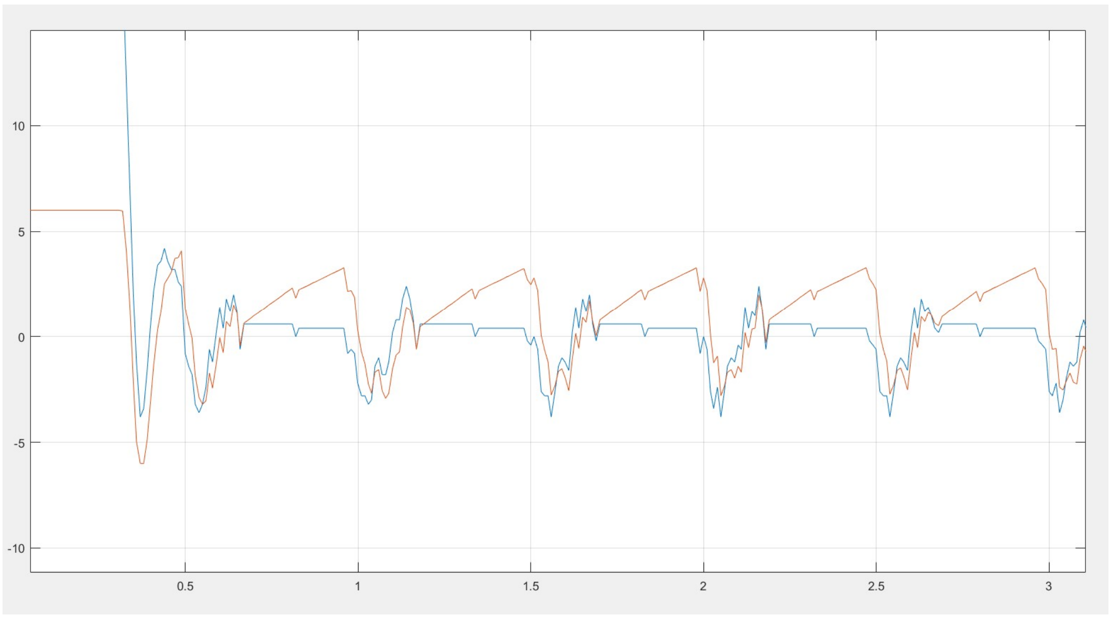
3. Non-linear dry friction in motor2

The controller is successful in motor1. However, when we tried to do the same trick on motor2, thing got strange. Here is the step response:



There are many small peaks on this curve. What happened?

We think this is a typical stick slip phenomenon which usually happens in low speed. The stick slip phenomenon consists of two phases. The first one, micro elastic deformation. The driven force is not enough to overcome the static dry friction. Energy increased with deformation. The second one, elastic force is big enough and triggers a movement. Because dynamic friction coefficient is small than static one, there is an acceleration. Once the energy stored in phase one is fully consumed, the movement stop and we go back to phase one. We can observe such two phased clearly in the error and voltage diagram. The voltage below has the unit of [Volt]



The blue line is error and the red line is voltage.

We can see that the controller tries to drive the motor to reference position but the dry friction prevents motor2 from moving. Energy raises until the voltage is big enough. However, this causes the overshoot and the system goes back to phase 1.

By the way, we can feel this effect when we turn the wheel of motor 2 by hand…

Because of such a friction. It's hard to turn the motor to the reference precisely just by a proportional controller. We decided to give a small tolerance e = 0.01 rad so that the motor can stop.

4. Using a PI controller to deal with the non-linear friction in motor2

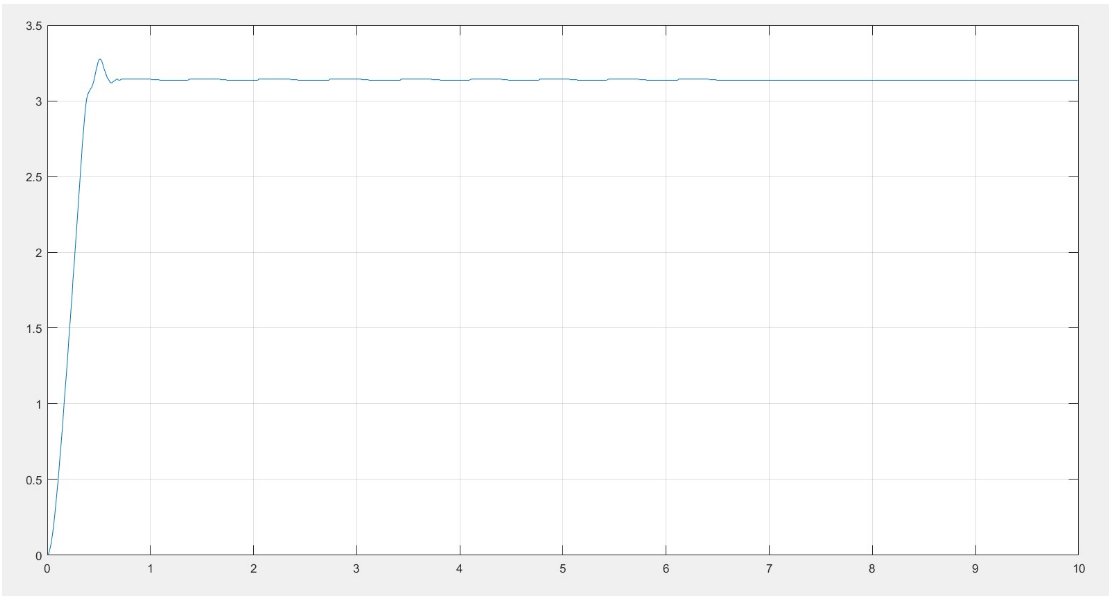
The drive signal is not big enough when stick slip happens. Therefore, we think a PI controller, which uses the help from an accumulation of error, can fix this problem.

We didn't work on frequency domain this time because the non-linear friction is hard to model. Instead, we design the PI controller by tuning the Ki factor. Seeing the code may make it clear:

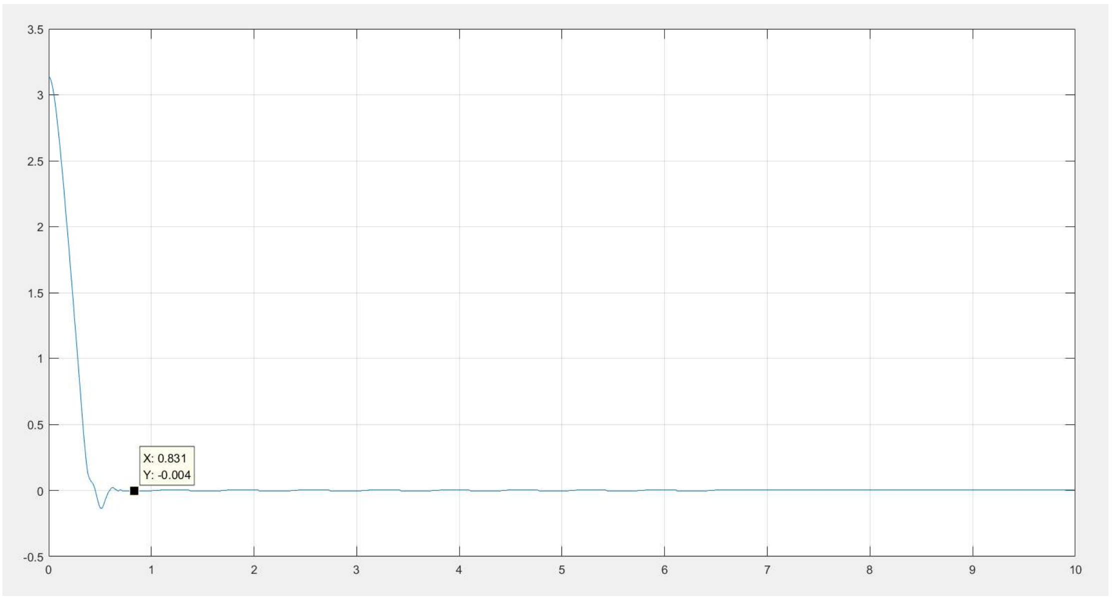
```
float Ki = System.getGPinFloat(2);
if(ep2<0.1&&ep2>-0.1){
    integral += ep2;
    rv2 += Ki*integral;
}
```

We only use the integral signal when the error is small. This could solve the integral windup. An integral windup happens when there is a sudden big change in our system. The effect of such a big error will continue even after the big error has disappeared.

After some tuning work, we choose the Ki = 10. This is the test result:



And the error is shown below:



There is a small oscillation in our response with a magnitude of 0.004 rad. However, the resolution of our encoder is 0.004 rad which means such a small oscillation is acceptable.

Something strange still happens. Our PI controller doesn't work all the time. Once in 5~10 times, it will fail. There must be something hiding behind it... Maybe the non-linear friction will change at different angle...