Hand-In Exercise 3: State Space Control of Flying Drone

Emil Asgeirsson (emasg18), Kasper Mikkelsen (kamik18) & Mads Christiansen (mach018)

1 System Analysis

1.1 State Space Model

Defining a state space model of the system. The system is sketched in Figure 1 and written in equation (1).

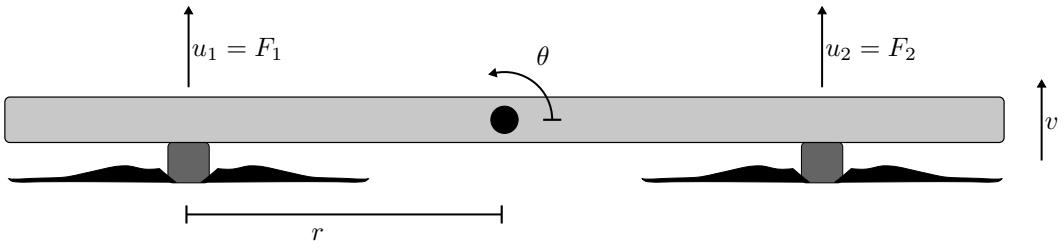


Figure 1: Diagram of flying drone that is controlled via two propellers, which apply forces F_1 and F_2 to the drone.

Parametes: $b_1 = 1[\frac{\text{Nm}}{\frac{\text{rad}}{\text{s}}}]$, $b_2 = 0.5[\frac{\text{Nm}}{\frac{\text{m}}{\text{s}}}]$, $b_3 = 1[\frac{\text{N}}{\frac{\text{m}}{\text{s}}}]$, $m = 5$ [kg], $J = 1$ [kgm^2],r = $1[m]$

$$\begin{aligned} J\ddot{\theta} &= r(u_2 - u_1) - b_1\dot{\theta} - b_2 v \\ m\dot{v} &= u_1 + u_2 - b_3 v \end{aligned} \tag{1}$$

The formula we use is:

$$\dot{x} = Ax + Bu \tag{2}$$

In order to write the equation (2), we first need isolate $\ddot{\theta}$ and $\dot{v}$ from equation (1).

$$\ddot{\theta} = \frac{r(u_2 - u_1)}{J} - \frac{b_1\dot{\theta}}{J} - \frac{b_2 v}{J} \tag{3}$$

$$\dot{v} = \frac{u_1}{m} + \frac{u_2}{m} - \frac{b_3 v}{m} \tag{4}$$

Defines our x values:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \theta \\ p \\ \dot{\theta} \\ \dot{p} \end{bmatrix} \tag{5}$$

Defines our $\dot{x}$ values:

$$\dot{x} = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} \dot{\theta} \\ \dot{p} \\ \ddot{\theta} \\ \ddot{p} \end{bmatrix} \tag{6}$$

Combining equation (1), equation (5) and equation (6), into the state space form we get:

$$\begin{bmatrix} \dot{\theta} \\ \dot{p} \\ \ddot{\theta} \\ \ddot{p} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{b_1}{J} & 0 & -\frac{b_2}{J} \\ 0 & 0 & 0 & -\frac{b_3}{m} \end{bmatrix} \cdot \begin{bmatrix} \theta \\ p \\ \dot{\theta} \\ \dot{p} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -\frac{r}{J} & \frac{r}{J} \\ \frac{1}{m} & \frac{1}{m} \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{7}$$

To define our y, we use the following formula:

$$y = Cx + Du \tag{8}$$

We here define C and D to be:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 & 0 \end{bmatrix} \tag{9}$$

C is defined to measure the angle θ and the position p of the drone. D is zero because we have no direct feed through in the system.

Writing in equation (7), and equation (9). Since the D matrix is zero, the last part of equation (8), is zero and not included in this color box.

$$\dot{x} = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} \dot{\theta} \\ \dot{p} \\ \ddot{\theta} \\ \ddot{p} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{b_1}{J} & 0 & -\frac{b_2}{J} \\ 0 & 0 & 0 & -\frac{b_3}{m} \end{bmatrix} \cdot \begin{bmatrix} \theta \\ p \\ \dot{\theta} \\ \dot{p} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -\frac{r}{J} & \frac{r}{J} \\ \frac{1}{m} & \frac{1}{m} \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \theta \\ p \\ \dot{\theta} \\ \dot{p} \end{bmatrix}$$

where m is the drone mass in kg, b_1 is our drag coefficients in $\frac{\text{Nm}}{\text{rad/s}}$, and b_2 is our drag coefficients in $\frac{\text{Nm}}{\text{m/s}}$, and b_3 is our drag coefficients in $\frac{\text{Nm}}{\text{m/s}}$. Coefficient r is half the spacing between the propellers in m, and J is the inertia in kgm. The forces u_1 & u_2 is our forces F_1 & F_2 measured in $\frac{\text{m}}{\text{s}^2}$.

1.2 Observability and Controlability

We Determine the observability and controlability by making the observability matrices, $\mathcal{O}$ & $\mathcal{C}$,

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \end{bmatrix} \tag{10}$$

Calculates the rank of equation (15) in matlab, and get the rank to be 4. To calculate the cannonical controlability matrix

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B & A^3B \end{bmatrix} \tag{11}$$

The matrices to use:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{b_1}{J} & 0 & -\frac{b_2}{J} \\ 0 & 0 & 0 & -\frac{b_3}{m} \end{bmatrix} \tag{12}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -\frac{r}{J} & \frac{r}{J} \\ \frac{1}{m} & \frac{1}{m} \end{bmatrix} \tag{13}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{14}$$

Using equation (10) gives the following equation:

$$\mathcal{O} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{b_1}{J} & 0 & -\frac{b_2}{J} \\ 0 & 0 & 0 & -\frac{b_3}{m} \\ 0 & 0 & 0 & \frac{b_2}{J}\frac{b_3}{m} - \frac{b_1}{J} \\ 0 & 0 & 0 & \frac{{b_3}^2}{m^2} \end{bmatrix} \tag{15}$$

Using equation (11), we get:

$$\mathcal{C} = \begin{bmatrix} 0 & 0 & -\frac{r}{J} & \frac{r}{J} & -\frac{b_2}{J}\frac{}{m} & -\frac{b_2}{J}\frac{}{m} & -\frac{\frac{b_1}{J} - \frac{b_2}{J}\frac{b_3}{m}}{m} & -\frac{\frac{b_1}{J} - \frac{b_2}{J}\frac{b_3}{m}}{m} \\ 0 & 0 & \frac{1}{m} & \frac{1}{m} & -\frac{b_3}{m^2} & -\frac{b_3}{m^2} & \frac{{b_3}^2}{m^3} & \frac{{b_3}^2}{m^3} \\ -\frac{r}{J} & \frac{r}{J} & -\frac{b_2}{J}\frac{}{m} & -\frac{b_2}{J}\frac{}{m} & -\frac{\frac{b_1}{J} - \frac{b_2}{J}\frac{b_3}{m}}{m} & -\frac{\frac{b_1}{J} - \frac{b_2}{J}\frac{b_3}{m}}{m} & \frac{\frac{b_1}{J}\frac{b_3}{m} - \frac{b_2}{J}\frac{{b_3}^2}{m^2}}{m} & \frac{\frac{b_1}{J}\frac{b_3}{m} - \frac{b_2}{J}\frac{{b_3}^2}{m^2}}{m} \\ \frac{1}{m} & \frac{1}{m} & -\frac{b_3}{m^2} & -\frac{b_3}{m^2} & \frac{{b_3}^2}{m^3} & \frac{{b_3}^2}{m^3} & -\frac{{b_3}^3}{m^4} & -\frac{{b_3}^3}{m^4} \end{bmatrix} \tag{16}$$

Calculates the rank of equation (15) and (16) in matlab, we get the rank 4.

Inserting equation (15) with the parameters, we get $\mathcal{O}$, and using equation (16) with the parameters, we get $\mathcal{C}$.

$$\mathcal{O} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & -\frac{1}{5} \\ 0 & 0 & 0 & -\frac{9}{10} \\ 0 & 0 & 0 & -\frac{1}{25} \end{bmatrix}$$

$$\mathcal{C} = \begin{bmatrix} 0 & 0 & -1 & 1 & -\frac{1}{10} & -\frac{1}{10} & -\frac{9}{50} & -\frac{9}{50} \\ 0 & 0 & \frac{1}{5} & \frac{1}{5} & -\frac{1}{25} & -\frac{1}{25} & \frac{1}{125} & \frac{1}{125} \\ -1 & 1 & -\frac{1}{10} & -\frac{1}{10} & -\frac{9}{50} & -\frac{9}{50} & \frac{9}{250} & \frac{9}{250} \\ \frac{1}{5} & \frac{1}{5} & -\frac{1}{25} & -\frac{1}{25} & \frac{1}{125} & \frac{1}{125} & -\frac{1}{625} & -\frac{1}{625} \end{bmatrix}$$

Since the rank of the $\mathcal{O}$ matrix is 4, we can conclude it has full rank, and therefore is observable. We get a similar result for the $\mathcal{C}$ matrix with rank 4, and we can for that reason conclude the $\mathcal{C}$ matrix is controllable.

2 Controller Design

2.1 Integral Control

sigma poles place extended In this section we will use the following method to deduce our u. To find our poles needed for the system with a α of 5% and a settling time of 1 s we calculate σ which indicates where our poles need to be placed. After deriving the poles we create extended matrices of A, B and C to find the F_e matrix together with the poles in the place function in matlab. Given the settling time of 1 s and α of 5 % we can calculate σ using equation (17).

$$t_s = \frac{-\log\left(\alpha/100\right)}{\omega_n \zeta} \tag{17}$$

$$t_s = \frac{-\log\left(\alpha/100\right)}{\sigma} \tag{18}$$

$$1 = \frac{-\log\left(\frac{5}{100}/100\right)}{\sigma} \tag{19}$$

$$\sigma = -\log\left(\frac{5}{100}/100\right) \tag{20}$$

$$\sigma = 3.301 \tag{21}$$

Defines our poles using equation (21).

$$p = \begin{bmatrix} 2 \times [-\sigma \pm 1i] & 2 \times [-\sigma * 5 \pm 1i] & 2 \times [-\sigma * 6 \pm 1i] \end{bmatrix} \tag{22}$$

$$p = \begin{bmatrix} -3.301 - 1i & -3.301 + 1i & -16.5050 + 1i & -16.5050 - 1i & -19.8060 + 1i & -19.8060 - 1i \end{bmatrix} \tag{23}$$

Defines matrices A_e, equation (24), B_e, equation (25) and C_e, equation (26):

$$A_e = \begin{bmatrix} A & 0 \\ C & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -\frac{b_1}{J} & 0 & -\frac{b_2}{J} & 0 & 0 \\ 0 & 0 & 0 & -\frac{b_3}{m} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{24}$$

$$B_e = \begin{bmatrix} B \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -\frac{r}{J} & \frac{r}{J} \\ \frac{1}{m} & \frac{1}{m} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{25}$$

$$C_e = \begin{bmatrix} C & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{26}$$

Using Matlabs place function, $-\text{place}(A_e, B_e, poles)$, we will get F_e, equation (27), which can be reformulated into F, equation (28) and F_I, equation (29).

$$Fe = \begin{bmatrix} 512.2 & -1263.3 & 28.5 & -103 & 2947.5 & -3891.4 \\ -134.1 & -1291.5 & -9.4 & -102.9 & 2381.8 & -4229.5 \end{bmatrix} \tag{27}$$

$$F = \begin{bmatrix} 512.2 & -1263.3 & 28.5 & -103 \\ -134.1 & -1291.5 & -9.4 & -102.9 \end{bmatrix} \tag{28}$$

$$F_I = \begin{bmatrix} 2947.5 & -3891.4 \\ 2381.8 & -4229.5 \end{bmatrix} \tag{29}$$

The formula for finding u is:

$$u = F \cdot x + F_I \cdot x_I \tag{30}$$

$$u = \begin{bmatrix} F & F_I \end{bmatrix} \cdot \begin{bmatrix} x \\ x_I \end{bmatrix} \tag{31}$$

Defines the parameters F and x_e:

$$F_e = \begin{bmatrix} F & F_I \end{bmatrix} \tag{32}$$

$$x_e = \begin{bmatrix} x \\ x_I \end{bmatrix} = \begin{bmatrix} \theta \\ p \\ \dot{\theta} \\ \dot{p} \\ x_{I_1} \\ x_{I_2} \end{bmatrix} \tag{33}$$

hvor x_I er:

$$x_I = \int_0^t y(\tau) - r(\tau) d\tau \tag{34}$$

$$x_I = \begin{bmatrix} x_{I_1} \\ x_{I_2} \end{bmatrix} = \begin{bmatrix} \int_0^t y_1(\tau) - r_1(\tau) d\tau \\ \int_0^t y_2(\tau) - r_2(\tau) d\tau \end{bmatrix}$$

(35)

In equation (35) y is the output og r is the reference.
Below in figure 2 is the simulink model of the integral control.

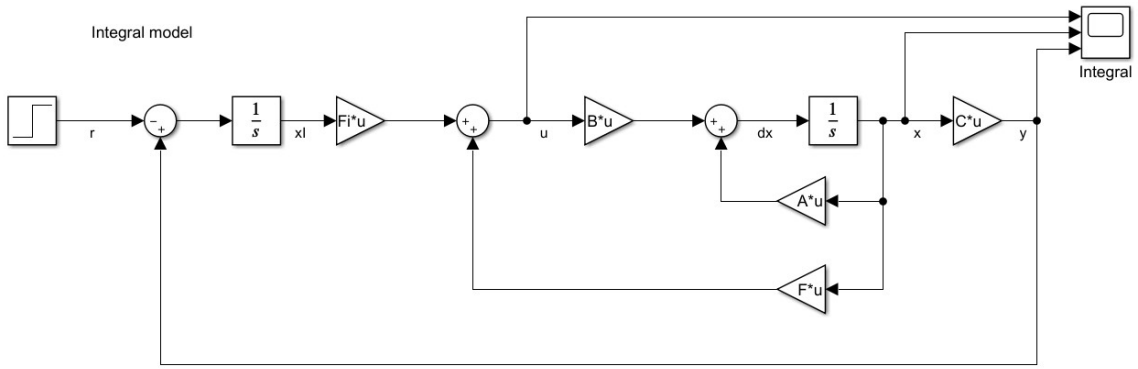


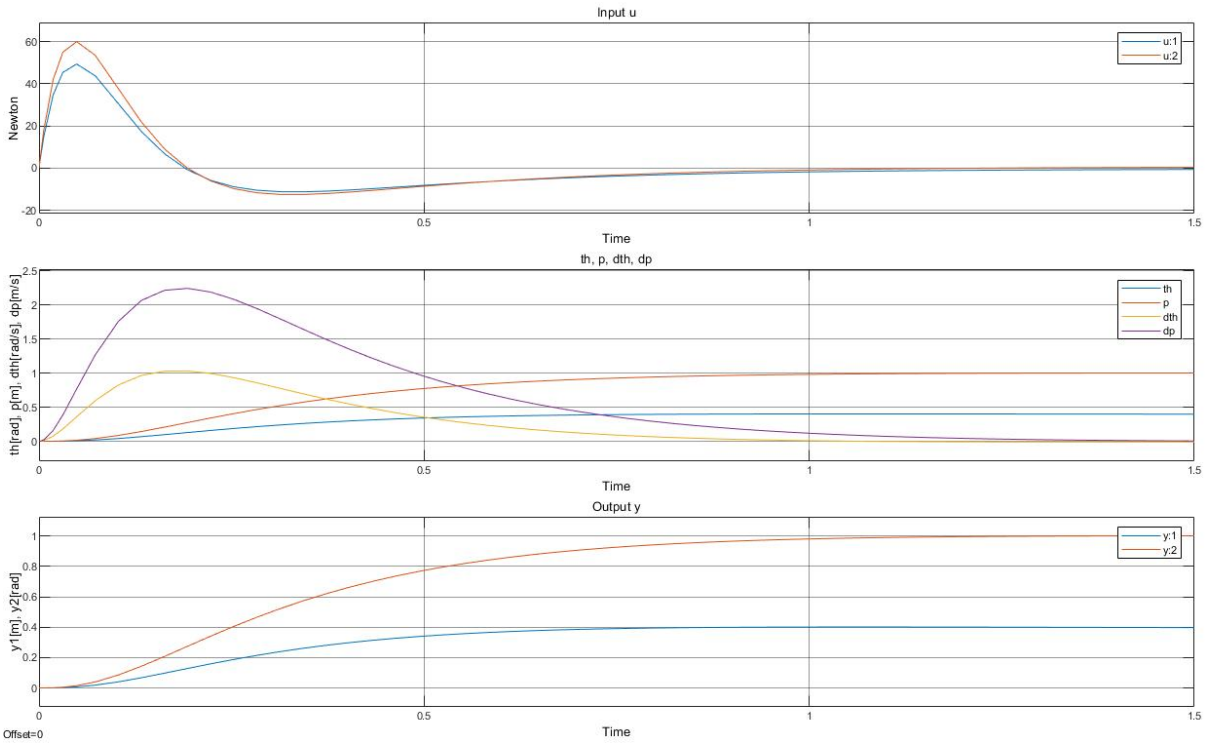
Figure 2: Simulink Integral design model

The following control law is an integral control that ensures a 5 % settling time of 1 s

$$u = \begin{bmatrix} 512.2 & -1263.3 & 28.5 & -103 \\ -134.1 & -1291.5 & -9.4 & -102.9 \end{bmatrix} \begin{bmatrix} \theta \\ p \\ \dot{\theta} \\ \dot{p} \end{bmatrix} + \begin{bmatrix} 2947.5 & -3891.4 \\ 2381.8 & -4229.5 \end{bmatrix} \begin{bmatrix} x_{I_1} \\ x_{I_2} \end{bmatrix}$$

2.2 Performance Evaluation

In Figure 3 we have plotted the system going from $(p, \theta) = (0, 0)$ to $(p, \theta) = (1, \pi/8)$.
The first graph displays the input values u_1 and u_2. The second graph displays the values of our x matrix. The third graph displays the output values y_1 [rad] and y_2 [m].



Plot of u, y, and the state x

Figure 3: Performance evaluation

Looking at the plot of the system we see that a 5% settling time of 1s is achieved. At the time 1s, output y_1 reads 0.9816 [m] and y_2 reads 0.401 [rad]. The first graph corresponds to requirements that the right side of the drone needs to be higher than the left side, where u1 (left side) is lower than u2 (right side).

3 Observer Design

3.1 Pole Selection

To select the poles for the observer design we use the same reference where we want to place our poles approximately 5-6 times further out than our sigma. Our poles have to differ from the integral design and to differ from this we have chosen the following poles:

$$O_p = \begin{bmatrix} -\sigma \cdot 5.1 + 1i & -\sigma \cdot 5.1 - 1i & -\sigma \cdot 6.1 + 1i & -\sigma \cdot 6.1 - 1i \end{bmatrix} \tag{36}$$

$$O_p = \begin{bmatrix} -16.8351 + 1.0000i & -16.8351 - 1.0000i & -20.1361 + 1.0000i & -20.1361 - 1.0000i \end{bmatrix} \tag{37}$$

3.2 Observer Design

In order to get the observer model we need to extract our L matrix. The L matrix is given by the Matlabs function place, $-\text{place}(A', B', Obs_p).'$. This is shows in equation (38).

$$L = \begin{bmatrix} -37.4875 & 0.5294 \\ 0.1651 & -36.2549 \\ -349.3967 & 22.6137 \\ -0.8813 & -323.6928 \end{bmatrix} \tag{38}$$

Adding Anti windup to the system.

$$M_p = \begin{bmatrix} -\sigma \cdot 5.5 + 1i & -\sigma \cdot 5.5 - 1i & -\sigma \cdot 6.5 + 1i & -\sigma \cdot 6.5 - 1i \end{bmatrix} \tag{39}$$

$$M_p = \begin{bmatrix} -18.1555 + 1i & -18.1555 - 1i & -21.4565 + 1i & -21.4565 - 1i \end{bmatrix} \tag{40}$$

Using Matlabs place function, $-\text{place}((A + L \cdot C)', F', M_p).'$, we will get M, equation (41).

$$M = \begin{bmatrix} -0.0017 & 0.0015 \\ -0.0004 & 0.0007 \\ 0.0323 & 0.0285 \\ 0.0080 & 0.0120 \end{bmatrix} \tag{41}$$

Below in figure 4 a simulink model of the system is illustrated.

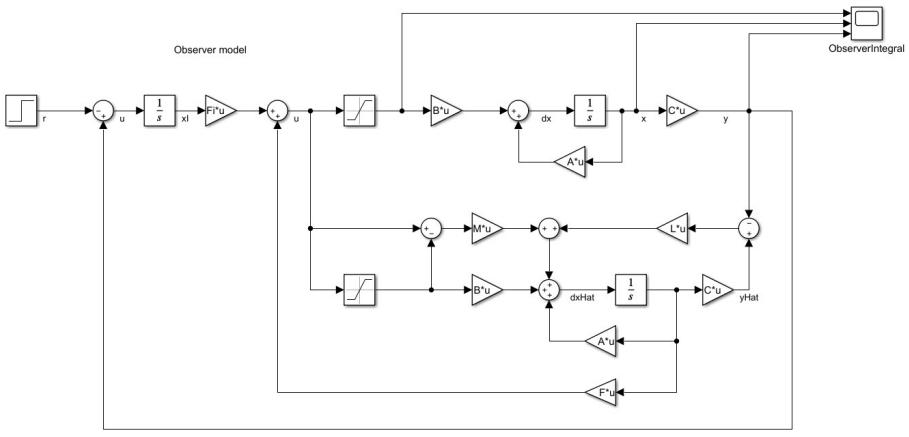


Figure 4: Simulink Observer design model with anti-windup

4 Simulation

We have simulated the observer-based controller, which is shown in figure 5, and in figure 6. The simulation is from $(p, \theta) = (0, 0)$ to $(p, \theta) = (1, \pi/8)$, and we have decided to add a saturation block at ± 40 newton, to visualize that our anti windup works. This saturation of 40 newton probably is on the lower side of the actual true performance of the actuators, but it shows that our system behave as intended.

In figure 5, we see the plant input, u, which shows the amount of power delivered from each engine. We can see that u_2 is greater than u_1 in the beginning, and this forces the drone to rotate around its center point. We can also see the states, x, where the position and angle is mowing towards the desired target in about 1 second. We also see $\dot{\theta}$ and $\dot{p}$ is high when we move towards the target, and levels out at 0 when we reach the target. Finally we see the output, y, which shows the drones orientation and height. We can see it reaches the angle and position in about 1 second as intended. We can also see it has no overshoot, and a slowly curving velocity in the beginning. We can from this plot evaluates that the performance seems reasonable for a drone with these dimensions.

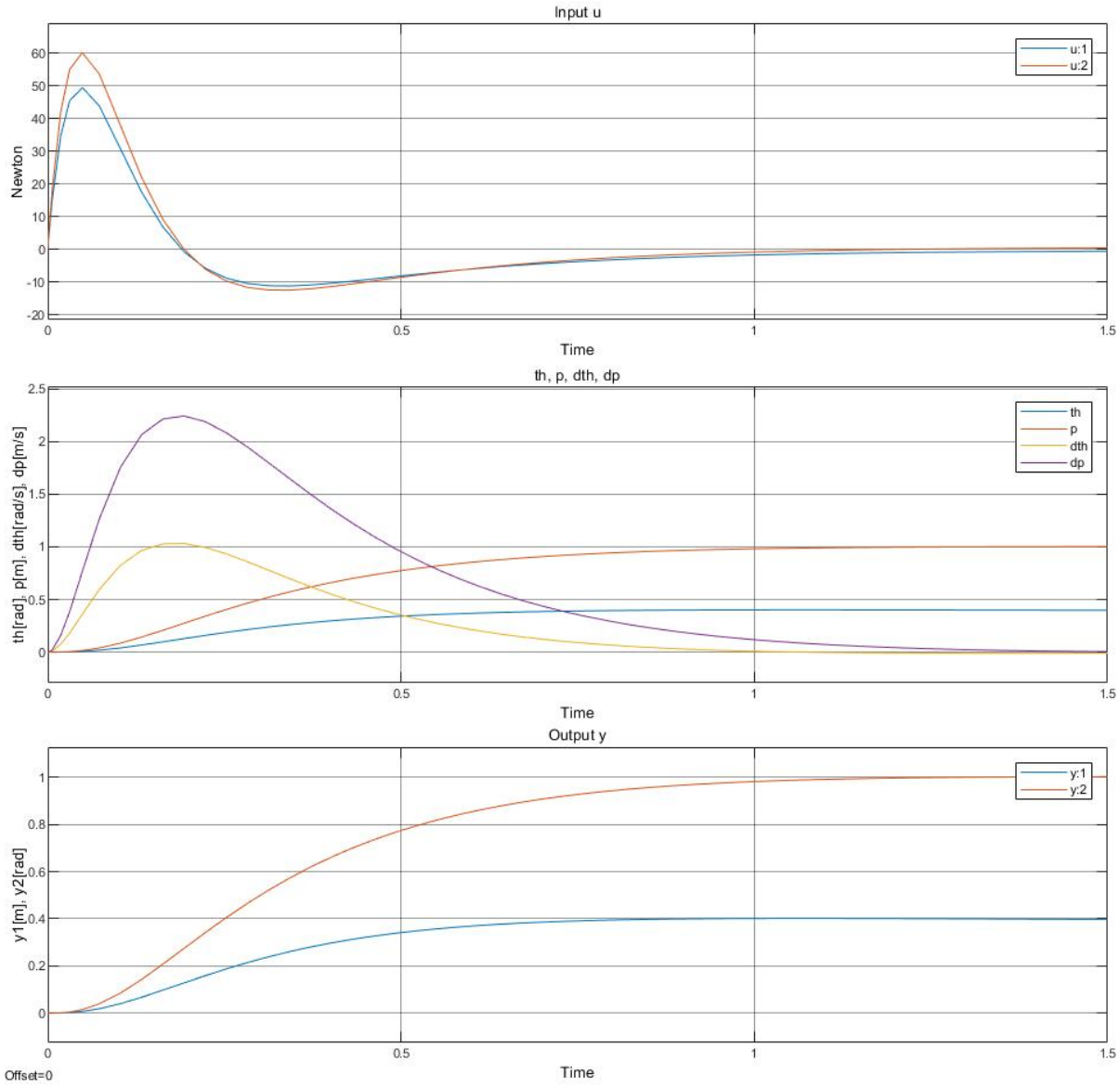


Figure 5: Simulation of the observer-based controller, simulated from $(p, \theta) = (0, 0)$ to $(p, \theta) = (1, \pi/8)$.

In figure 6, we see the plant input, u, which shows the amount of power delivered to each engine as in figure 5. We can see that u_1 and u_2 saturates at 40 newton. After around 0.1 seconds u_1 begins to drop while u_2 remains saturated, and this forces the drone to rotate around its center point. It then decreases to a lower value than u_1, which stabilizes the rotation. We can also see the states, x, where the position and angle is mowing towards the desired target in about 1 second. We also see $\dot{p}$ is high when we move towards the target, and levels out at 0 when we reach the target, as well with $\dot{\theta}$. But $\dot{\theta}$ is low in the beginning, due to the saturation block, which prohibits u_2 in rotating the drone. Finally we see the output, y, which shows the drones orientation and height. We can see it reaches the angle and position in about 1 second as intended. We also see that rotation of the drone starts after around 0.2 seconds, and increases the rotation speed faster than in figure 5, since it need to catch up to satisfy the requirement.

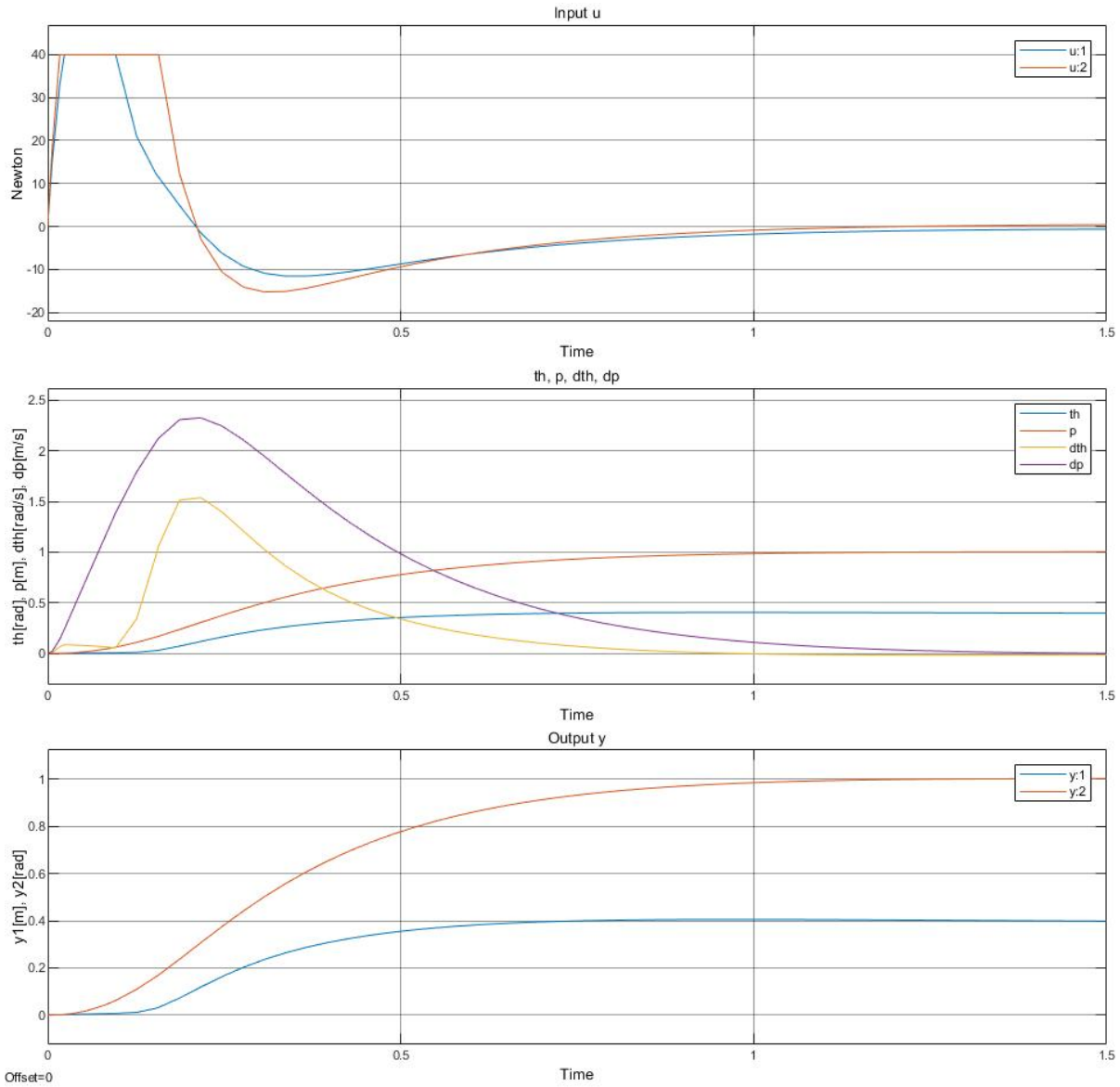


Figure 6: Simulation of the observer-based controller with anti-windup, simulated from $(p, \theta) = (0, 0)$ to $(p, \theta) = (1, \pi/8)$.