ECE 141 Project Report

Design a quad-rotor controller

Name: Ma Yunhui

UID: 505-179-815

1. Write the equations of motion for the quad-rotor assumed to be a point mass that can only move along a vertical line and is subjected to the action of two forces: gravity and thrust.

2. Knowing that there are 4 propellers, and that each propeller produces a trust force given by:

$$C_T w^2;$$

where the thrust coefficient C_T is given by $C_T = 1.536*10\^7\ Ns^2$ and w is the propeller's rotational speed, compute the value of w required to make the quad-rotor hoover when its total mass is 0:027 Kg.

3. With the objective of tracking step altitude commands, linearize the equations of motion by treating w as the control input.

4. Design a controller tracking steps inputs and show that it achieves a rise time no longer than 3 seconds and a settling time no greater than 5 seconds when acting on the nonlinear plant model.

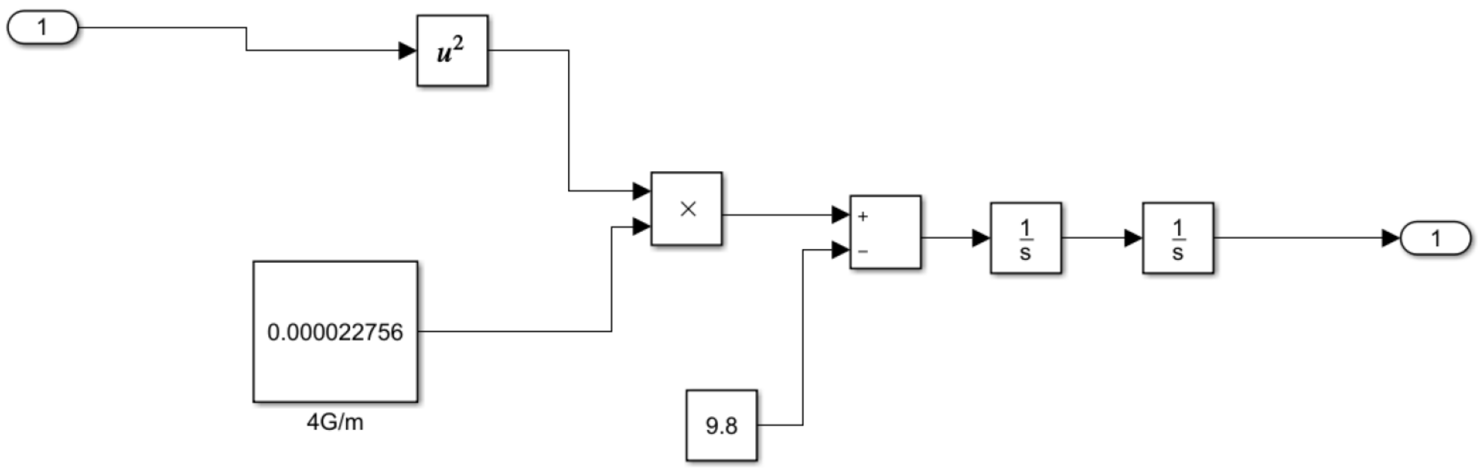


Figure1. Non-linear subsystem

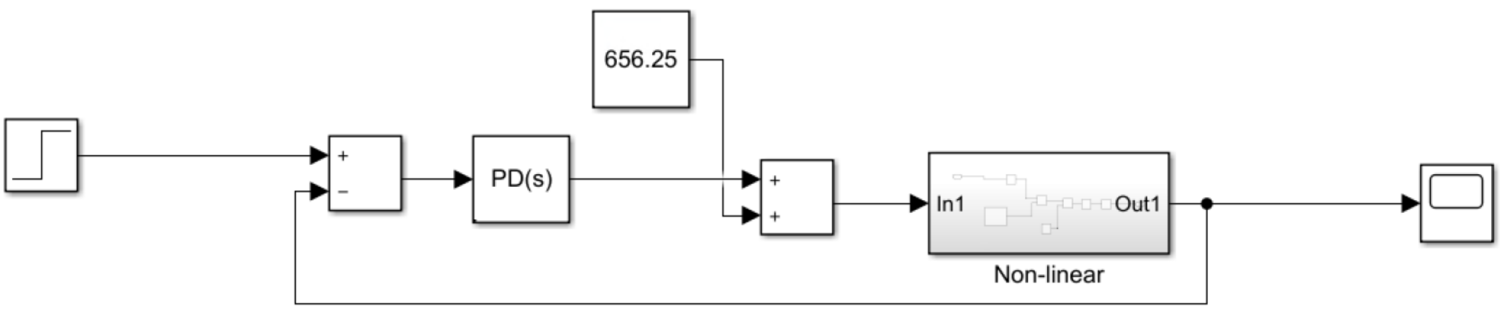


Figure 2. PD controller and non-linear system.

In the non-linear subsystem, the non-linear function has second derivation on the height (distance). Therefore, we need to use two integrals (1/s) to get the distance (y). u^2 is the input rotational speed. 9.8 is the gravity acceleration. Base on the calculation, we design a non-linear system as above.

Base on the calculation, in the linear function, $w_e = 656.25$ rad/s and $x_e = 0$. The input and output are Δw and Δx. we need to add 656.25 after the controller to make the Δw to be w in the non-linear system. Because the x_e is 0, we don’t need to do the subtract for the output.

Base on the calculate, we try varying controller values and find out that the Controller $K_P = 60$, $K_D = 70$;

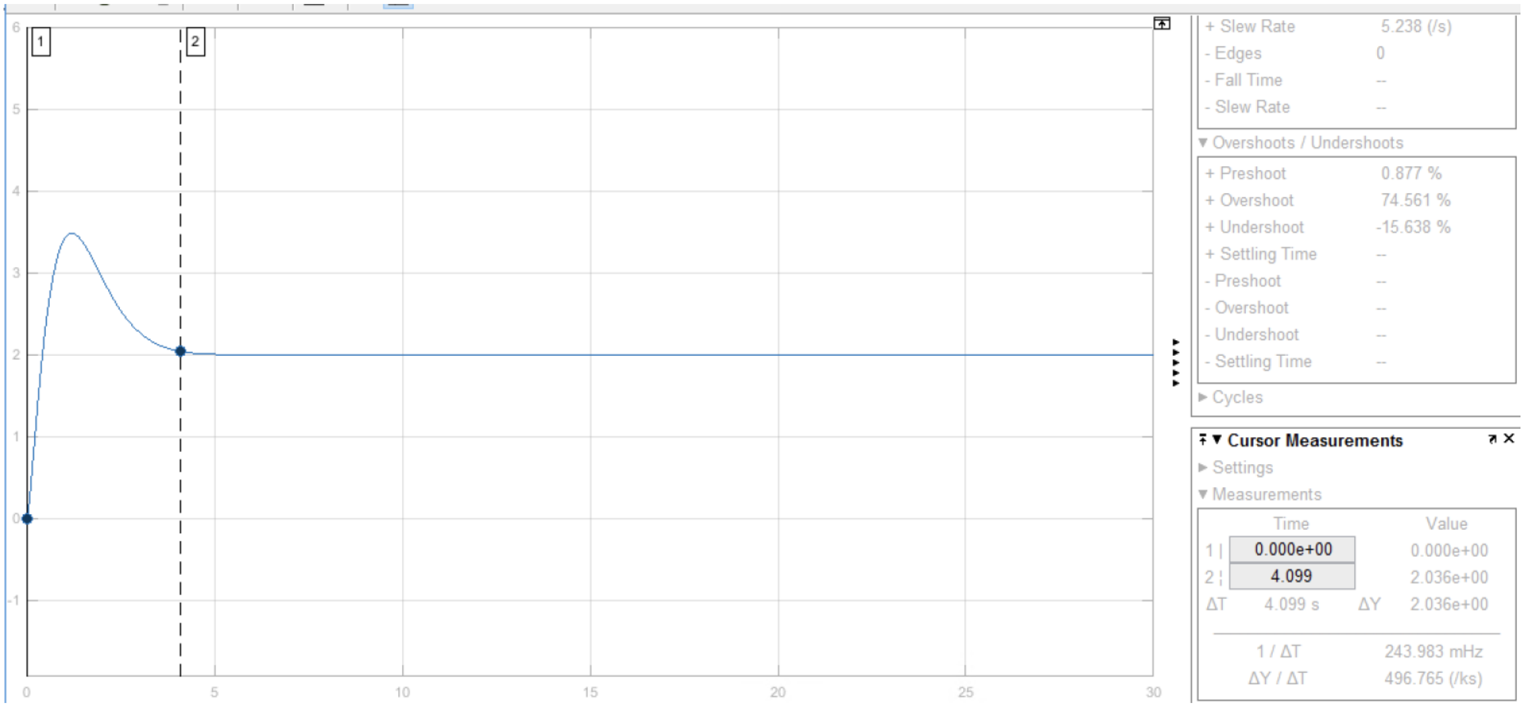


Figure 3. Non-linear: Step input

From the scope graph, the Rise time is 0.304s, and the Settling Time is about 4 second. This controller (PD) satisfies the requirement which the rise time is less than 3 second and settling time is less than 5 second.

5. Design a controller that tracks ramp commands.

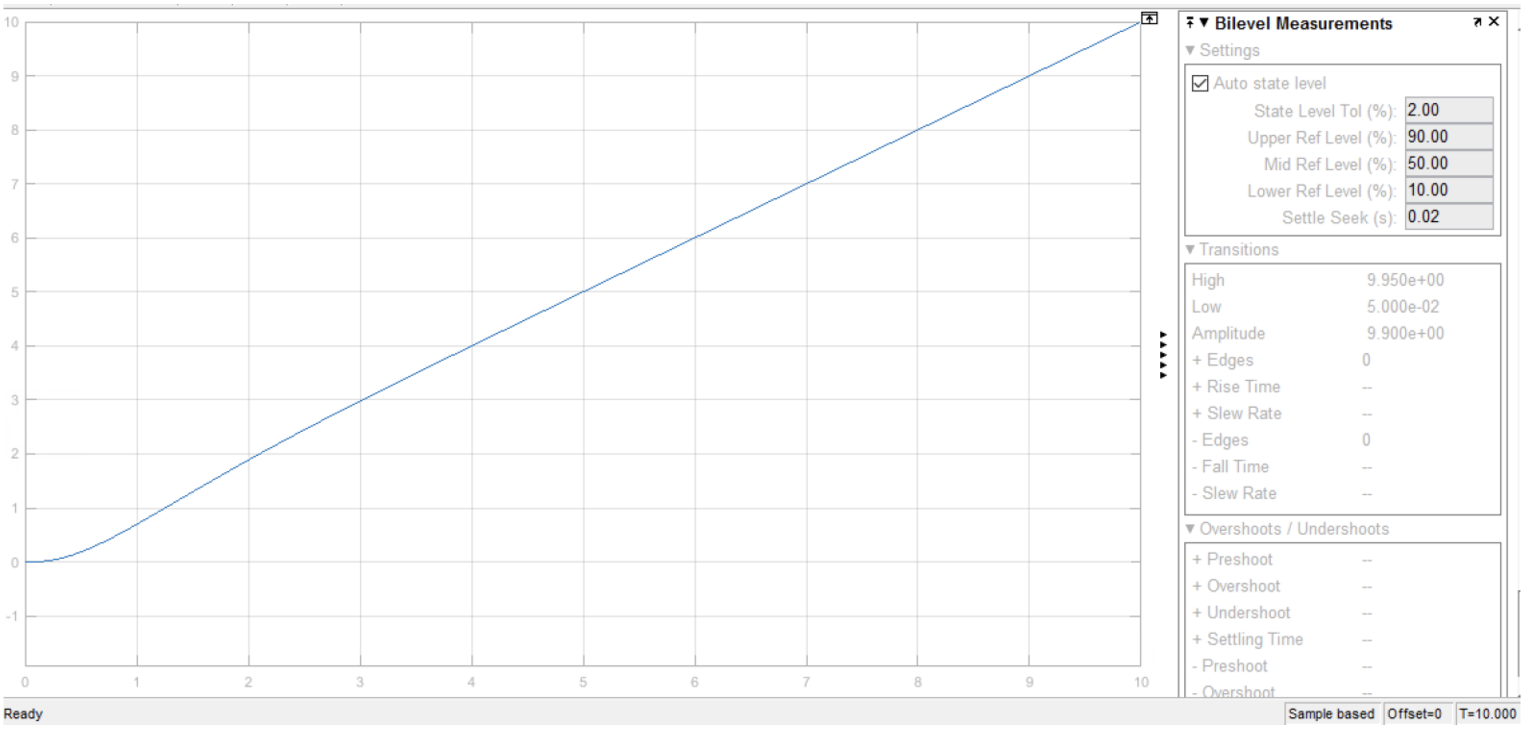


Figure 4. Non-linear: Ramp input

For part 5, we change the input to be ramp. However, the controller (PD) $K_P = 60$, $K_D = 70$ also works for ramp input, which can track ramp commands.

Therefore, we can only use one controller for the step and ramp inputs.

Part 6:
Compare the performance of the following two control strategies with the objective of making the quad-rotor hover at 2 meters:

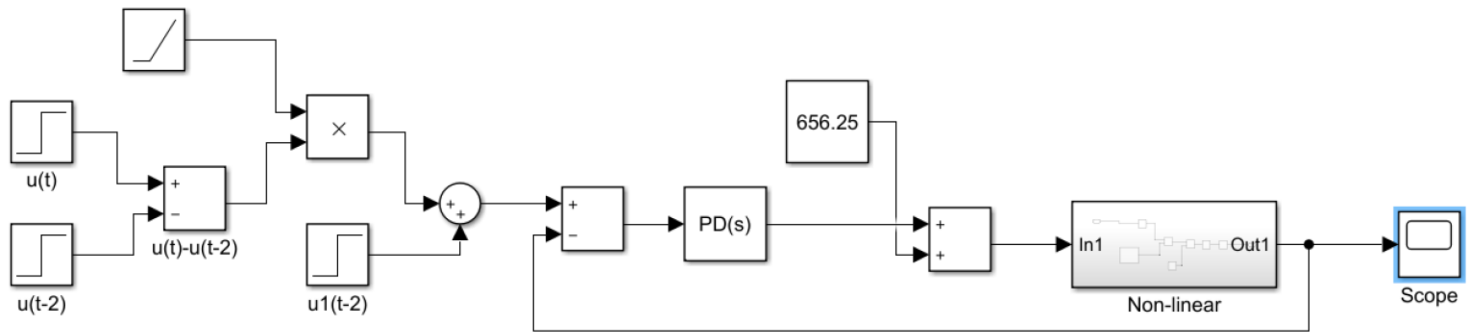
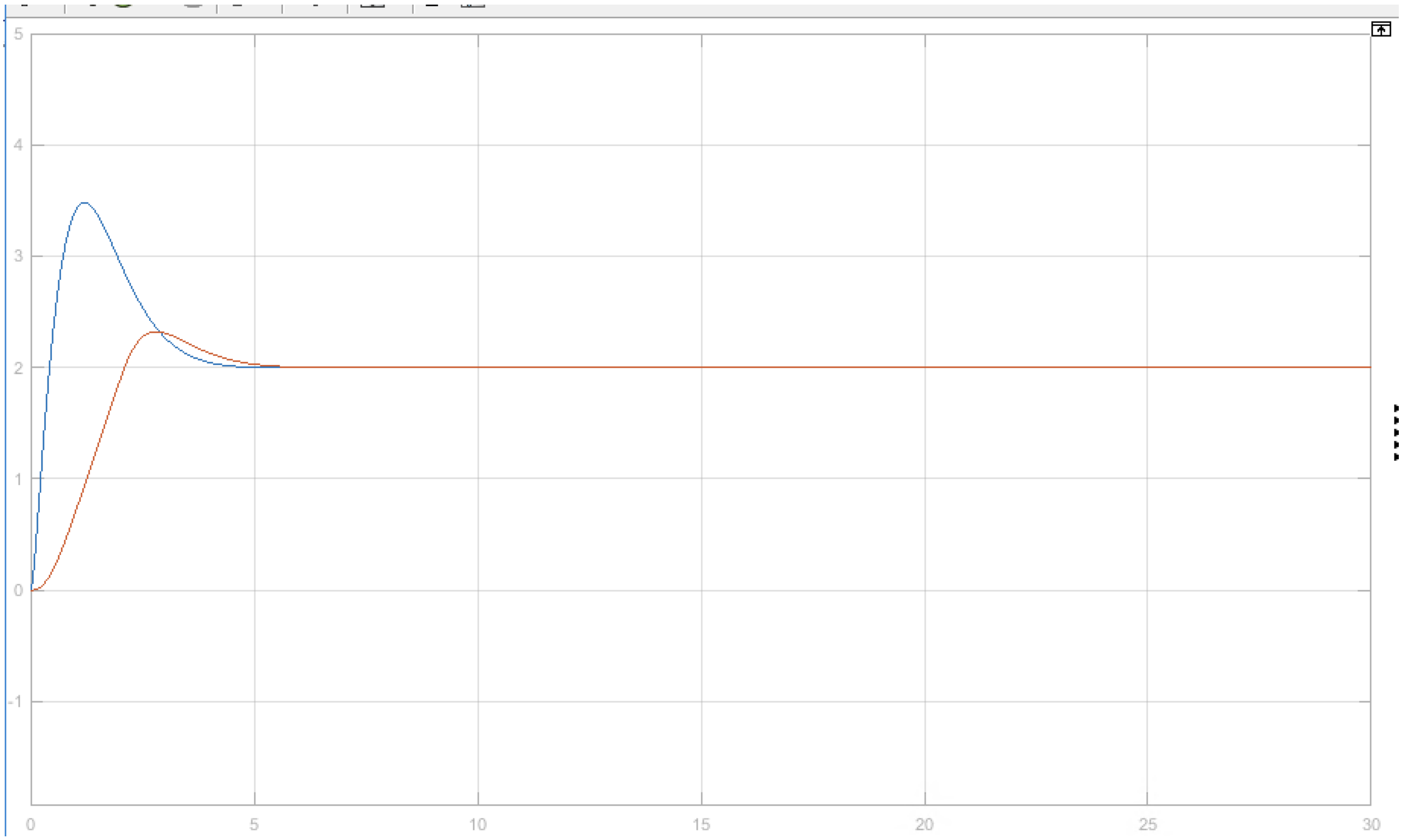


Figure 5. Ramp input followed by step input

(a) Give a step input of 2 meters;
(b) Give a ramp input followed by a step input of 2 meters (you are free to design the value of the ramp and the time at which the reference changes from a ramp to a step).

For part (b), I set the time range for ramp input is from 0 to 2 sec. After 2 sec, the input becomes step function and its amplitude is 2 which is correspond to 2 meters.



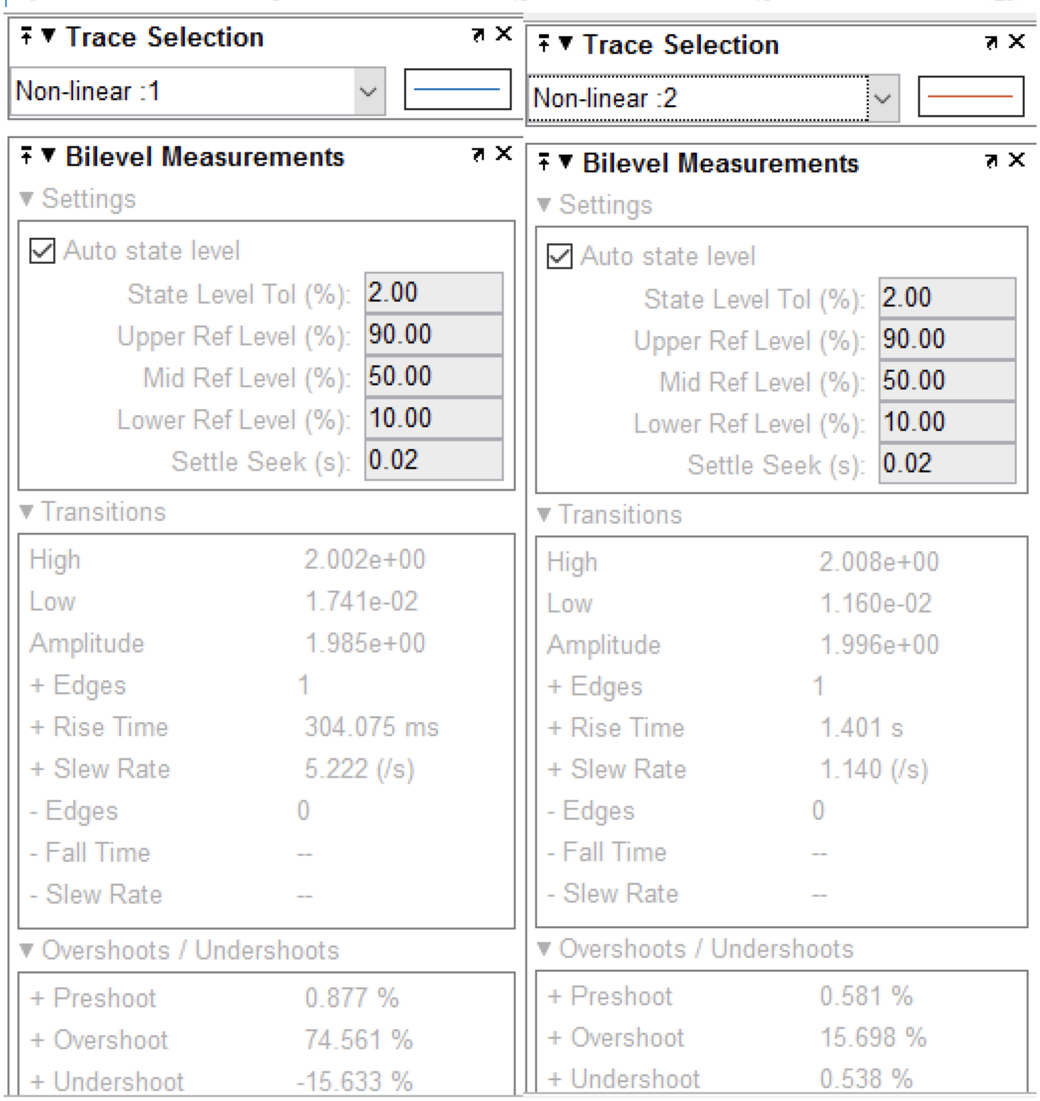


Figure 6. Scope of Ramp input followed by step input

In order to generate an input including a ramp input followed by a step input, I use u(t)-u(t-tc) times a ramp input, then I give a step input 2u(t-tc) after the ramp input. In this diagram, I set tc is 2 which means that after 2 second, the input becomes to a step function.

At the beginning, the slope of the blue line which is the step input only is higher than the slope of the red line which is the ramp input followed by step input. By sending a ramp input followed by step input, the drone can be able to rise smoothly. However, if we just send a step input signal to the system, the drone would rise rapidly. Even though it can stabilize at 2 meters, the drone should rise stably in order to have a better experience. From the data, we can also see that the blue line has higher overshoot compared to the red line. The step input only signal would make it more unstable when it reaches the final position. Therefore, it’s better to use a ramp input followed by step input as an input signal.

Part7)

7. Compare again the previous strategies when the altitude sensor is affected by noise according to the model:

$$y(t) = h(t) + n(t);$$

where y is the sensor output, h is the drone's altitude, and n(t) 2 [-0.1, 0.1] is an arbitrary time-varying signal modeling sensor noise.

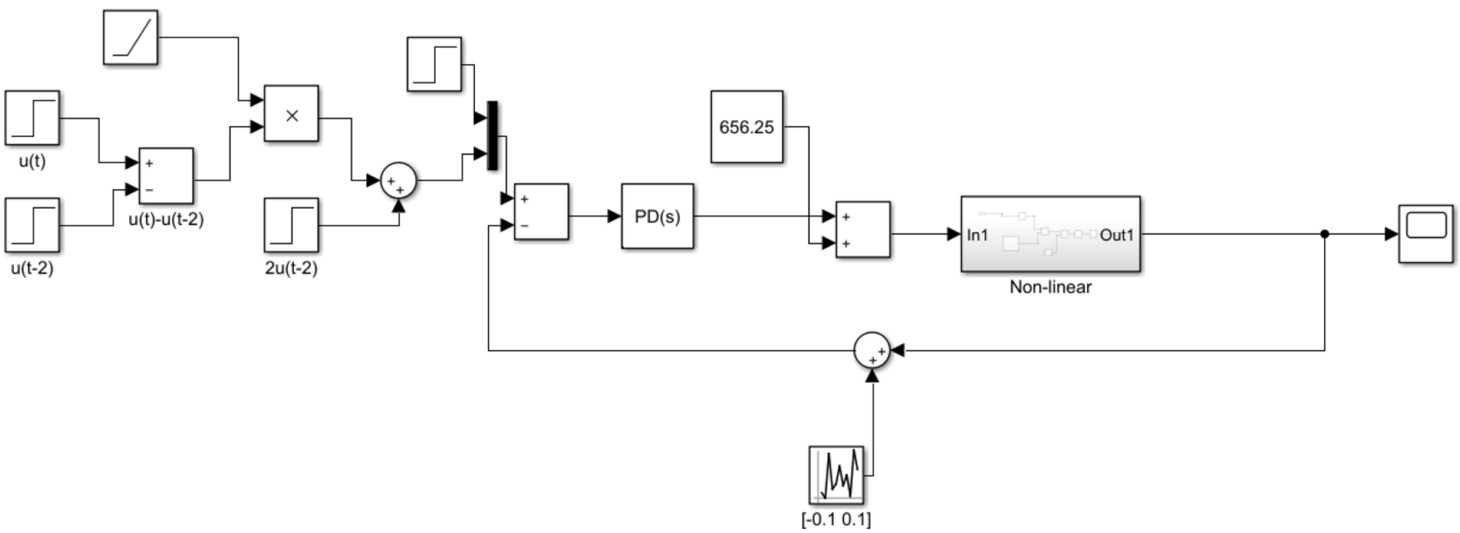
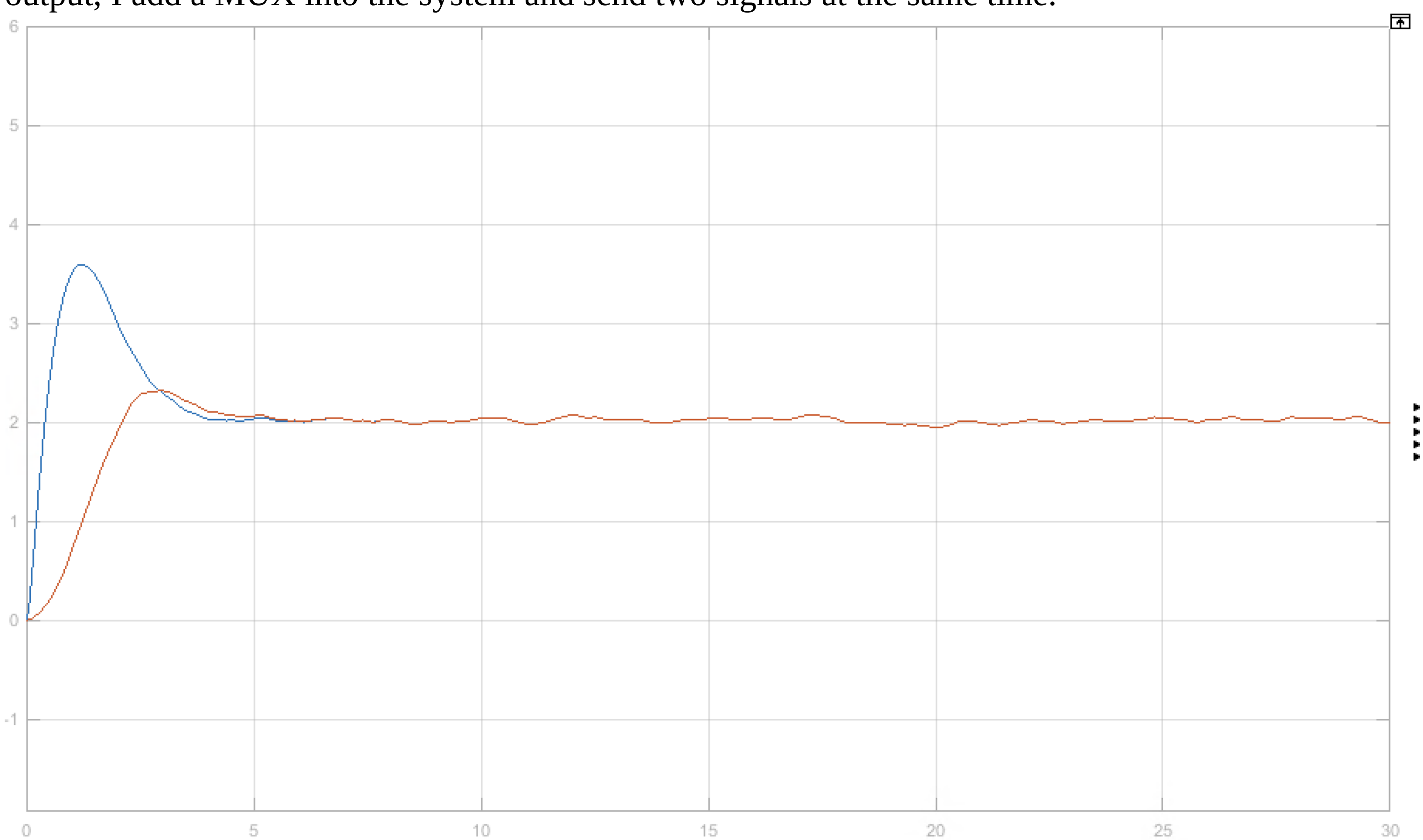
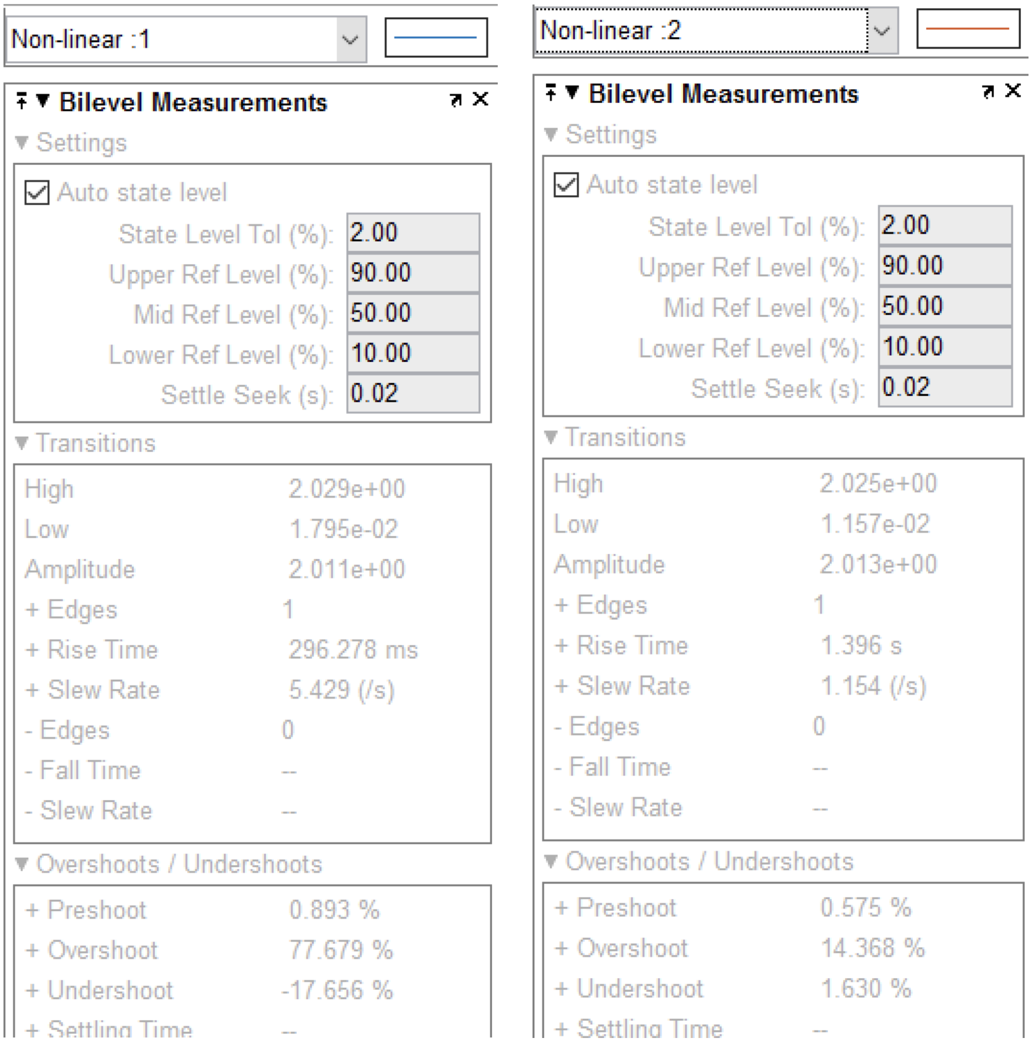


Figure 7. the system with arbitrary time-varying signal modeling sensor noise

In this part, I add the noise [-0.1 0.1] into the feedback loop. In order to compare two signals output, I add a MUX into the system and send two signals at the same time.





Even though it is affected by the noise, the output shows that the drone still can be able to stabilize at about 2 meters. Although it is floating at the final position, it doesn't make a significant effect. In reality, the drone must be affected by a lot of noise signal. It's not possible for a drone to stay at the same high. It's allowed to have some floats. However, the general shapes are similar to the graph from part 5. Therefore, this controller (PD) can be handle all of these conditions and make the drone stable.