Robotic Control Theory
Coursework 2

Zinan Liu, ID: 20173922
Zhiyuan Song, ID: 20053603
Shenheng Yu, ID:180043999

November 13, 2020

1 Question 1

1.1 a

It is convenient for the computation to have linear equation of motion in the inertial frame. Moreover, the rotational equations of motion are useful to us in the body frame, so that we can express rotations about the center of the quadcopter. This strategy can help to achieve this purpose and the input can be computed by τ and T.

Also, by adapting this strategy, the dynamical system of the quadcopter is controlled without directly making use of the input, therefore we are able to analyze the system with the Lyapunov stability.

1.2 b

A similar strategy described in the lecture slides that is similar to this strategy is autonomous system, which is also controlled without directly making use of the input.

2 Question 2

2.1 a

Linearisation. As the system is a nonlinear system, to present the model in the state space, the model is needed to be linearised around the operating point, meaning that, the system is approximated to a linear function around the operating point in a certain range.

2.2 b

To reduce the approximation error, reduce the range of approximation of linearisation. In the implementation, it can be achieved by reduce the dt to a small enough number.

2.3 c

One method to update the state without using this approximation is using the MATLAB toolbox to compute continuous integration over time instead of taking discrete time sample and calculate, thus avoiding using approximation of dt.

Another way to interpret this question without using the state space equation in the form $\dot{x} = Ax + Bu$, which implement the linearisation around the operating point, the state equations from page 8 in [1] is an approach. The state was updated in a nonlinear model system which is the exact model, hence the update is the exact update. If the state is updated without using approximation, we will use angle derivatives as input to update the state:

$$\omega = \begin{bmatrix} 1 & 0 & s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\theta c_\phi \end{bmatrix} \dot{\theta}$$

Where ω s the angular velocity vector in the body frame and $\dot{\theta}$ is the time derivative of yaw, pitch and roll. And thus, we can update the angular acceleration by:

$$\omega = I^{-1}(-(I))$$

Where τ is the vector of external torques and calculated by:

$$\tau_B = \begin{bmatrix} Lk({\omega_1}^2 - {\omega_3}^2) \\ Lk({\omega_2}^2 - {\omega_4}^2) \\ b({\omega_1}^2 - {\omega_2}^2 + {\omega_3}^2 - {\omega_4}^2) \end{bmatrix}$$

Where ω_i is the angular velocity of the motor of the i_{th} propeller.

3 Question 3

3.1 a

From the external reference [2], the parameter is determined as The parameters are based from

b	0.007
k	1
k_d	0.1
m	1 kg
g	9.81 ms^{-2}
L	0.25 m
I_{xx}	0.04 kgm^2
I_{yy}	0.04 kgm^2
I_{zz}	0.008 kgm^2

Table 1: Parameter Defined

the real product from the market, and it is able to complete the tasks in question 5, therefore it is a reasonable setup.

3.2 b

In this question, the aim is to stabilize the quadcopter at the equilibrium point where $\omega_x = \omega_y = \omega_z = \phi = \theta = \psi = 0$. With the predetermined parameter, the linear model is represented in the state space model $\dot{x} = Ax + Bu, y = Cx$ and the state space model is

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = \begin{bmatrix} O_{3x3} & I_{3x3} & O_{3x3} & O_{3x3} \\ O_{3x3} & -0.1I_{3x3} & O_{3x3} & O_{3x3} \\ O_{3x3} & O_{3x3} & O_{3x3} & I_{3x3} \\ O_{3x3} & O_{3x3} & O_{3x3} & O_{3x3} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \phi \\ \theta \\ \psi \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} + 0.001* \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 6.25 & 0 & -6.25 & 0 \\ 0 & 6.25 & 0 & -6.25 \\ 0.8750 & -0.8750 & 0.8750 & -0.8750 \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{bmatrix} \tag{1}$$

where γ_i=ω_i^2. Through MATLAB, the discrete model was achieved. Please go through the MATLAB script namely Q3b.m for detail explanation.

4 Question 4

4.1 a

As the only accessible measurements are $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$, the control inputs are shown in page 9[1]. The PD controller will only make use of the first derivatives of the angles and then can provide the derivatives of the error, and the integral gives the actual error of the system.

$$\gamma_1 = \frac{mg}{4kcos\theta cos\phi} - \frac{2be_\phi I_{xx} + e_\psi I_{zz}kL}{4bkL} \tag{2}$$

$$\gamma_2 = \frac{mg}{4kcos\theta cos\phi} + \frac{e_\psi I_{zz}}{4b} - \frac{e_\theta I_{yy}}{2kL} \tag{3}$$

$$\gamma_3 = \frac{mg}{4kcos\theta cos\phi} - \frac{-2be_\phi I_{xx} + e_\psi I_{zz}kL}{4bkL} \tag{4}$$

$$\gamma_4 = \frac{mg}{4kcos\theta cos\phi} + \frac{e_\psi I_{zz}}{4b} + \frac{e_\theta I_{yy}}{2kL} \tag{5}$$

where $e_{angle} = K_d \dot{angle} + K_p \int_0^T \dot{angle}\, dt$, based on the page 8 and 9 from [1]
To derive an output equation $y = f(x(t))$ using the gyroscope measurements as input, the output y_θ can be set as the roll, pitch and yaw angles and y_ω can be set as the angular velocity,

thus the equation can be expressed as shown below:

$$y = \begin{bmatrix} y_\theta \\ y_\omega \end{bmatrix} = \begin{bmatrix} \phi \\ \theta \\ \psi \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

Therefore, the output equation can be written as:

$$y(t+\delta t) = \begin{bmatrix} y_\theta(t+\delta t) \\ y_\omega(t+\delta t) \end{bmatrix}$$

Referring back to our answer in Question 2c, where $y_\theta(t+\delta t)$ and $y_\omega(t+\delta t)$ can be expressed as:

$$y_\theta(t+\delta t) = \begin{bmatrix} \phi(t) + \dot{\phi}\delta t \\ \theta(t) + \dot{\theta}\delta t \\ \psi(t) + \dot{\psi}\delta t \end{bmatrix}$$

$$y_\omega(t+\delta t) = \begin{bmatrix} 1 & 0 & s_{\dot{\theta}\delta t} \\ 0 & c_{\dot{\phi}\delta t} & c_{\dot{\theta}\delta t} s_{\dot{\phi}\delta t} \\ 0 & -s_{\dot{\phi}\delta t} & c_{\dot{\theta}\delta t} c_{\dot{\phi}\delta t} \end{bmatrix} \dot{y}_\theta$$

Substitute the new input shown above to the u in the state space model $\dot{x} = Ax + Bu$,$y = Cx$, the output of system can be obtained.

4.2 b

If an altitude sensor and a horizontal GPS sensor are added to the quadcopter to measure the vertical and horizontal positions, then these two sensors could be combined to provide the measurement of the quadcopter's position in the 3D space, which can be written as (x, y, z). Then, a linear relationship between the states and the output $y = f(x(t))$ using all measurements including the gyroscope measurements, the altitude sensor measurements and the GPS sensor measurements as input, and the output y_θ can be set as the roll, pitch and yaw angles, y_ω can be set as the angular velocity, and y_p can be set as the 3D space position. The equation of y could be derived as:

$$y = \begin{bmatrix} y_\theta \\ y_\omega \\ y_p \end{bmatrix} = \begin{bmatrix} \phi \\ \theta \\ \psi \\ \omega_x \\ \omega_y \\ \omega_z \\ x \\ y \\ z \end{bmatrix}$$

Therefore, the output equation can be written as:

$$y = \begin{bmatrix} y_\theta(t+\delta t) \\ y_\omega(t+\delta t) \\ y_p(t+\delta t) \end{bmatrix}$$

5 Question 5

5.1 a

The task require to complete the task including rise to altitude 2.5m, following a circular trajectory then land to the original point with less than $0.1m/s$ landing speed when hitting the ground.

There are 2 functions in the code for each task, the first function response to the rising task and the second function complete the rest. The functions are shown as follow

```
function obj = change_pos_and_orientation_h_5a(obj) %function
   for levitate and stay at 2.5
function obj = change_pos_and_orientation_circle_5a(obj) %
   function for circle trajectory with radius and altitude 2.5
```

In part A, as there is no limitation to state, hence the controller can be built based on the state information. For the first task, in order to rise to 2.5m and maintain at the position $[0; 0; 2.5]$, a PID controller is applied and the error is difference between altitude of drone and the reference height $0.25m$. The system response is shown as following As shown in the figure above,

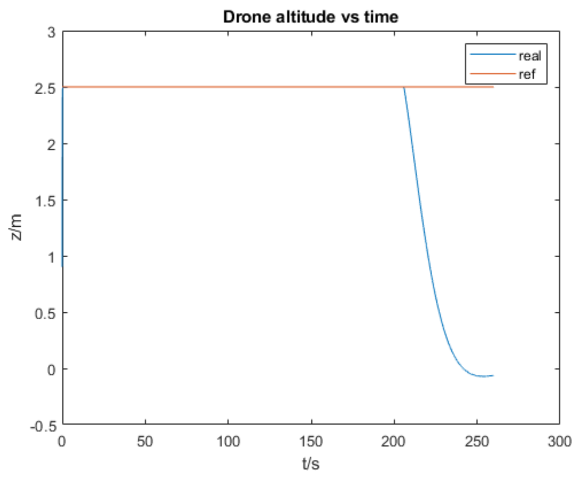
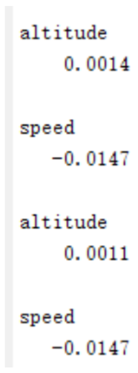


Figure 1: Drone altitude vs time

the drone has a fast and stable response system. Moreover, from the right blue curve, it can be observed that the drone is able to slow down the speed during landing when it is about to hit the ground. From the matlab data record, the landing speed when close to the ground is $-0.0147m/s$ shown in figure 2. In the landing process, the controller is switched from PID controller to PD controller with much lower Kp gain and Kd gain to guarantee the system will not overshoot, meaning that the drone will not hit the ground.

For the circular trajectory following, strategy is updating the target for every iteration and those target should be on the circle. In this case, for every iteration in the loop, we have the position of the drone, hence we have the central angle between 2 lines which are from the centre of the circle to the point $[0; 0; 2.5]$ and from the centre of the circle to the position $[x; y; 2.5]$

CONTINUED



```
altitude
    0.0014

speed
   -0.0147

altitude
    0.0011

speed
   -0.0147
```

Figure 2: landing speed

where x and y represent the position of the drone. Hence the next target can be updated as the central angle plus a small rotation angle, in our case is $\frac{pi}{25}^{\circ}$. Based on the new central angle, a new cartesian coordinate (x and y) can be calculated. The goal is to reduce the displacement between the new target and current drone position and this can be achieve by PID control.

In this case the errors are $e_x = x_{newtarget} - x_{currentposition}$ and $e_y = y_{newtarget} - y_{currentposition}$. Feed the errors to the PID controller we have the control signal as PID_x and PID_y. Based on the drone mechanics, a non zero roll angle lead to the drone fly along the y axis and a non zero pitch angle can result in a movement along x axis. In equation 2 to 5, e represent the control signal from PID controller and the input to the PID controller should be the difference between 0 and orientation angle because in the horizontal position, the roll, pitch and yall angle should be 0. To approach to the desired (x, y) position, the drone require a certain roll and pitch angle, hence the reference orientation angle should be change from 0 to PID_y for roll angle and PID_x for pitch angle, as shown in the code. However, the yaw angle is not dominant in the horizontal motion control, so its reference is set to 0.

```
 %error of orientation PID controller
phi_e = phi - posy_PID;      %phi orientation difference, (phi
   - variable) if variable (+,negative y;-,positive y)
theta_e = theta + posx_PID;  %theta orientation difference, (
   theta + variable) if variable (+,negative x;-, positive x)
psi_e = psi;                 %psi orientation difference
```

Using the PID controller on the orientation angle error above, we have the $[e_{phi}; e_{theta}; e_{psi}]$. Then the control signal can be calculated through equations 2 to 5, and the system response for the circle trajectory is

As shown in figure 3, it takes around 200 second for the drone to complete the trajectory, and the drone can follow the trajectory precisely via the PID controller.

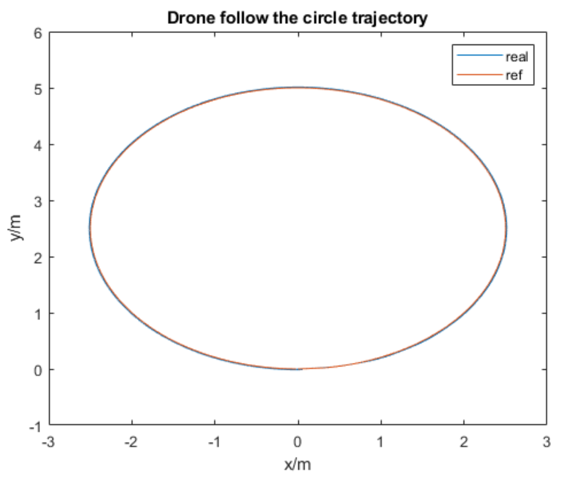


Figure 3: Round trajectory following

5.2 b

In task b, similar approach as task a, but without full state access. In this task, as the drone only equip with gyroscope, altitude sensor and GPS, meaning that only $[\dot{phi}, \dot{theta}, \dot{psi}, x, y, z]$ can be accessed. From the approach in task A, the controller require $[\phi, \theta, \psi]$, which are the roll, pitch and yall angle. It can be obtained by integral $[\dot{phi}, \dot{theta}, \dot{psi}]$. As the result same approach as task A can be implemented based on the estimated state $[\hat{\phi}, \hat{\theta}, \hat{\psi}]$. The altitude system response is

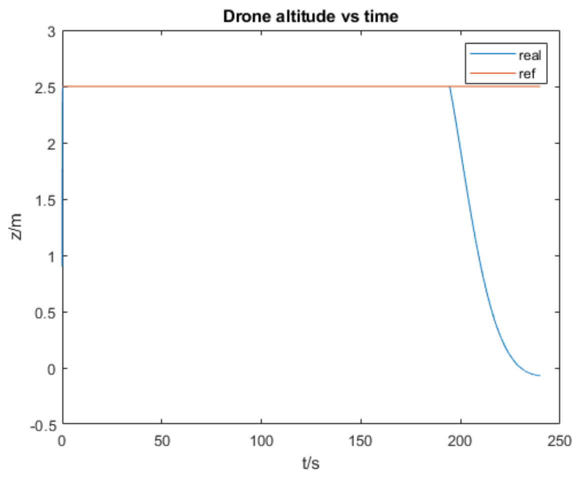


Figure 4: Drone altitude vs time(Part of state access)

Compare figure 4 with figure 1, they are highly similar to each other, meaning that the estimate state $[\hat{\phi}, \hat{\theta}, \hat{\psi}]$ is closed to the real state $[\phi, \theta, \psi]$. Note that for the right blue curve in figure 4, the gradient is rapidly reduced when approach to the ground meaning the drone significantly reduce the speed. It might be observed that the altitude slightly go over than 0 into negative and it is due to the PD controller simulation. In reality, once the drone lands on the ground, the altitude of the drone should be 0. Furthermore, the circular trajectory is shown as following

CONTINUED

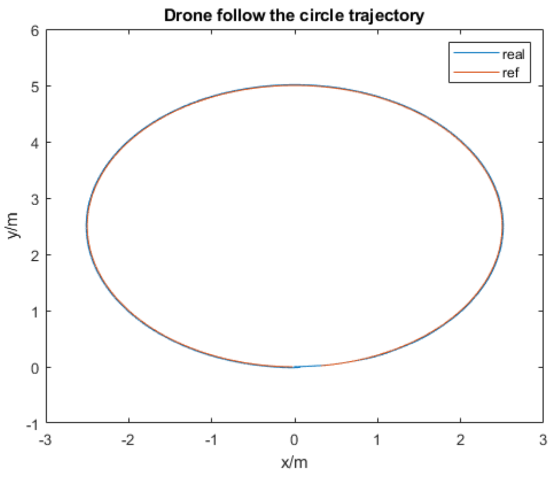


Figure 5: Round trajectory following(Part of state access)

As the control approach is same and there is a reasonable state estimation, there is a high similarity between figure 5 and figure 3.

5.3 c

The noise from sensor will affect the measurement of $[\dot{phi}, \dot{theta}, \dot{psi}, x, y, z]$. Hence in the implementation, the noise can be model by adding a Gaussian distribution random number to the accessed state, such as

```
phi_dot  =  obj.orientdot(1)+obj.noise;
theta_dot  =  obj.orientdot(2)+obj.noise;
psi_dot  =  obj.orientdot(3)+obj.noise;
obj.pos  =  obj.pos  +  obj.time_interval*obj.posdot+obj.noise;
```

The 1st 3 lines are the noise applied on $[\dot{phi}, \dot{theta}, \dot{psi}]$ and the last line is the noise to $[x, y, z]$. After several investigation, the margin of magnitude of the noise which can fail the system is between $0.0025 * Gaussiannoise$ and $0.00275 * Gaussiannoise$. The corresponding height vary along time and circle trajectory following are shown in figure 6 and figure 7

Compare figure 6 with figure 7, with the lower Gaussian, the drone can land much safer than the one with high Gaussian noise. Moreover, the situation with lower noise the drone is stable on the altitude around 0.85m, and this can be the result of over damping from PD controller during landing to guarantee the drone will not hit the ground.

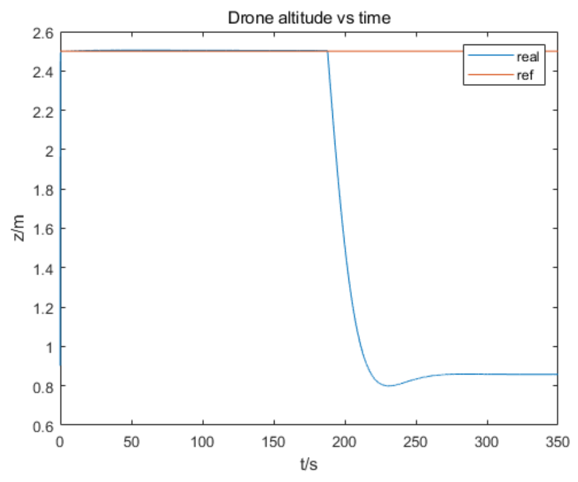


Figure 6: altitude noise 0.0025

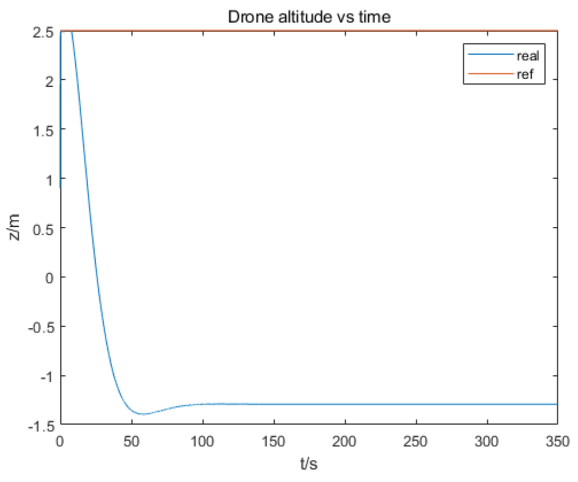


Figure 7: altitude noise 0.00275

Compare figure 8 and figure 9, it is obvious that with a $0.00275 * Gaussiannoise$ the drone is not able to complete the circle meanwhile the other one is still able to handle the task. As there is only a slightly change in the magnitude of Gaussian noise which can completely change the system response, hence it is estimated that $0.00275 * Gaussiannoise$ is the margin of the system before failing.

Figure 10 and figure 11 show the system response with a even lower Gaussian noise with magnitude 0.0001. It can be observed that in terms of the circle trajectory following and landing task, it has a much better performance compare with previous 2 cases.

CONTINUED

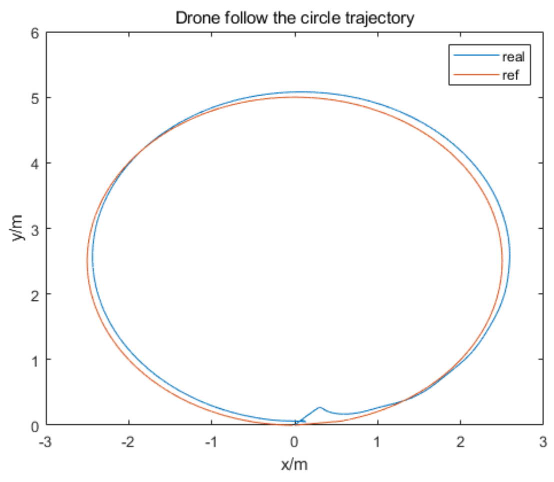


Figure 8: trajectory noise 0.0025

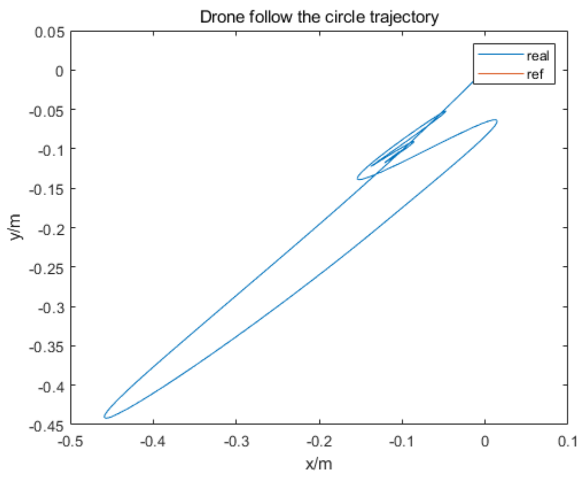


Figure 9: altitude noise 0.00275

5.4 d

In this part the wind model is consists of 2 components which are the mean wind field and the turbulent wind. Based on the reference, the mean wind field from random direction can be computed by

$$\omega_c = \begin{bmatrix} cos(wdirection) & sin(wdirection) & 0 \\ -sin(wdirection) & cos(wdirection) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{20}\frac{log(\frac{-p_z}{0.15})}{log(\frac{20}{0.15})} \\ 0 \\ 0 \end{bmatrix} \tag{6}$$

where the first matrix is the wind direction matrix and wdirection is a random angle range from 0 to 2π. The second matrix is the navigation-frame wind vector ω_c at altitude $-p_z$.

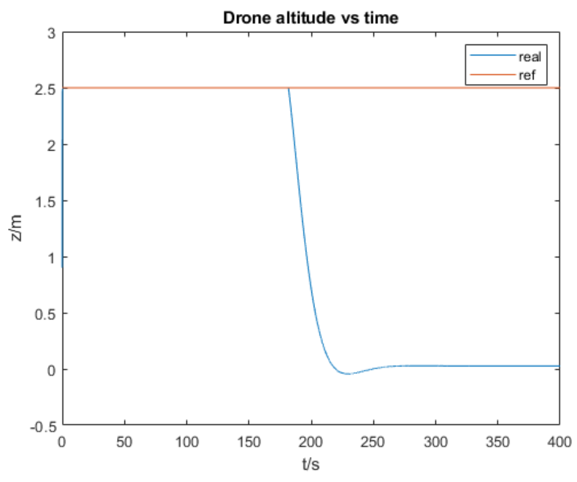


Figure 10: altitude noise 0.0001

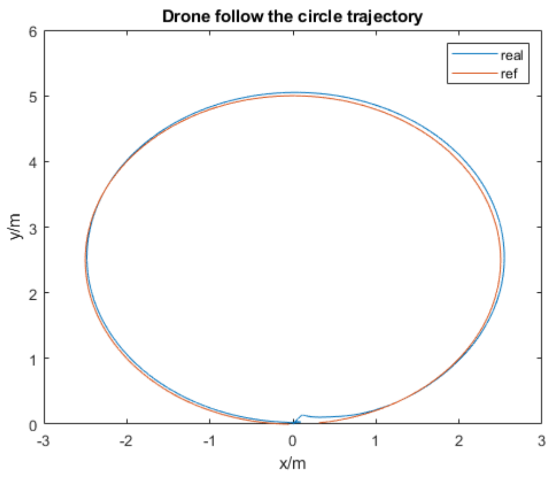


Figure 11: trajectory noise 0.0001

The turbulent wind can be computed by

$$\begin{bmatrix} \omega_u^{t+T} \\ \omega_v^{t+T} \\ \omega_w^{t+T} \end{bmatrix} = \begin{bmatrix} (1-\frac{VT}{L_u})\omega_u^t + \sqrt{\frac{2VT}{L_u}}\epsilon_u \\ (1-\frac{VT}{L_v})\omega_v^t + \sqrt{\frac{2VT}{L_v}}\epsilon_v \\ (1-\frac{VT}{L_w})\omega_w^t + \sqrt{\frac{2VT}{L_w}}\epsilon_w \end{bmatrix} \tag{7}$$

Hence the wind model can be computed by

$$windv = \omega_c + \begin{bmatrix} \omega_u^{t+T} \\ \omega_v^{t+T} \\ \omega_w^{t+T} \end{bmatrix} \tag{8}$$

where windv is the velocity of wind. To investigate the wind effect to the drone, it is required to transfer the wind velocity to wind force via equation from [**3**]

$$F = 0.5V^2 \tag{9}$$

CONTINUED

where ρ is the density of the air in kg/m^3, V is the wind velocity in m/s, A is the area in m^2, F is wind force in N. In this case A is consisted of $[A_x, A_y, A_z]$, hence we have the force in x,y and z direction. Through Newton 2nd Law, the wind force can be transfer to the acceleration and add to the linear acceleration of drone state as the disturbance.

```
% wind model acceleration update
obj.posdotdot = acceleration(obj) + obj.wind_m*a_wind;
```

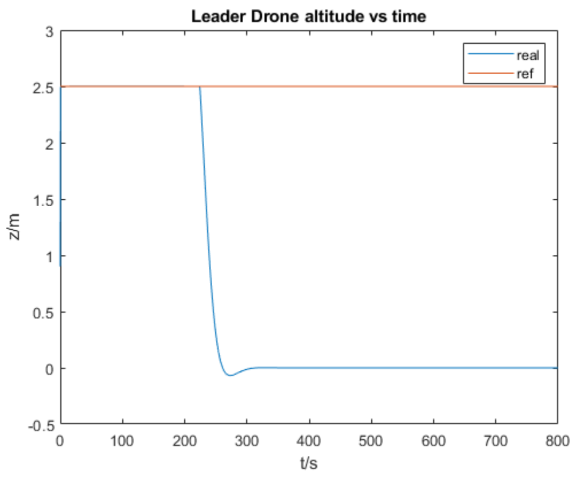


Figure 12: altitude wind 1.075

Figure 12 shows the altitude of drone along time. It can be observed that the drone is able to complete the rising and landing task describe in task a.

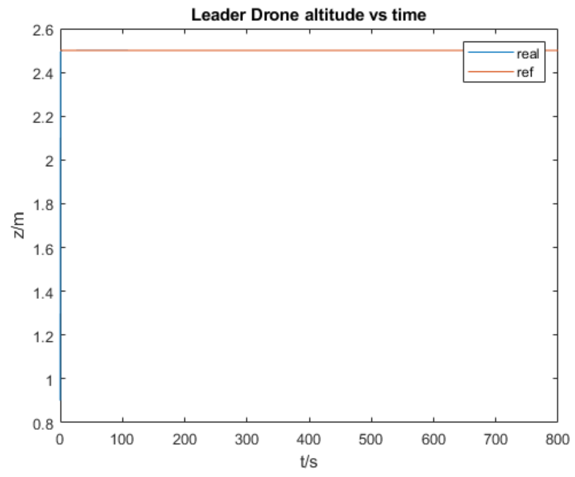


Figure 13: altitude wind 1.15

However, from figure 15, with higher magnitude of wind model, the drone is not able to land.

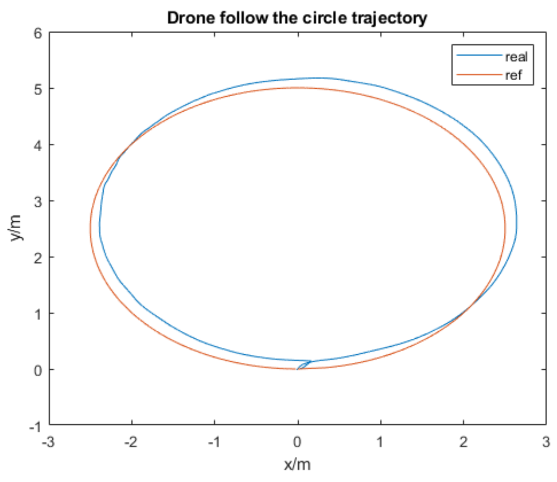


Figure 14: trajectory wind 1.075

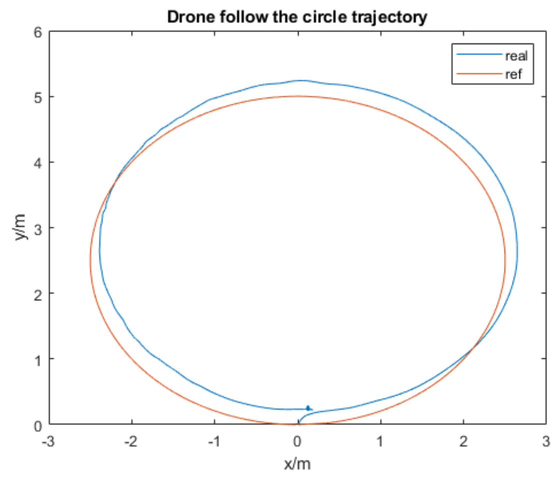


Figure 15: trajectory wind 1.15

According to figure 14 and 15, on the horizontal motion site, in both wind model, the drone can complete the circle with some extend of deviation.

CONTINUED

5.5 e

In this task, the goal is to control a follower drone to complete the trajectory as same as the first drone's meanwhile keeping a 0.3m gap between 2 drones. As the trajectory of first robot is rising from $[0; 0; 0]$ to $[0; 0; 2.5]$, then follow a circle with centre $[0; 2.5; 2.5]$ and radius $2.5m$. At last come back to the position $[0; 0; 2.5]$ then land to $[0, 0, 0]$. As the initial position of follower drone is $[0.3; 0; 0]$, the drone can follow the same track of the first but shift $0.3m$ along the positive x direction. Hence in every iteration, the target of follower drone should be the target of 1st drone with its x coordinate plus 3. The result is shown as follow.

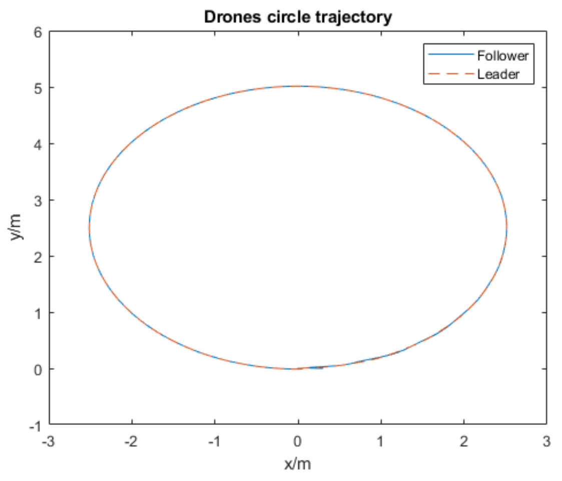


Figure 16: Drowns trajectory

Figure 16 shows the trajectory of 2 drones which can be observed that they are almost overlapped.

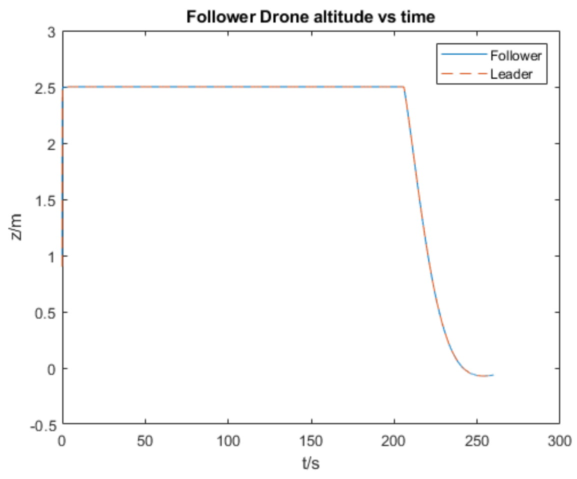


Figure 17: Drowns altitude

Figure 17 shows the altitude of 2 drones which are always in the same altitude.

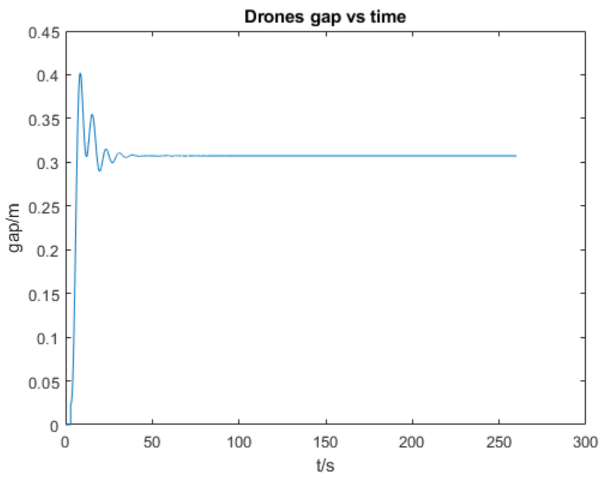


Figure 18: Drones gap

Figure 18 shows the gap between 2 drones along the time. Although there is an overshoot at the beginning, it can quickly adapt the change and stabilise at 0.3m which satisfy the requirement.

References

A. Gibiansky *Quadcopter Dynamics, Simulation, and Control*.

Dechao Meng *Quadcopter Simulator In Matlab*.

Andrew Symington *Simulating Quadrotor UAVs in Outdoor Scenarios*.

END OF COURSEWORK