ARS5 Project - Report - Multi-agents (3 agents) with a control law for stabilizing them in a desired position and tracking a trajectory

Runkun Luo & Poidatz Guillaume & Jichuan Zhang & Boyang Mu

January 16, 2024

1 Introduction

The goal of this project is to achieve cooperative control of multiple agents (three in this project) by employing control laws, ensuring their stability at desired positions and tracking trajectories.

Built upon the foundation of agents being quadcopter drones, the implementation of the project posed various challenges. These challenges included the complex dynamic modeling of quadcopter drones, design and verification of nonlinear control laws, and the design of cooperative control for multiple agents.

To address these issues, at the theoretical design level, we ultimately employed the Newton-Euler method for dynamic modeling of quadcopter drones. We utilized backstepping control laws to govern the dynamique system of the drones. Additionally, we introduced a cooperative control approach based on a multi-agent leader structure to realize the theoretical framework of this project.

At the numerical analysis level, we conducted numerical simulations using MATLAB (and a little part of Simulink). While ensuring that the system could accomplish single-drone as well as multi-drone with formation maintenance tracking trajectory, we also incorporated random noise into the six states of the drones and performed disturbance analysis. In the end, we successfully achieved the objectives of this project.

2 Dynamic system modeling

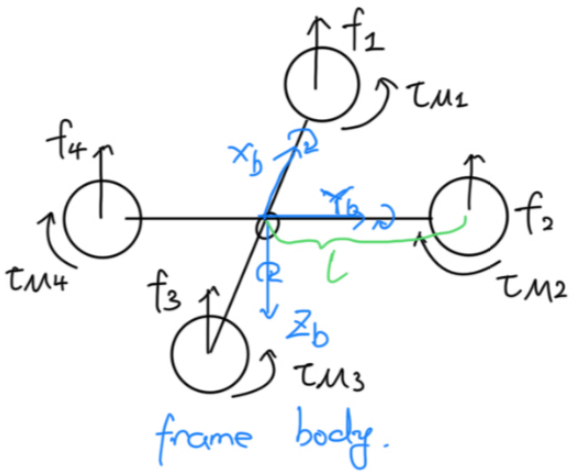

Figure 1: Dynamic Analysis of Quadrotor drone

First, we simplify the quadcopter drone into the structure shown in the figure 1. The informa-

tion represented by each variable in the figure is as follows:

X_e, Y_e, Z_e : Frame of Earth,
X_b, Y_b, Z_b : Body frame of drone,
$f_i\ (i = 1, 2, 3, 4)$: Pulling force provided by rotors.
$\tau_{Mi}\ (i = 1, 2, 3, 4)$: Torque provided by rotors.
l : Distance between center of the gravity and rotor.

In the following part, we will use b and e to present the variable is in which frame.

To simplify the dynamic analysis of a quadcopter drone, we only consider the gravitational force acting on the drone itself and the traction forces generated by its four propellers. By employing the Newton-Euler method, we obtain the following expressions:

$$F_{\text{all}} = m\ddot{X} \tag{1}$$
$$\tau_{\text{all}} = J\dot{\omega} + \omega \times J\omega \tag{2}$$

with

$$X = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \text{Position of drone.} \quad \omega_b = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad \text{Angular velocity of drone in body frame.}$$

F_{all} : Combined force
τ_{all} : Combined torque.

2.1 Detailed equation (1)

By considering the gravity of the frone is mg, we have :

$$m\ddot{v_e} = mg_e - f_b, \quad \ddot{v_e} = g_e - \frac{f_b}{m} \tag{3}$$

Since, in the subsequent steps, we aim to implement formation control among three drones, constructing the drone's dynamic model in the Earth coordinate system is more convenient for describing the state of each drone later on. For this purpose, we need a transformation matrix from the body frame to the Earth frame. Considering rotation matrices for object rotation around each axis and achieving the transformation between frames following the rotation sequence of XYZ axes, we obtain the rotation matrix as follows:

$$\begin{aligned} R_b^e &= R_z R_y R_x \\ &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta\cos\psi & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \cos\theta\sin\psi & \sin\psi\sin\theta\sin\phi - \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \end{aligned}$$

with

$$\theta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \quad \text{Euler angle}$$

By considering equation (3), in the Earth coordinate system, we have the detailed equations as follows:

$$\dot{v_e} = g_e - R_b^e \frac{f_b}{m}$$
$$\begin{bmatrix} \dot{V}_x^e \\ \dot{V}_y^e \\ \dot{V}_z^e \end{bmatrix} = g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \frac{f_b}{m} R_b^e \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

We got :

$$\dot{V}_x^e = -\frac{f_b}{m}(\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi) \tag{4}$$

$$\dot{V}_y^e = -\frac{f_b}{m}(\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi) \tag{5}$$

$$\dot{V}_z^e = g - \frac{f_b}{m}\cos\phi\cos\theta \tag{6}$$

2.2 Detailed equation (2)

The combined torque is calculated as follows:

$$\tau_{\text{all}} = G_a + \tau_{\text{rotors}}$$

τ_{rotors} : Torque from rotors

$$\tau_{\text{rotors}} = \begin{bmatrix}\tau_x\\ \tau_y\\ \tau_z\end{bmatrix} = \begin{bmatrix}(f_2 - f_4)l\\ (f_3 - f_1)l\\ \tau_{M2} + \tau_{M4} - \tau_{M1} - \tau_{M3}\end{bmatrix}$$

G_a : Spiral torque

$$G_a = \begin{bmatrix}G_{a\phi}\\ G_{a\theta}\\ G_{a\psi}\end{bmatrix} = \begin{bmatrix}J_r q(\omega_1 + \omega_2 - \omega_3 - \omega_4)\\ J_r p(-\omega_1 - \omega_2 + \omega_3 + \omega_4)\\ 0\end{bmatrix}$$

with

J_r : Inertia of the entire motor and rotor around the axis of the body

$\omega_i, i = 1, 2, 3, 4$: Angular velocity for each rotor.

In the following section we define:

$$\Omega = -\omega_1 - \omega_2 + \omega_3 + \omega_4$$

so

$$G_a = \begin{bmatrix}J_r q(-\Omega)\\ J_r p\Omega\\ 0\end{bmatrix}$$

Now before we continue the calculation, we need to find out the relation between Euler angular velocity and angular velocity of drone in the body frame.

If we want to express the model in the earth frame we need to change angular velocity of drone in the body frame to Euler angular velocity cause Euler angular is in the earth frame.

Normally :

$$\dot{\theta} = W\omega_b$$

with

W : A transfor mation matrix

In our project, we consider that the changes for all angles are small enough that we can consider the angular velocity of drone in the body frame is equal to Euler angular velocity. In this case:

$$J = I = \begin{bmatrix}J_x & 0 & 0\\ 0 & J_y & 0\\ 0 & 0 & J_z\end{bmatrix}$$

$$\tau_{\text{all}} = J\dot{\omega} + \omega \times J\omega$$

$$J\dot{\omega} = \tau_{\text{all}} - \omega \times J\omega$$

$$\begin{bmatrix}J_x\ddot{\phi}\\ J_y\ddot{\theta}\\ J_z\ddot{\psi}\end{bmatrix} = \begin{bmatrix}\tau_x\\ \tau_y\\ \tau_z\end{bmatrix} + \begin{bmatrix}J_r\dot{\theta}(-\Omega)\\ J_r\dot{\phi}\Omega\\ 0\end{bmatrix} + \begin{bmatrix}\dot{\theta}\dot{\psi}(J_y - J_z)\\ \dot{\phi}\dot{\psi}(J_z - J_x)\\ \dot{\phi}\dot{\theta}(J_x - J_y)\end{bmatrix}$$

We got:

$$\ddot{\phi} = \frac{\dot{\theta}\dot{\psi}(J_y - J_z)}{J_x} - \frac{J_r\dot{\theta}\Omega}{J_x} + \frac{\tau_x}{J_x} \tag{7}$$

$$\ddot{\theta} = \frac{\dot{\phi}\dot{\psi}(J_z - J_x)}{J_y} + \frac{J_r\dot{\phi}\Omega}{J_y} + \frac{\tau_y}{J_y} \tag{8}$$

$$\ddot{\psi} = \frac{\dot{\phi}\dot{\theta}(J_x - J_y)}{J_z} + \frac{\tau_z}{J_z} \tag{9}$$

In the end, By combining equations (4)(5)(6)(7)(8)(9) we have derived the dynamic model of the drone in the Earth coordinate system :

$$\begin{aligned}
\ddot{x} &= -\frac{f_b}{m}(\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi) \\
\ddot{y} &= -\frac{f_b}{m}(\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi) \\
\ddot{z} &= g - \frac{f_b}{m}\cos\phi\cos\theta \\
\ddot{\phi} &= \frac{\dot{\theta}\dot{\psi}(J_y - J_z)}{J_x} - \frac{J_r\dot{\theta}\Omega}{J_x} + \frac{\tau_x}{J_x} \\
\ddot{\theta} &= \frac{\dot{\phi}\dot{\psi}(J_z - J_x)}{J_y} + \frac{J_r\dot{\phi}\Omega}{J_y} + \frac{\tau_y}{J_y} \\
\ddot{\psi} &= \frac{\dot{\phi}\dot{\theta}(J_x - J_y)}{J_z} + \frac{\tau_z}{J_z}
\end{aligned}$$

3 Movement control

In this part we will try to propose a controller using the approach Backstepping in order to let the system converge to a desired position or finish the trajectory tracking.

According to the dynamic model we have got in the art before, we first define the inputs U and outputs Y as following :

$$U = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} f \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}$$

$$Y = \begin{bmatrix} \phi & \theta & \psi & z & x & y \end{bmatrix}^T$$

Then in order to use the Backstepping approach, we need to rewrite the system model into the form of state spaces.

By defining the states as following :

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix} = \begin{bmatrix} \phi \\ \dot{\phi} \\ \theta \\ \dot{\theta} \\ \psi \\ \dot{\psi} \\ z \\ \dot{z} \\ x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}$$

We rewrite the model of the system as below :

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \\ \dot{x_6} \\ \dot{x_7} \\ \dot{x_8} \\ \dot{x_9} \\ \dot{x_{10}} \\ \dot{x_{11}} \\ \dot{x_{12}} \end{bmatrix} = \begin{bmatrix} x_2 \\ x_4x_6a_1 - x_4b_1 + c_1U_2 \\ x_4 \\ x_2x_6a_2 - x_2b_2 + c_2U_3 \\ x_6 \\ x_2x_4a_3 + c_3U_4 \\ x_8 \\ g - U_1 cos x_1 cos x_3/m \\ x_{10} \\ -U_1U_x/m \\ x_{12} \\ -U_1U_y/m \end{bmatrix}$$

with

$$\begin{aligned} a_1 &= J_y - J_z/J_x; \quad b_1 = J_r\Omega/J_x; \quad c_1 = 1/J_x \\ a_2 &= J_z - J_x/J_y; \quad b_2 = J_r\Omega/J_y; \quad c_2 = 1/J_y \\ a_3 &= J_x - J_y/J_z; \quad c_3 = 1/J_z \\ U_x &= \cos x_5 \sin x_3 \cos x_1 + \sin x_1 \sin x_5 \\ U_y &= \sin x_5 \sin x_3 \cos x_1 + \sin x_1 \cos x_5 \end{aligned}$$

It should be noted that Ux and Uy in the above variables will be used as two additional inputs to the system . The specific reasons will be explained later.

Next we begin to using the Backstepping approach. In order to make the analysis process clearer and avoid too many elements appearing at the same time, we divide the system into two small subsystems based on the characteristics of states x1-x6 controlling the system attitude (S1) and states x7-x12 controlling the system position (S2) in the overall state of the system, so as to carry out controller design and verification process.

3.1 Controller design and verification for the S2

In order to make each state quantity in the system converge to the target value, we need to first replace the state with the error between the state and the target value. In this way, when we implement the design of the controller, the error will converge to 0, which means The state of the system converges to the target value. By defining the errors :

$$\begin{aligned} e_z &= x_7 - x_{7d} \\ e_x &= x_9 - x_{9d} \\ e_y &= x_{11} - x_{11d} \end{aligned}$$

The state space expression is rewritten as follows:

$$\dot{e_z} = x_8 - \dot{x_{7d}} \tag{10}$$
$$\ddot{e_z} = g - U_1 cos x_1 cos x_3/m - \ddot{x_{7d}} \tag{11}$$
$$\dot{e_x} = x_{10} - \dot{x_{9d}} \tag{12}$$
$$\ddot{e_x} = -U_1U_x/m - \ddot{x_{9d}} \tag{13}$$
$$\dot{e_y} = x_{12} - \dot{x_{11d}} \tag{14}$$
$$\ddot{e_y} = -U_1U_y/m - \ddot{x_{11d}} \tag{15}$$

We begin to design the controller and at the same time analyze whether this controller can achieve stable results.

Through the first equation (10) we have obtained so far, we can design a quadratic equation containing this state according to the Lyapunov analysis method as follows :

$$V_1 = 1/2e_z^2$$

Then we design the controller so that the quadratic equation meets the three basic requirements of the Lyapunov function, so that we can get a controller that can keep the system stable as it approaches the target value. The design analysis is as follows :

Because now the quadratic equation V1 is already match with V1(0) = 0, V1 >= 0 two requirements of the Lyapunov function, so we just need to verify the third condition which is the derivative of V1 <= 0.

$$\begin{aligned}\dot{V_1} &= e_z \dot{e_z} \\ &= e_z(x_8 - \dot{x_{7d}})\end{aligned}$$

By proposing

$$x_8^p = \dot{x_{7d}} - k_1 e_z$$

With

$$\begin{aligned}&k_1 > 0 \\ &\dot{V_1} = -k_1 e_z^2 \\ &\quad <= 0\end{aligned}$$

We have successfully designed the V1 a Lyapunov function, but untill now we havn't built the relation between the controller and the input ,so we still need to verify the error between the proposed value of state x8 and the real value of state x8 to try to link the controller and the input.

$$\begin{aligned}e_2 &= x_8 - x_8^p \\ &= x_8 - \dot{x_{7d}} + k_1 e_z\end{aligned}$$

$$\dot{e_2} = \ddot{e_z} + k_1 \dot{e_z} \tag{16}$$

Now we can replace the equation (11) by the new state we obtained equation (16),and we redesign a quadratic equation V2 and do the same process as before.

$$\begin{aligned}&V_2 = 1/2 e_z^2 + 1/2 e_2^2 \\ &\dot{V_2} = -k_1 e_z^2 + e_2(g - U_1 cos x_1 cos x_3/m - \ddot{x_{7d}} + k_1 \dot{e_z})\end{aligned}$$

By propsing

$$U_1 = m(g - \ddot{x_{7d}} + k_1 \dot{e_z} + k_2 e_2)/cos x_1 cos x_3 \tag{17}$$

With

$$\begin{aligned}&k_2 > 0 \\ &\dot{V_2} = -k_1 e_z^2 - k_2 e_2^2 \\ &\quad <= 0\end{aligned}$$

So in this way, by design the controller like above, we can let V1 and V2 both match the requirements of Lyapunov function, which can also prove that the system is stable under this controller.

Because the remaining five states are highly similar to this state when designing a controller using the back-stepping control method, for reasons of space, we only expand on the above design analysis process in detail in this article. The control design of rest two states in this system is as follows :

$$U_x = m(-\ddot{x_{9d}} + k_3 \dot{e_x} + k_4 e_3)/U_1 \tag{18}$$

With

$$\begin{aligned}\dot{e_x} &= x_{10} - \dot{x_{9d}} \\ e_3 &= x_{10} - \dot{x_{9d}} + k_3(x_9 - x_{9d}) \\ k_3, k_4 &> 0\end{aligned}$$

$$U_y = m(-\ddot{x_{1d}} + k_5\dot{e_y} + k_6 e_4)/U_1 \tag{19}$$

With

$$\begin{aligned} \dot{e_y} &= x_{12} - \dot{x_{1d}} \\ e_4 &= x_{12} - \dot{x_{1d}} + k_5(x_{11} - x_{11d}) \\ k_5, k_6 &> 0 \end{aligned}$$

The reason why we use Ux and Uy as system inputs to design the controller here is that we can see from the initial system state equation that the U1 input will be combined with the three states in the attitude subsystem to affect the three positions of the system. status has an impact.

Therefore, when we use a relatively simple z state variable to design the controller of U1, we cannot guarantee that the controlled U1 can also make the position states of the system in x and y equally convergent and stable. Therefore, we redesign the two inputs Ux and Uy additional control. In this way, through the joint control of UxU1 and UyU1, we can also achieve the effect of x and y state convergence.

Through the expressions after Ux and Uy are controlled, we can also find their relationship with the joint control state of U1. Both of them are divided by U1.

And because Ux and Uy are composed of three quantities in the attitude system, these two input quantities also have the following relationship with the attitude state.

$$\begin{aligned} U_x &= cosx_5 sinx_3 cosx_1 + sinx_1 sinx_5 \\ U_y &= sinx_5 sinx_3 cosx_1 + sinx_1 cosx_5 \\ \phi &= arcsin(U_x sinx_5 - U_y cosx_5) \\ \theta &= arcsin(U_x/(cosx_1 cosx_5) - (sinx_1 sinx_5)/(cosx_1 cosx_5)) \end{aligned}$$

3.2 Controller design and verification for the S1

As mentioned above, the six state quantities have extremely high similarities when designing controllers using the back-stepping control method. Therefore, the results will be shown in the form of formulas below. Errors define :

$$\begin{aligned} e_\phi &= x_1 - x_{1d} \\ e_\theta &= x_3 - x_{3d} \\ e_\psi &= x_5 - x_{5d} \end{aligned}$$

The state space expression is rewritten as follows:

$$\dot{e_\phi} = x_2 - \dot{x_{1d}} \tag{20}$$
$$\ddot{e_\phi} = x_4 x_6 a_1 - x_4 b_1 + c_1 U_2 - \ddot{x_{1d}} \tag{21}$$
$$\dot{e_\theta} = x_4 - \dot{x_{3d}} \tag{22}$$
$$\ddot{e_\theta} = x_2 x_6 a_2 - x_2 b_2 + c_2 U_3 - \ddot{x_{3d}} \tag{23}$$
$$\dot{e_\psi} = x_6 - \dot{x_{5d}} \tag{24}$$
$$\ddot{e_\psi} = x_2 x_4 a_3 + c_3 U_4 - \ddot{x_{5d}} \tag{25}$$

By using the control law, the controller are as below :

$$U_2 = (-x_4 x_6 a_1 + x_4 b_1 + \ddot{x_{1d}} - k_7\dot{e_\phi} - k_8 e_5)/c_1 \tag{26}$$

With

$$\begin{aligned} \dot{e_\phi} &= x_2 - \dot{x_{1d}} \\ e_5 &= x_2 - \dot{x_{1d}} + k_7(x_1 - x_{1d}) \\ k_7, k_8 &> 0 \end{aligned}$$

$$U_3 = (-x_2 x_6 a_2 + x_2 b_2 + \ddot{x_{3d}} - k_9\dot{e_\theta} - k_{10} e_6)/c_2 \tag{27}$$

With

$$\begin{aligned} \dot{e_\theta} &= x_4 - \dot{x_{3d}} \\ e_6 &= x_4 - \dot{x_{3d}} + k_9(x_3 - x_{3d}) \\ k_9, k_{10} &> 0 \end{aligned}$$

$$U_4 = (-x_2 x_4 a_3 + \ddot{x_{5d}} - k_{11}\dot{e_\psi} - k_{12}e_7)/c_3 \tag{28}$$

With

$$\begin{aligned} \dot{e_\psi} &= x_6 - \dot{x_{5d}} \\ e_7 &= x_6 - \dot{x_{5d}} + k_{11}(x_5 - x_{5d}) \\ k_{11}, k_{12} &> 0 \end{aligned}$$

The equations (17)(18)(19)(26)(27)(28) are the designed controllers with the relation mention between the Ux Uy and the three altitude states of the system.

4 Matlab implementation of single-agent

In the previous sections we established the dynamics model of a single drone, which can be computed to obtain the position as well as the pose of the drone at each instant. Meanwhile, considering that the control of each agent in multi-agent control is almost the same, we can start our study with a single drone.

In our case, we assume that the acceleration of gravity is g = 9.81, the mass of the drone is m = 0.6150 kg, with :

$$\begin{aligned} J_x &= 0.0154 \\ J_y &= 0.0154 \\ J_z &= 0.0309 \\ J_r &= 0.05 \end{aligned}$$

In the following parts, we will study three scenarios : travelling to the target position without noise, travelling to the target position with noise, and travelling to the target position along a given trajectory.

4.1 Target position without noise

The code for this section can be found in the file ***control_for_one_noise .m***.

In this part, since the drone is moving to a fixed position, we set the velocity and acceleration of the drone to 0 at the initial position and the target position.

$$\begin{aligned} x_I = \dot{x_I} = \ddot{x_I} &= 0 \\ y_I = \dot{y_I} = \ddot{y_I} &= 0 \\ z_I = \dot{z_I} = \ddot{z_I} &= 0 \\ \dot{x_d} = \ddot{x_d} &= 0 \\ \dot{y_d} = \ddot{y_d} &= 0 \\ \dot{z_d} = \ddot{z_d} &= 0 \end{aligned}$$

In addition, we assume that the drone will be stable at the initial position and at the target

position, so the velocity and acceleration will be 0 for all three angles.

$$\phi_I = \dot{\phi}_I = \ddot{\phi}_I = 0$$
$$\theta_I = \dot{\theta}_I = \ddot{\theta}_I = 0$$
$$\psi_I = \dot{\psi}_I = \ddot{\psi}_I = 0$$
$$\dot{\phi}_d = \ddot{\phi}_d = 0$$
$$\dot{\theta}_d = \ddot{\theta}_d = 0$$
$$\dot{\psi}_d = \ddot{\psi}_d = 0$$

Once we have all this information, we can update the state values of the drone through a for loop. At the beginning of the loop, the current state values are substituted into the equations that we got in the part of movement control, we can obtain the second-order derivatives of each state value of the system. Finally, by making the Euler integration twice, we can obtain the state values of the drone, then they could be used in the next loop.

By changing the state values at the target position, we can obtain the following simulation results:

(1) The target position is relatively close to the initial position:

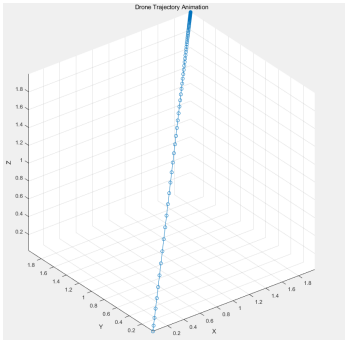

Figure 2: Drone trajectory (close, without noise)

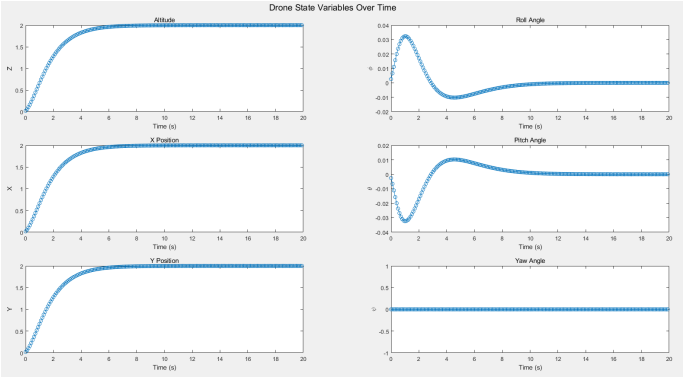

Figure 3: Drone state values over time (close, without noise)

(2) The target position is relatively far from the initial position:

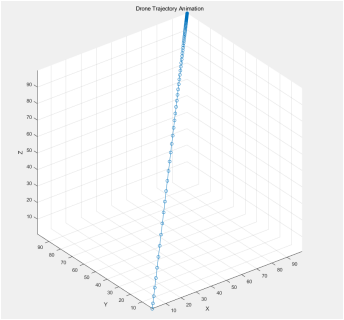

Figure 4: Drone trajectory (far, without noise)

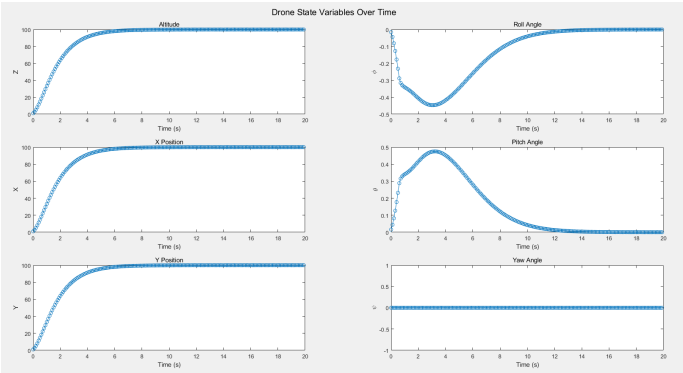

Figure 5: Drone state values over time (far, without noise)

By observing (1) and (2), we can find that: when there is no noise, the drone will reach the target position and pose along a straight line and the process is stable. The distance of the target position in this case does not affect the stability of the trajectory.

4.2 Noise Addition

In order to add noise to the system, we need to add a relatively small random number to each state value at the beginning of each loop, here we use the randn() function.

By changing the state values at the target position, we can obtain the following simulation results:

(1) The target position is relatively close to the initial position:

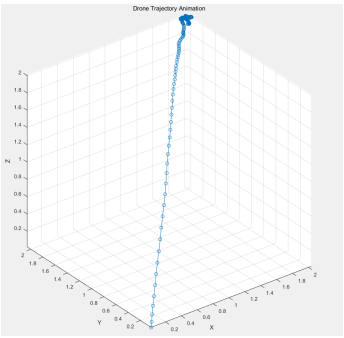

Figure 6: Drone trajectory (close, with noise)

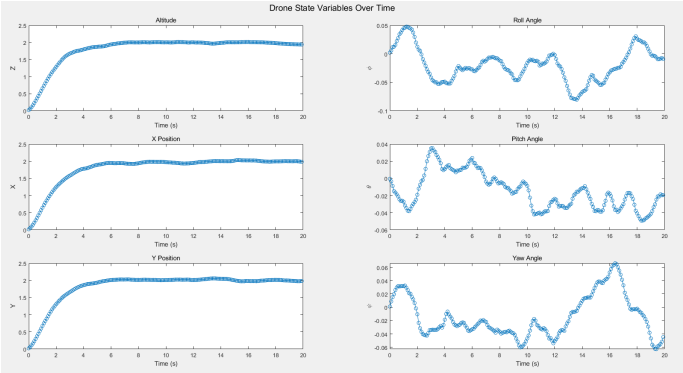

Figure 7: Drone state values over time (close, with noise)

(2) The target position is relatively far from the initial position:

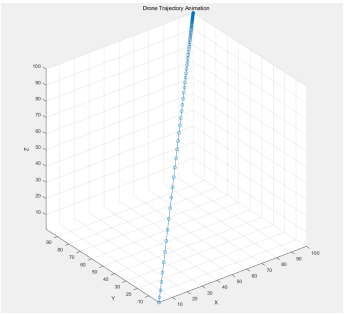

Figure 8: Drone trajectory (far, with noise)

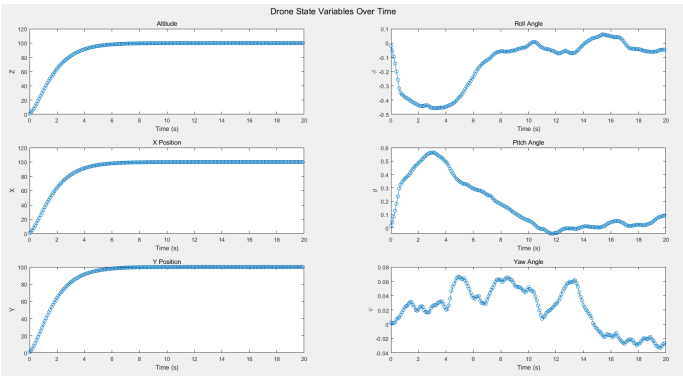

Figure 9: Drone state values over time (far, with noise)

We can observe that the distance to the target position still does not affect the convergence of the trajectories.

In addition, the added noise leads to oscillations in the process, and the oscillations in the trajectory are more evident when the target position is closer, because the amplitude of the drone displacement is smaller.

It is worth mentioning that in all the simulations we set the roll and pitch angles of the drone at the target position to 2, but eventually both angles go to 0. This is because in reality, if the drone is stopped at a fixed position, it is impossible for its roll and pitch not to be 0.

4.3 Trajectory tracking

The code for this section can be found in the file ***one_ noise_ ellipse .m***.

+In this part, we will not consider the case without noise in order to be closer to the real situation.

In order to enable the drone to follow a given trajectory, we need to provide the trajectory equation and use it to update the target position at the beginning of each loop. That means that the target position is no longer constant.

Here we have chosen to have the drone fly along an ellipse at a fixed altitude, so that the target position is updated according to the equation of the ellipse at the beginning of each loop.

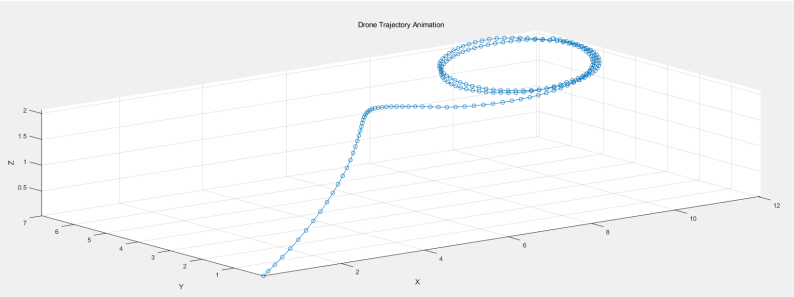

Figure 10: Drone trajectory (ellipse)

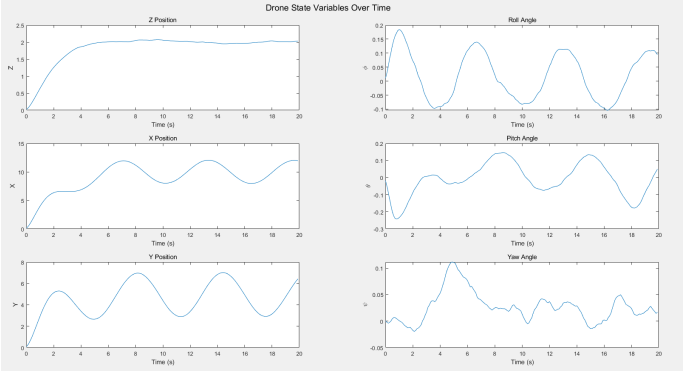

Figure 11: Drone state values over time (ellipse)

The drone will rise to a given altitude and then fly along an elliptical trajectory. Although there will be some small oscillations, every state value will be close to the target value, as we expect, except for the roll and pitch angles, which will return to 0.

5 Multi-agent coordination control

The three dimensions of the drone are independent of each other, for example, we think that the displacement, velocity and acceleration that change in the X direction will not affect the displacement, velocity and acceleration in the other two directions. Therefore, in order to simplify the writing, in the following process, we will only take the X-axis direction as an example, Y and Z are actually exactly the same as the X-axis.

5.1 Communication model

Considering the above graph of communication, it is clearly to note that our communication structure is a **Chain topology with input and output on the first agent**, which means drone 1 can communicate with drone 2, drone 2 can communicate with drone 3, all the communications are bidirectional.

5.2 Definition of the system

It is a three dimensions problem, therefore the theoretical and also the real dimension of the state vector should be 9×1. But as we indicated before, taking the dimension X as the example, so the states vector using in the following part in which we will detail the process of convergence

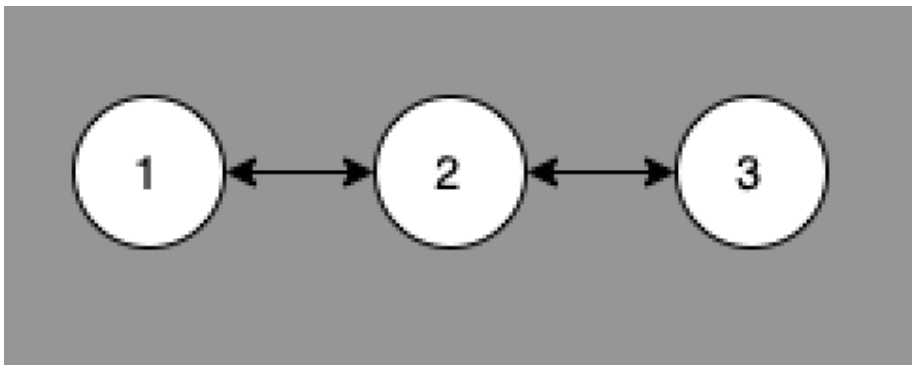

Figure 12: Graph of communication

proving is:

$$X = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T \tag{29}$$

and the derivation of the state vector is:

$$\dot{X} = \begin{bmatrix} \dot{x_1} & \dot{x_2} & \dot{x_3} \end{bmatrix}^T \tag{30}$$

Let we define the following three systems:

$$\dot{x_1} = u_1 \tag{31}$$
$$\dot{x_2} = u_2 \tag{32}$$
$$\dot{x_3} = u_3 \tag{33}$$

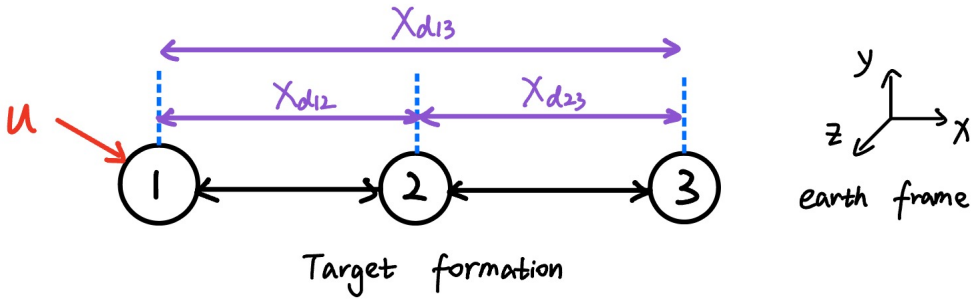

Figure 13: Target formation in world frames

Considering for the simplification of the simulation, the drones are considered as a point mass. All variables with a subscript of d indicate the desired target value to be achieved. For example: x_{d13} indicates the target distance between drone 1 and drone 3 to be maintained in the x direction, and x_{d23} indicates the target distance between drone 2 and drone 3 to be maintained in the x direction, and the same for x_{d12}.

$$x_{d12} = x_1 - x_2 = -x_d \neq 0 \tag{34}$$
$$x_{d23} = x_2 - x_3 = -x_d \neq 0 \tag{35}$$
$$x_{d12} = x_1 - x_2 = -2x_d \neq 0 \tag{36}$$

Since the target distance between the drones needs to be taken into account during the design of the controller u_i, we need to replace the variables x_1, x_2 and x_3 by p_{x1}, p_{x2} and p_{x3}:

$$p_{x1} = x_1 \tag{37}$$
$$p_{x2} = x_2 + x_{d12} \tag{38}$$
$$p_{x3} = x_3 + x_{d13} \tag{39}$$

Then, rewriting the three systems:

$$X = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \tag{40}$$

$$\dot{X} = \begin{bmatrix} \dot{p_{x1}} \\ \dot{p_{x2}} \\ \dot{p_{x3}} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{41}$$

Let we define the controllers for three drones:

$$u_1 = -p_{x1} + p_{x2} + u \tag{42}$$
$$u_2 = (-p_{x2} + p_{x1}) + (-p_{x2} + p_{x3}) = p_{x1} - 2p_{x2} + p_{x3} \tag{43}$$
$$u_3 = -p_{x3} + p_{x2} \tag{44}$$

According to the standard form: $\dot{X} = Ax + Bu$ and $y = Cx$, we will have:

$$\dot{X} = \begin{bmatrix} \dot{p_{x1}} \\ \dot{p_{x2}} \\ \dot{p_{x3}} \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} p_{x1} \\ p_{x2} \\ p_{x3} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \times u \tag{45}$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} p_{x1} \\ p_{x2} \\ p_{x3} \end{bmatrix} \tag{46}$$

In this case, the Laplacian matrix is then:

$$-\mathcal{L} = \begin{bmatrix} -1 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -1 \end{bmatrix} \tag{47}$$

For all three three dimensions X, Y and Z, we have the same Laplacian matrix.

5.3 Proof of convergence of states

In this part, we want to proof that with three controllers, u1, u2, and u3, the three drones can reach the target position maintaining the formation.

Step 1: Calculate eigenvalues and eigenvectors of the Laplacian matrix $\mathcal{L}$:

$$det\,(\lambda I - \mathcal{L}) = 0 \tag{48}$$
$$\begin{vmatrix} \lambda + 1 & -1 & 0 \\ -1 & \lambda + 2 & -1 \\ 0 & -1 & \lambda + 1 \end{vmatrix} = 0 \tag{49}$$
$$(\lambda + 1) \times [(\lambda + 2) \times (\lambda + 1) - 1] + (-1) \times [(-1) \times (\lambda + 1)] = 0 \tag{50}$$

We will have:

$$\Rightarrow \quad \lambda_1 = 0, \quad \lambda_2 = -1, \quad \lambda_3 = -3 \tag{51}$$

According to the theory: $W_i^T \times \mathcal{L} = \lambda_i \times W_i^T$

When $\lambda_1 = 0$, we have:

$$W_1^T = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \tag{52}$$

When $\lambda_2 = -1$, we have:

$$W_2^T = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \tag{53}$$

When $\lambda_3 = -3$, we have:

$$W_3^T = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \tag{54}$$

Step 2: Rewriting the equations of the systems: multiplying the systems by W_i^T:

$$W_i^T \times \dot{X} = W_i^T \times -\mathcal{L} \times X + W_i^T \times B \times u \tag{55}$$

Hence, the system becomes:

$$\dot{p_{x1}} + \dot{p_{x2}} + \dot{p_{x3}} = u \tag{56}$$
$$-\left(\dot{p_{x1}} - \dot{p_{x3}}\right) = (p_{x1} - p_{x3}) + u \tag{57}$$
$$\dot{p_{x1}} - 2\dot{p_{x2}} + \dot{p_{x3}} = -3\left(p_{x1} - 2p_{x2} + p_{x3}\right) + u \tag{58}$$

Define:

$$\dot{x_{cm}} = \frac{1}{3}\sum_{i=1}^{3}\dot{x_i} = \frac{1}{3}\left(\dot{p_{x1}} + \dot{p_{x2}} + \dot{p_{x3}}\right) = \frac{1}{3}u \tag{59}$$

Propose:

$$u = -3k\sigma\left(e_{cm}\right); \tag{60}$$
$$with \qquad e_{cm} = x_{cm} - x_{cm}^d \tag{61}$$

u : input of the system, also plays a role as the controller for drone 1
x_{cm} : the actual position of the center of mass of three drones
x_{cm}^d : the destination of the center of mass of three drones
e_{cm} : distance between the actual position of center mass and its target position
k : gain which controls the speed of convergence
σ : saturation gain seen as a velocity constraint

Therefore, when $t \to \infty$, we will have:

$$\dot{x_{cm}} = -k\sigma\left(x_{cm} - x_{cm}^d\right) \tag{62}$$
$$\Rightarrow x_{cm} \to x_{cm}^d \tag{63}$$
$$\Rightarrow u = 0 \tag{64}$$

$$p_{x1} - p_{x3} \to 0 \Rightarrow x_1 - x_3 \to x_{d13} \tag{65}$$
$$p_{x1} - p_{x2} \to 0 \Rightarrow x_1 - x_2 \to x_{d12} \tag{66}$$
$$\Rightarrow x_2 - x_3 \to x_{d23} \tag{67}$$

That means, with the controllers:

$$u_1 = -x_1 + x_2 + x_{d12} + u \tag{68}$$
$$u_2 = (-x_2 + x_1 - x_{d12}) + (-x_2 + x_3 + x_{d23}) \tag{69}$$
$$u_3 = -x_3 + x_2 - x_{d23} \tag{70}$$

Based on the above equations, we can find that x_2 will follow x_1 but keep a distance of x_{d12} from x_1, x_3 will follow x_2 but keep a distance of x_{d23} from x_2, and the centre of mass of the 3 drones will follow its pre-determined trajectory. Therefore we can get these three drones attained their target destinations, and kept the designed formation during the moving process.

6 Algorithm and structure of the .m code

Our code consists of two main parts, the first part is the main program, ***control_ for_ one _ noise _ v2.m***, used to calculate all states of three drones, another is a subfuncion part, ***multi_ agent _ positions_ calculus_ v2.m***, used to calculate target trajectories for 3 drones. In the main program:

Step 1: We first define some intrinsic parameters of the drones, such as rotational inertia in each direction, mass, etc. The initial position, initial velocity, initial relative position, and final relative distance of the drones should also be defined in this step.

Step 2: The target trajectory of the centre of mass is defined (Two types: 1 fixed point and an ascending spiral).

Step 3: The target trajectory of the centre of mass that has been defined in Step2 and some of the parameters defined in Step1 are used as inputs to our subfunction. In this subfunction, we first define the three controllers u1, u2 and u3 based on matrix Laplacian matrix $\mathcal{L}$ which has been detailed before in the Multi-agent coordination control part, then calculate the target trajectories of the 3 drones based on $\dot{x^i} = u^i$ and $x^i_{k+1} = x^i_k + dt \times u^i$, then return these trajectories to the main program.

Step 4: Establishing the dynamics model of the drones, through the **Backstepping control method**. Taking the target positions of the drones at each moment as the inputs, then calculating the real position, speed, acceleration, angle, angular velocity and angular acceleration of each drone (from the target trajectory to the real trajectory).

Step5: Draw the images

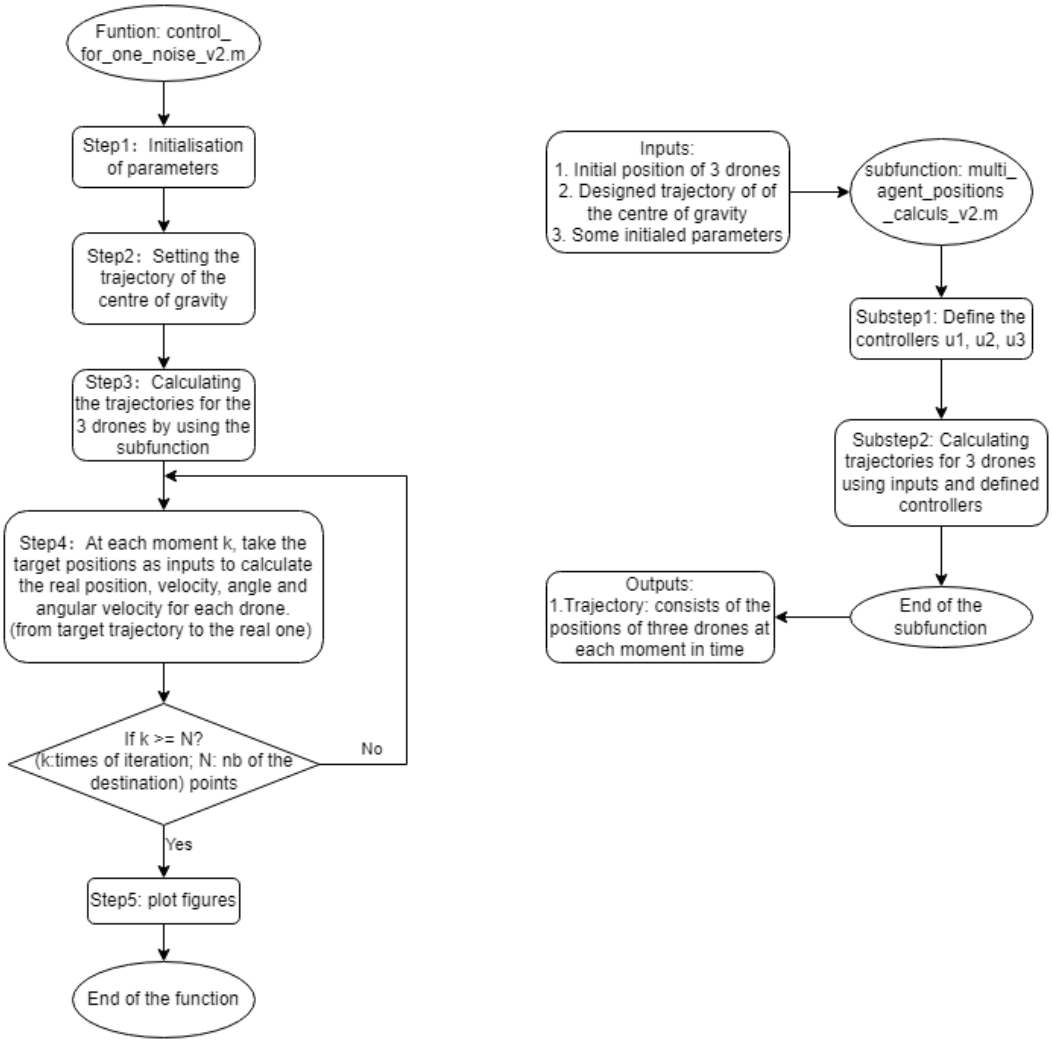

Figure 14: Code Logic Diagram

7 Numerical simulations

As shown in Figure 15 below, the trajectory we chose is an ascending spiral with a radius of 200 on the long axis and 100 on the short axis, and the starting point of the spiral is the initial position of the centre of mass. The position of the centre of mass is the average of the three drone positions.

The initial positions of the three drones are:

$$\begin{aligned} x_1 &= 0; & y_1 &= 0; & z_1 &= 0; \\ x_2 &= -10; & y_2 &= 0; & z_2 &= 0; \\ x_3 &= -30; & y_3 &= 0; & z_3 &= 0; \end{aligned}$$

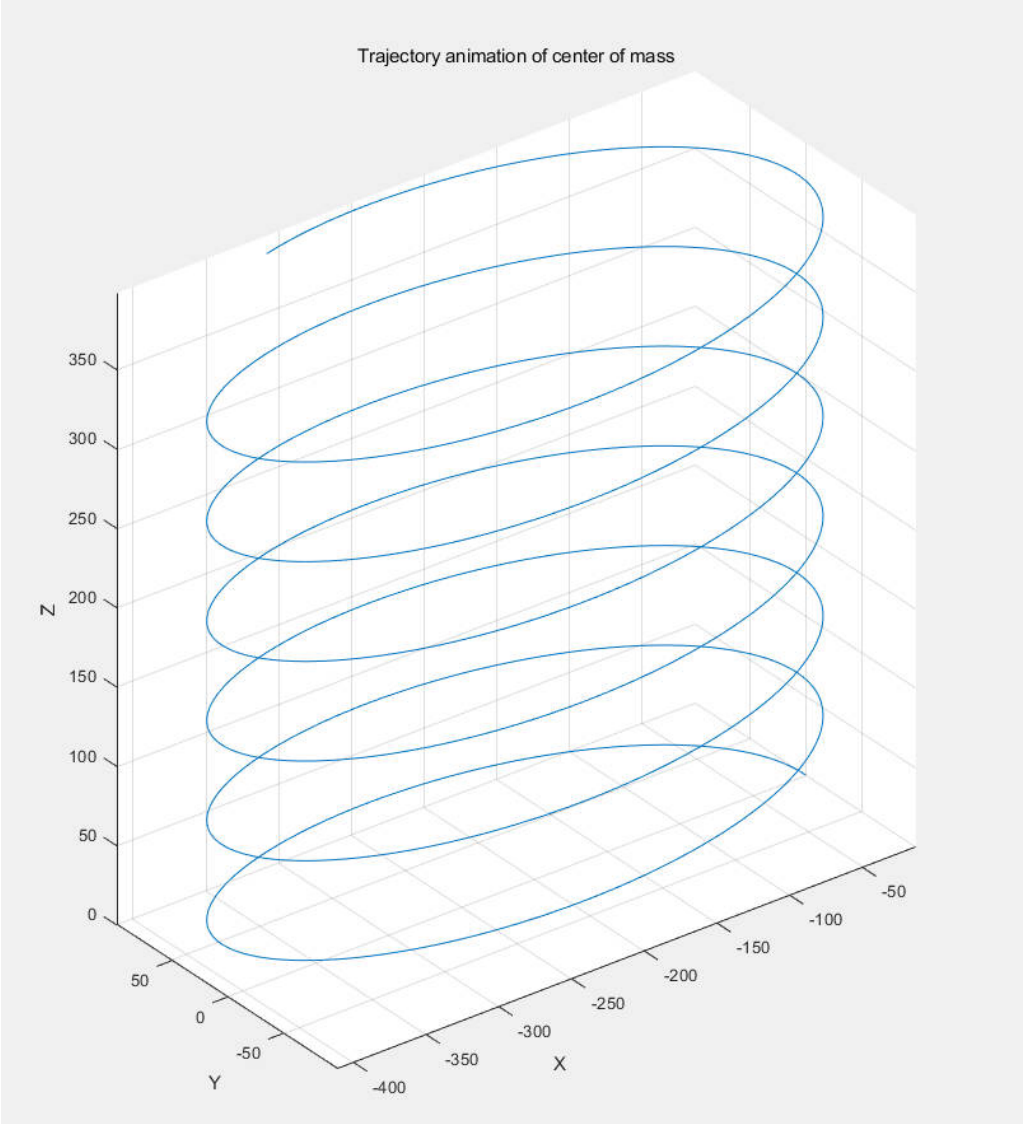

Figure 15: Target trajectory for center of mass

We define that the target distance between drone 1 and drone 2 is:

$$\begin{aligned} x_{d12} &= x_{d1} - x_{d2} = -20 \\ y_{d12} &= y_{d1} - y_{d2} = 20 \\ z_{d12} &= z_{d1} - z_{d2} = 0 \end{aligned}$$

and the target distance between drone 2 and drone 3 is:

$$x_{d23} = x_{d3} - x_{d3} = 40$$
$$y_{d23} = y_{d2} - y_{d3} = 0$$
$$z_{d23} = z_{d2} - z_{d3} = 0$$

Based on the three preliminarily designed controllers, u1, u2, and u3, and the target trajectories of the centre of mass, we obtain the target trajectories of the three drones under multi-agent control.

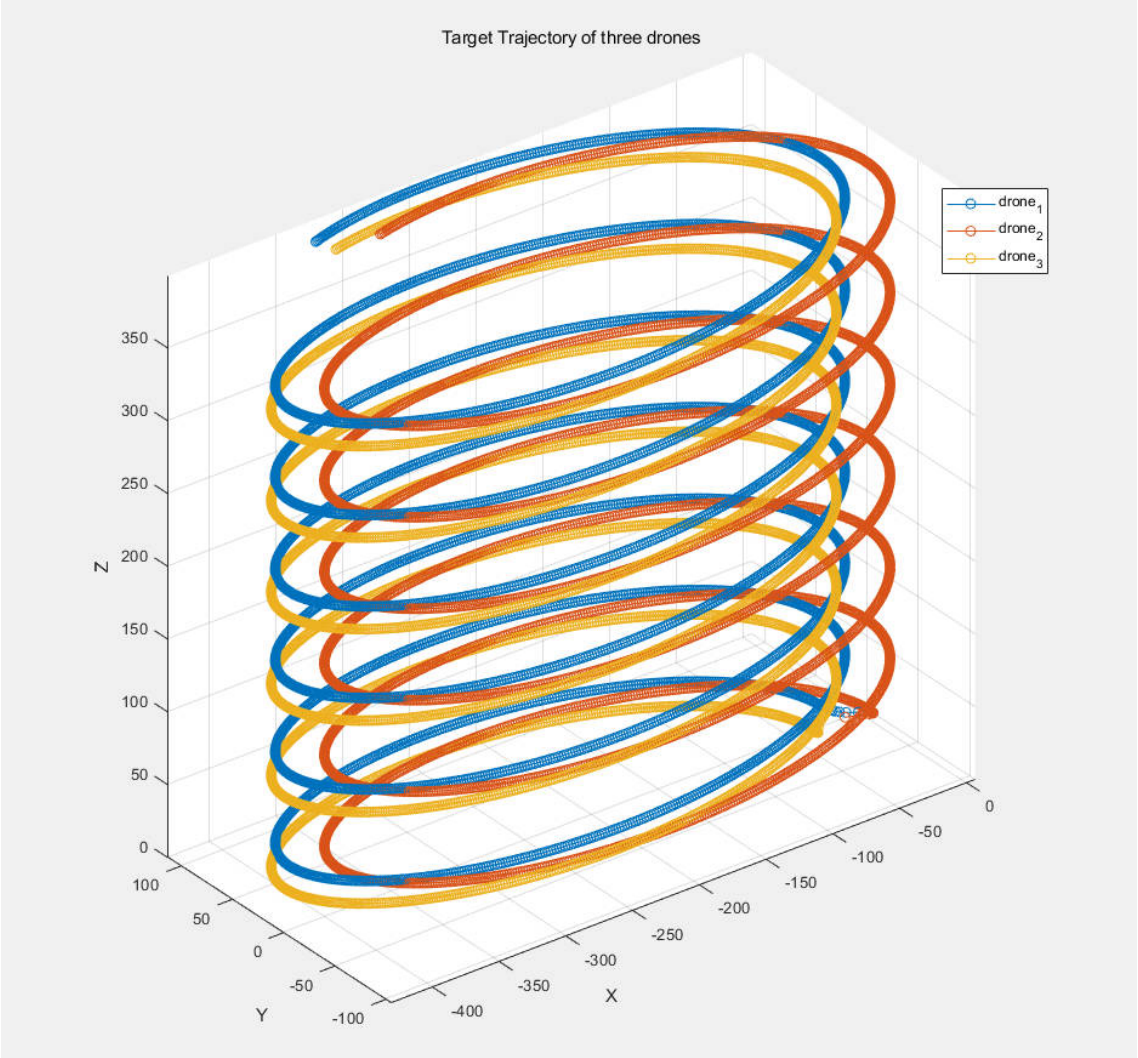

Figure 16: Target trajectories for three drones

Taking the obtained target trajectories of the drones as input, based on the dynamics model of the drone and applying Backstepping control, we can get the actual trajectory of the drones as shown in the figure below.

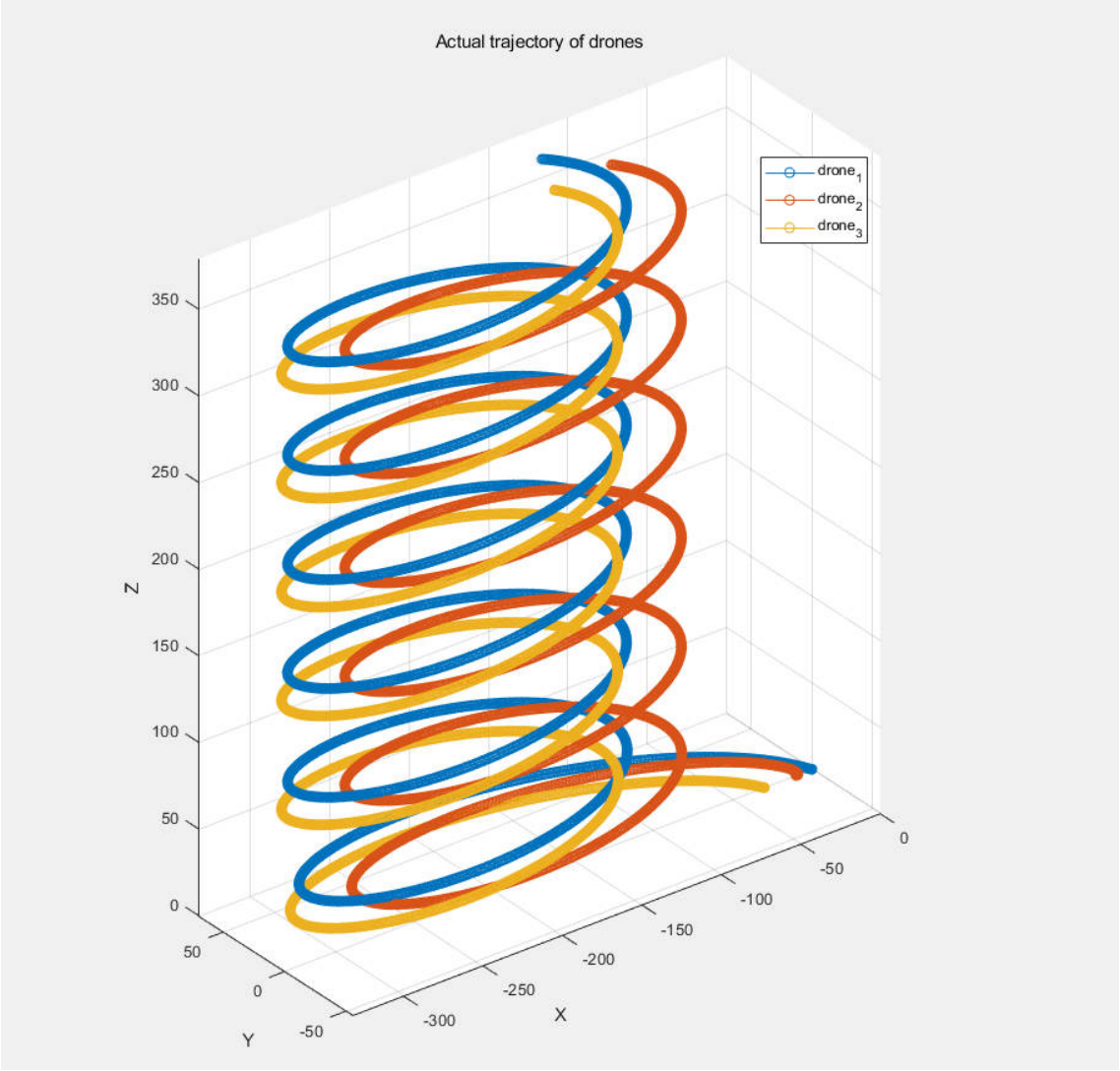

Figure 17: Actual trajectories for three drones

According to Figure 17 and Figure 16, it can be seen that our three drones are roughly following the spiralling upwards like the target trajectory. However, if we look closely, we can find that the actual trajectories of the drones are not exactly the same as the target trajectories. The specific differences between the them and the reasons will be analysed in detail later in this section and in the next section respectively.

At the same time, we can also plot the velocity evolution, angular evolution, and angular acceleration evolution of the drones.

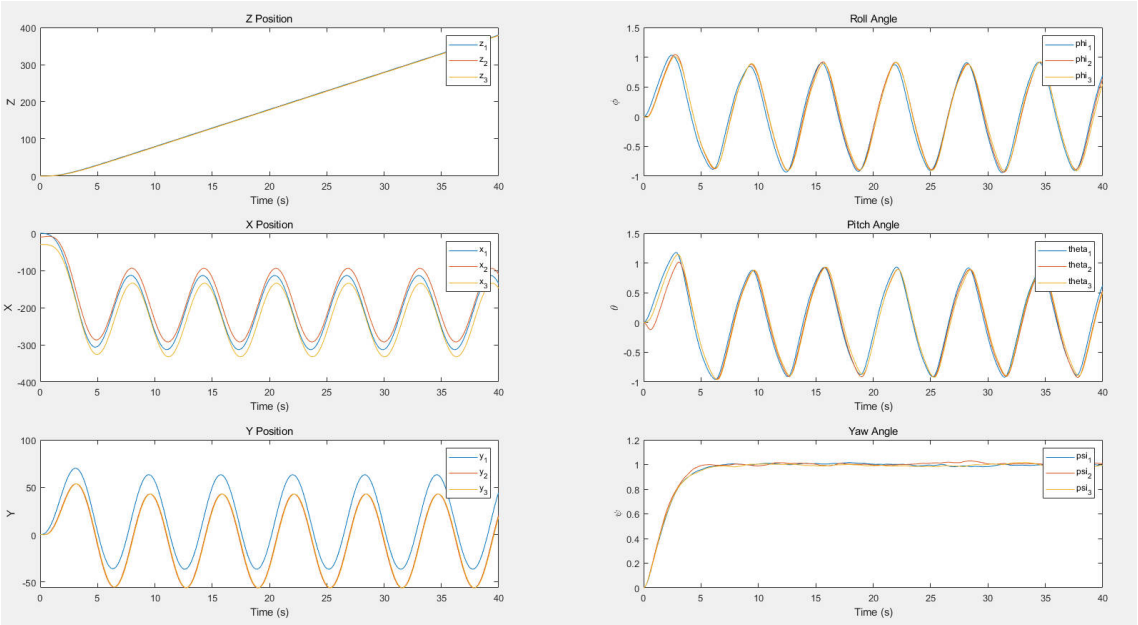

Figure 18: Evolution of the pose

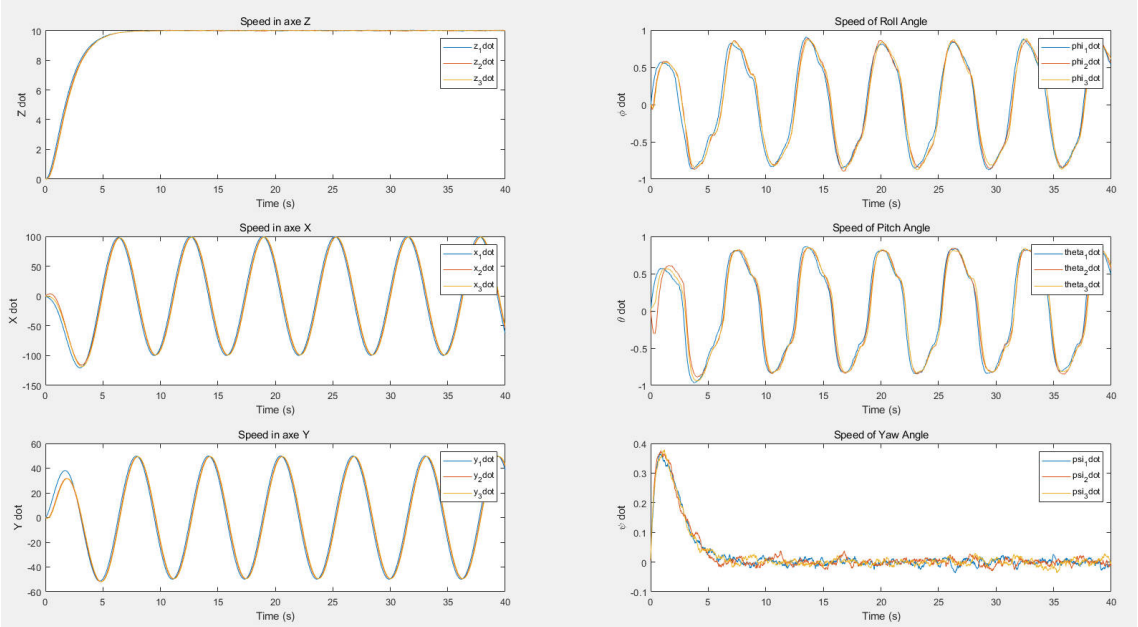

Figure 19: Evolution of the translation velocity and rotation velocity

From the above Figures 18 and 19, we can clearly see that the displacement of the drones in the Z direction increases linearly with time, the displacement evolution in the X and Y directions is similar to a sinusoidal function, the velocity in the Z direction first increases and then tends to be stable, and the evolution of the velocity in the X and Y directions is also similar to a sinusoidal function, and these are in line with the characteristics of the spirals, which are in accordance with the expected results. Drone 2 and Drone 3 also follow Drone 1 very well and the distance between the three is well maintained.

Figure 20 represents the evolution of the error between the actual distance between drone 1 and drone 2 and the predetermined desired distance and the evolution of the error between the actual distance between drone 2 and drone 3 and the predetermined desired distance. We can find that the error value in the direction of Z-axis tends to 0 with time, while the error in X-axis and Y-axis is similar to the evolution of sinusoidal function but also tends to 0. This indicates that the drones maintain the pre-set formation during the operation.

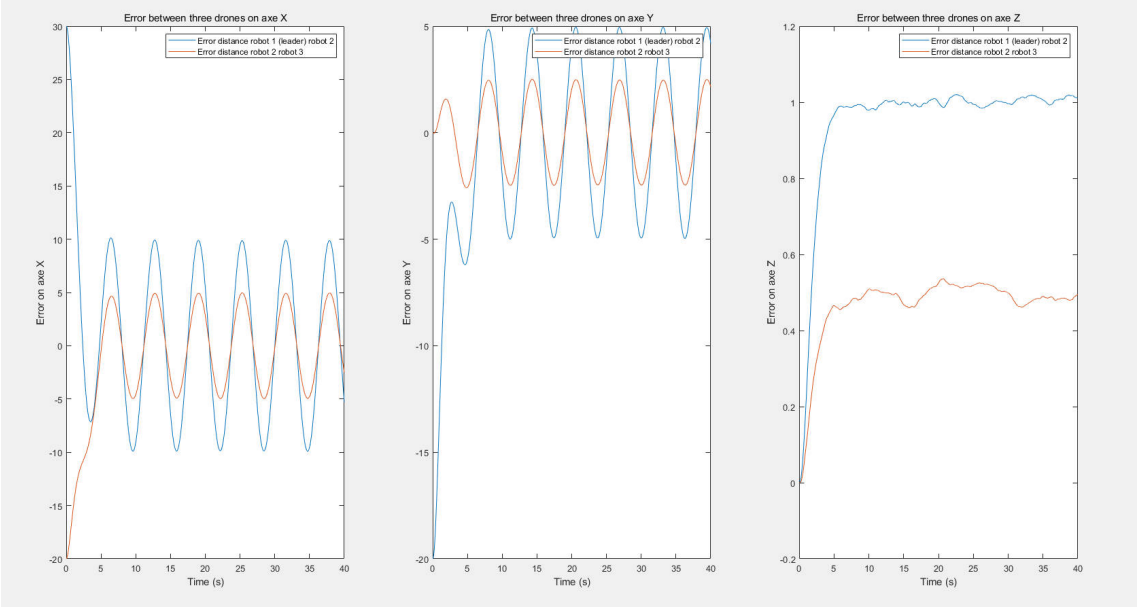

Figure 20: Evolution of the error between the actual distance the desired distance of drone's position

From the left image in Figure 21, we can see that the trend of the displacement of all three drones is consistent with the trend of the target trajectory of the centre of mass due to the presence of an item in u_1, u_2, and u_3 that controls the formation. In addition, we also find that the blue thick line and the green thick line basically coincide, which is consistent with our theory, because the error between the actual trajectory of the centre of mass and the target trajectory converges to 0 due to the existence of an item u in the controller u_1 which ensure that the drone can reach the target position.

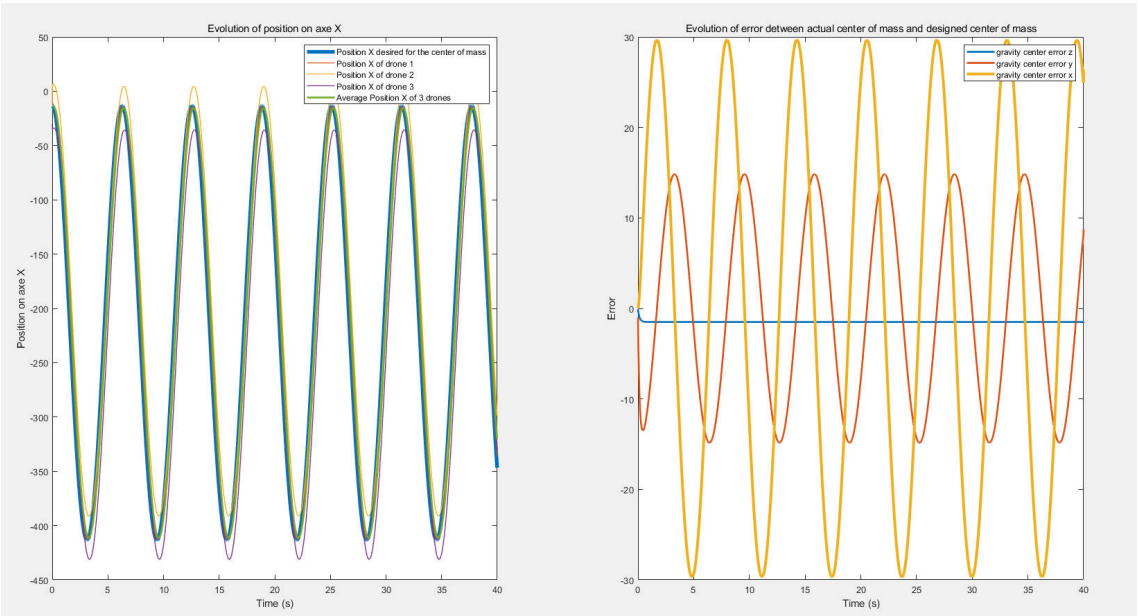

Figure 21: Evolution of the error between the actual distance between the desired distance of the center of mass

And from the right image, we can see that, in fact, there is still a certain difference between the actual distance of the centre of mass and the target distance. This difference can be reduced by increasing the value of k, but if the value of k is too large relative to the gain in the formation control term, it will negatively affect the accuracy of the formation control, so there should be a compromise for the gain k of the destination reaching control and the gain of formation maintaining control.

From Figure 22, we can see that there is a difference between the actual trajectory of the drone and the target trajectory, and the error is large. The error in the X direction and the Y direction is similar to a sinusoidal function, while the error in the Z direction increases from 0 to about 20, and then tends to stabilise. Combined with the previous analyses, we conclude that the actual trajectory of the centre of mass of the drone swarm is consistent with its target trajectory, but there is a discrepancy between the actual trajectory of individual drones and its target trajectory, which is obviously unreasonable. Based on our analyses, we believe that the problem appears in the design of the controller of the drone's multi-agent: $\dot{x_i} = u_i$. Because this controller only considers the velocity, with the formula: $x = x + \delta t \times u$, it can ensure that the displacement of the drone is continuous. But the acceleration of the drone is not taken into account, so we speculate that the velocity is not continuous, there is a sudden change in the evolution of velocity, which leads to, the target trajectory is not a trajectory that can reasonably exist in the real world because of its sudden change in the velocity.

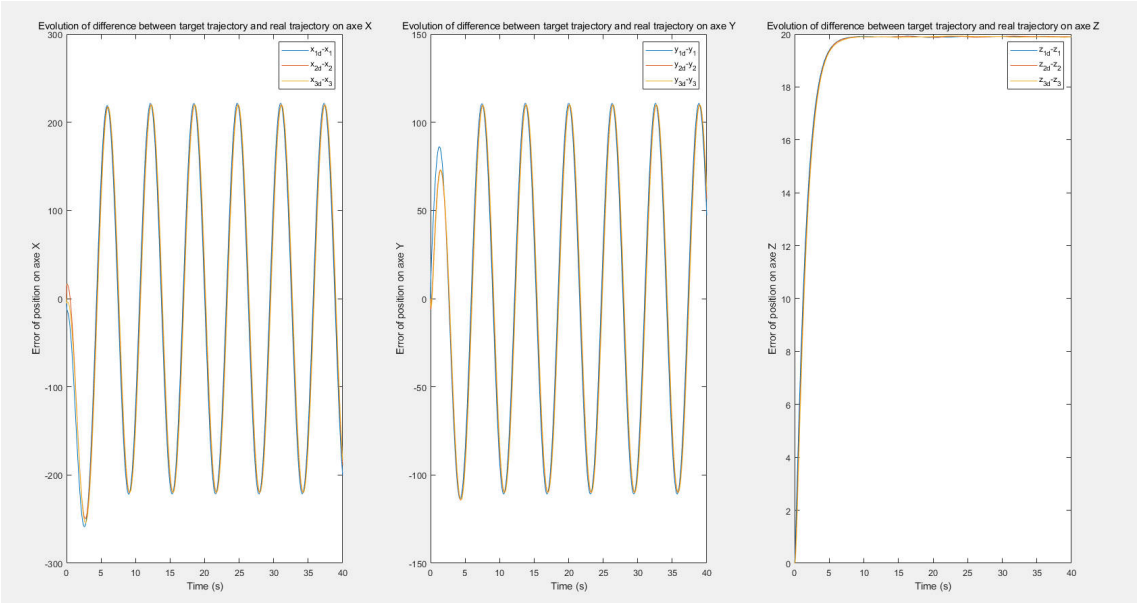

Figure 22: Evolution of the error between the actual distance and the desired distance of three drones

I found the corresponding information in the reference "$Multi_agents_RLozano.pdf$" that you put on moodle and tried to give the above explanation based on that.

4.1 Time-varying trajectory tracking

When the reference is varying in time there is a small bias in agents coordination. In this part, we consider the case of multiagent trajectory tracking of a time-varying reference. The reference, x^d_{CM}, is only given to the leader and we consider a chain topology.

Let the multi-agent system be :

$$\ddot{x}_i = u_i$$

Figure 23: Reference from Moodle

8 Results discussions

When the target trajectory is a fixed point, we can apply this type of Controllers: $\dot{x_i} = u_i$. Because there is no change in the velocity of the target trajectory involved and there is no sudden change in velocity. Let us do a test:

From the initial moment to the last, we consider the destination always a fixed point:

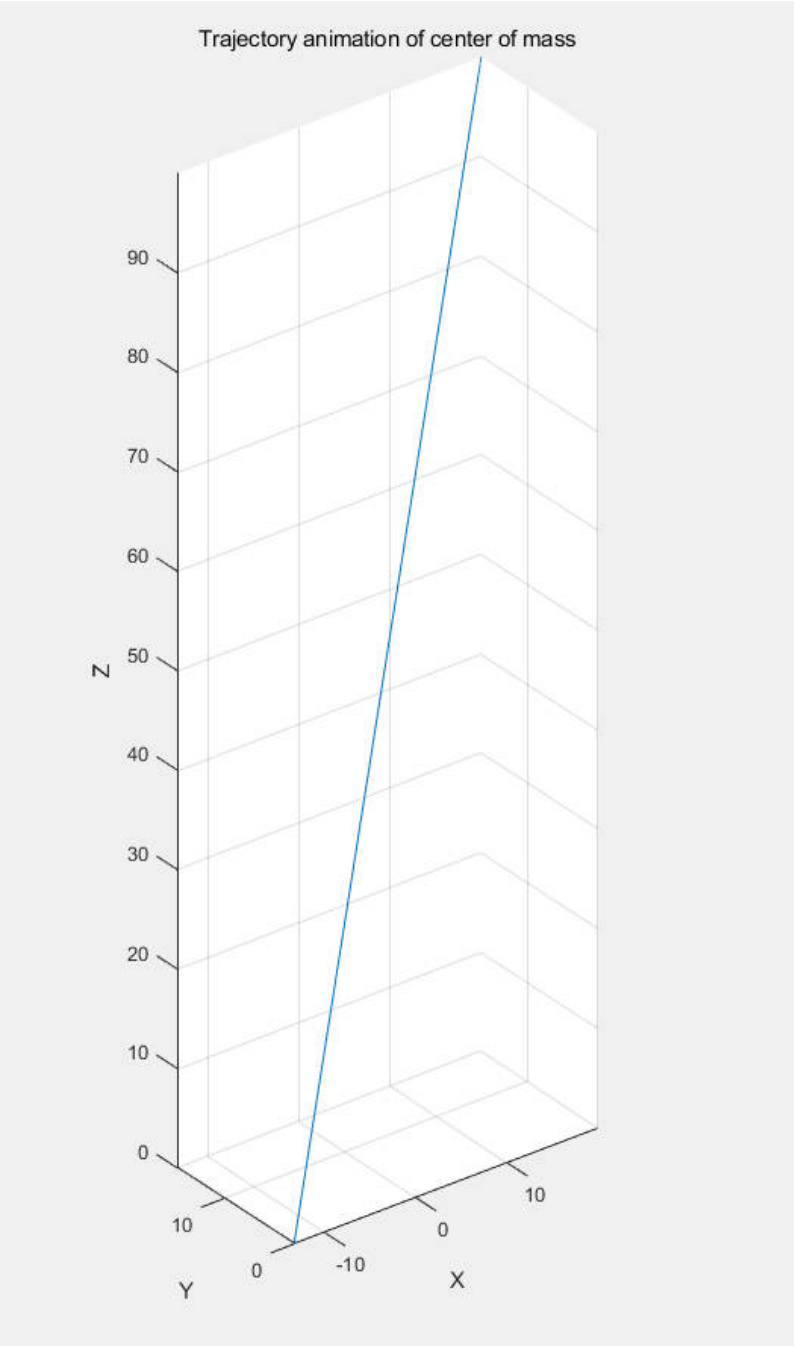

Figure 24: Target trajectory for center of mass

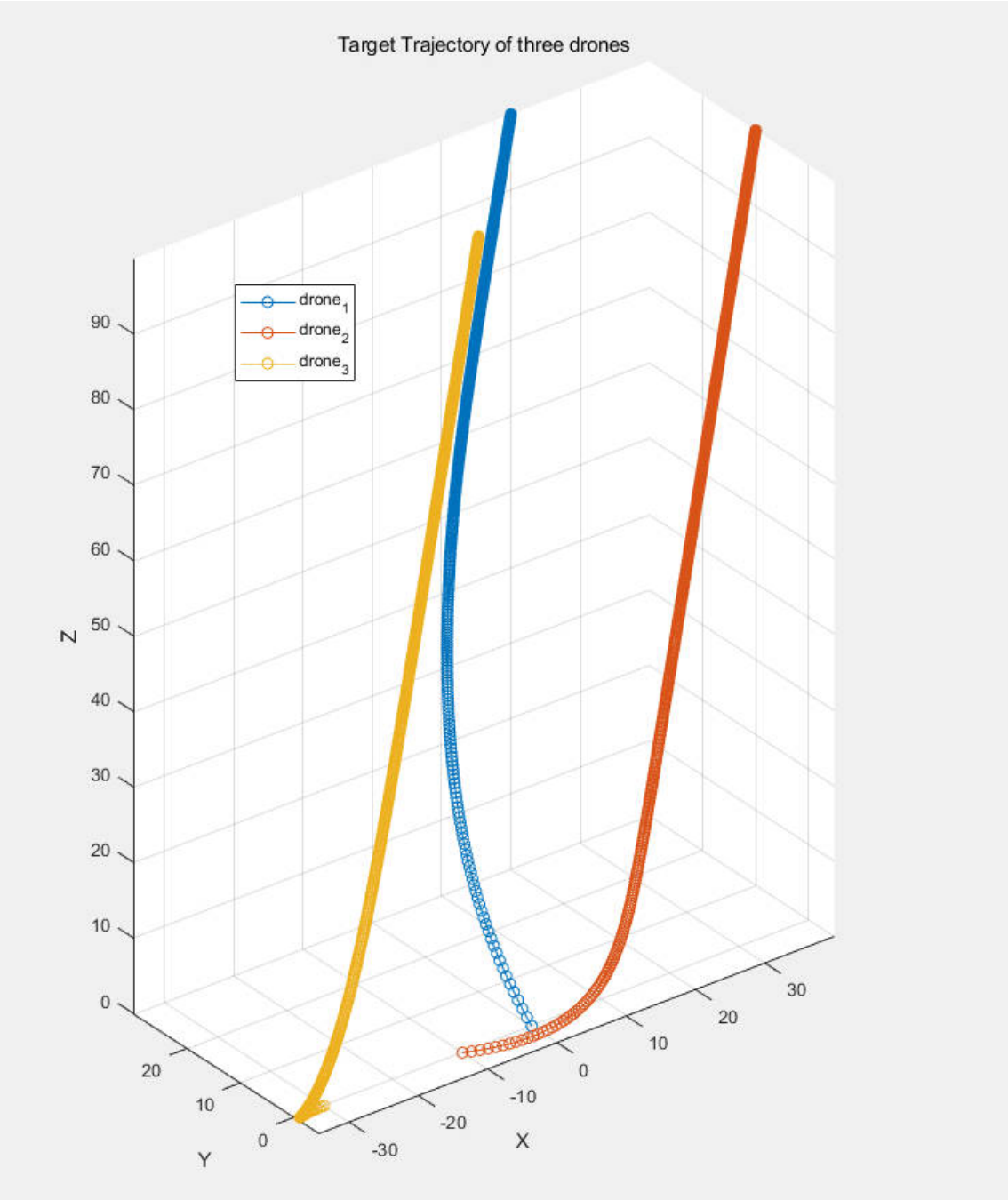

Figure 25: Target trajectories for three drones

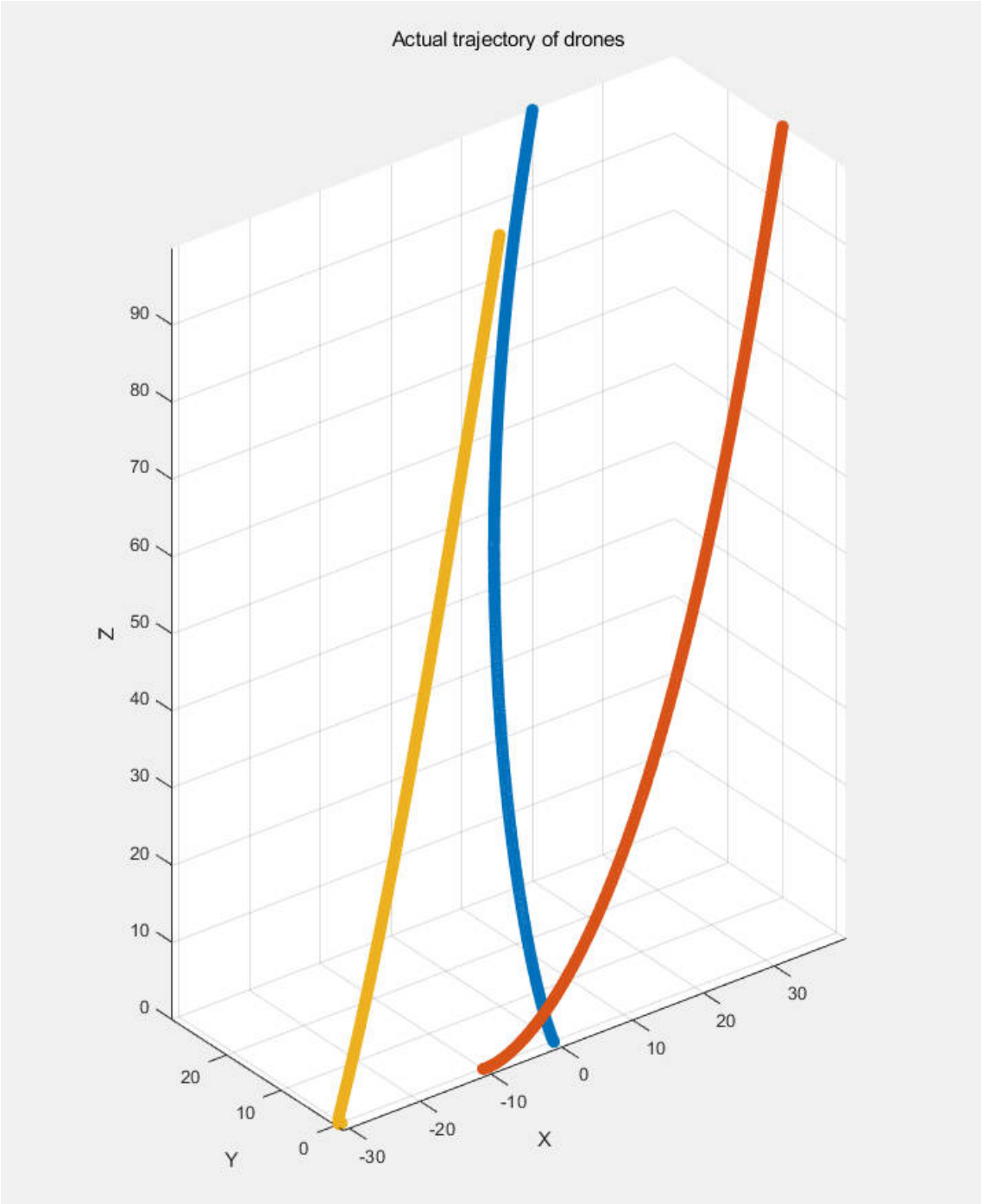

Figure 26: Actual trajectories for three drones

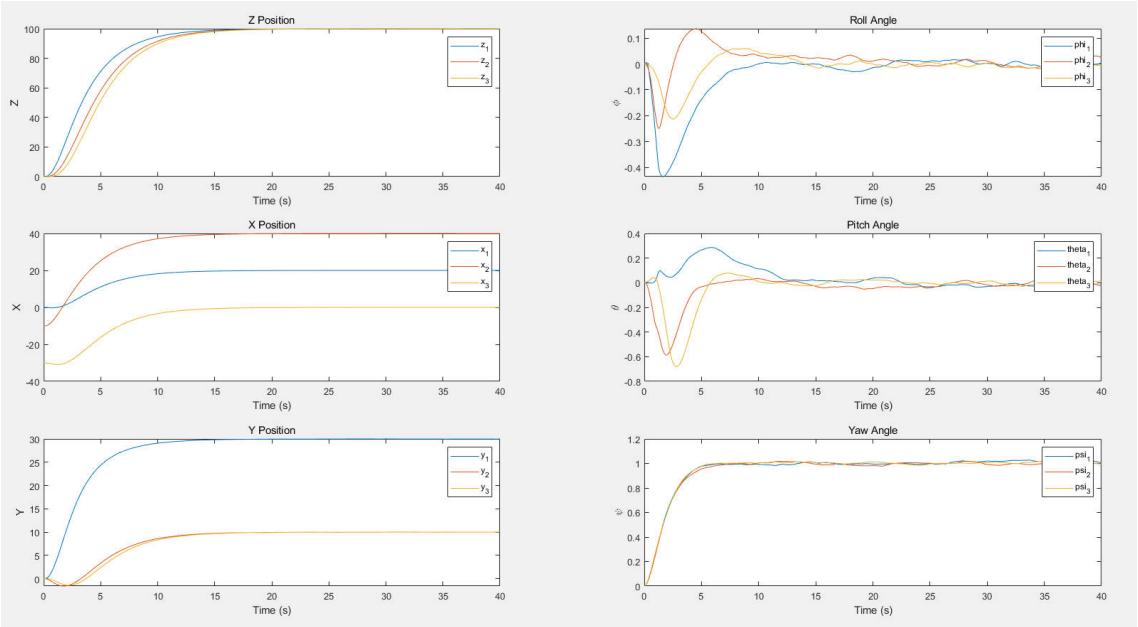

Figure 27: Evolution of the pose

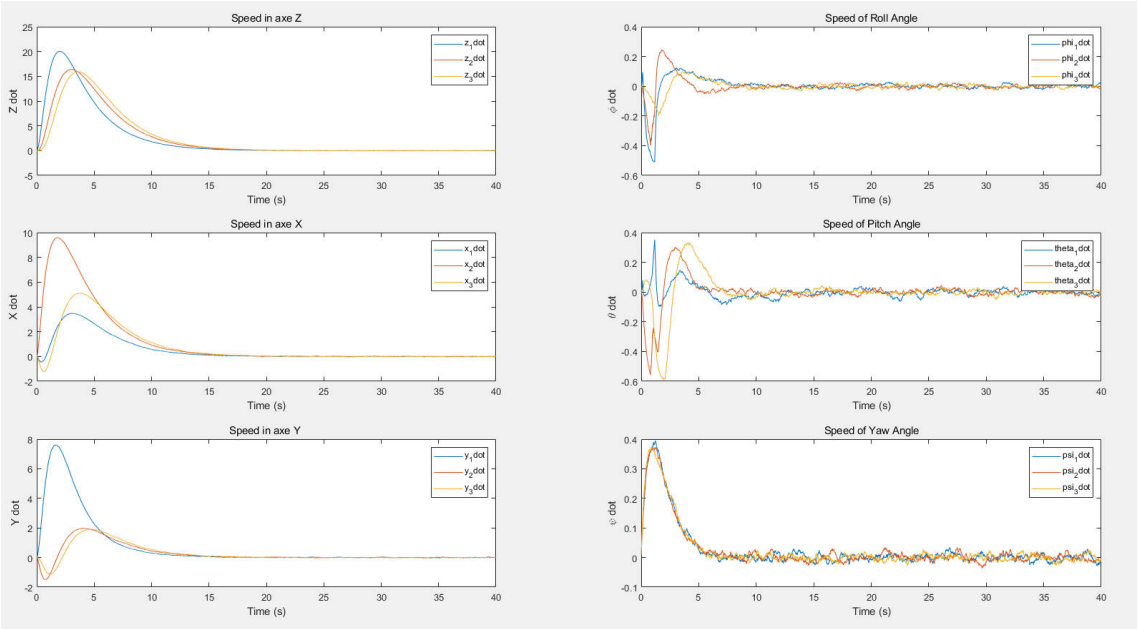

Figure 28: Evolution of the translation velocity and rotation velocity

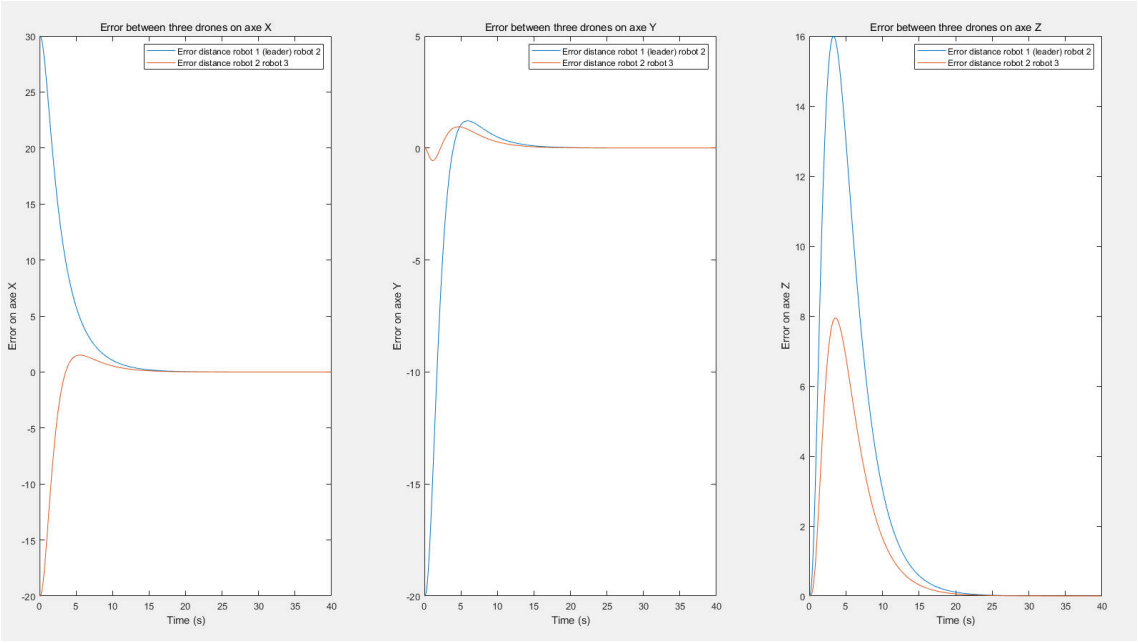

Figure 29: Evolution of the error between the actual distance the desired distance of drone's position

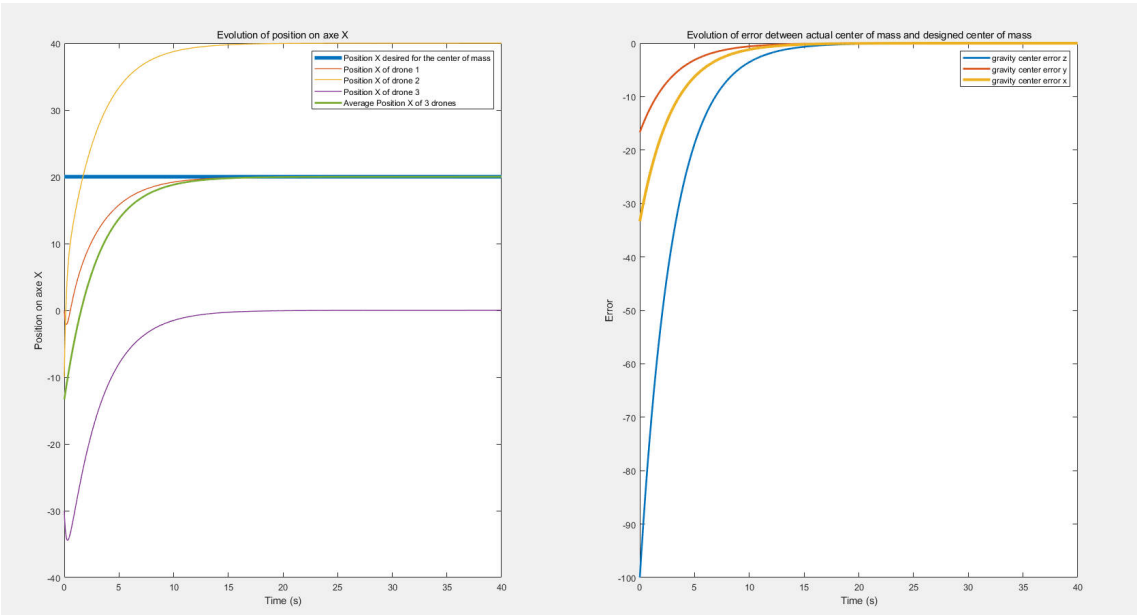

Figure 30: Evolution of the error between the actual distance between the desired distance of the center of mass

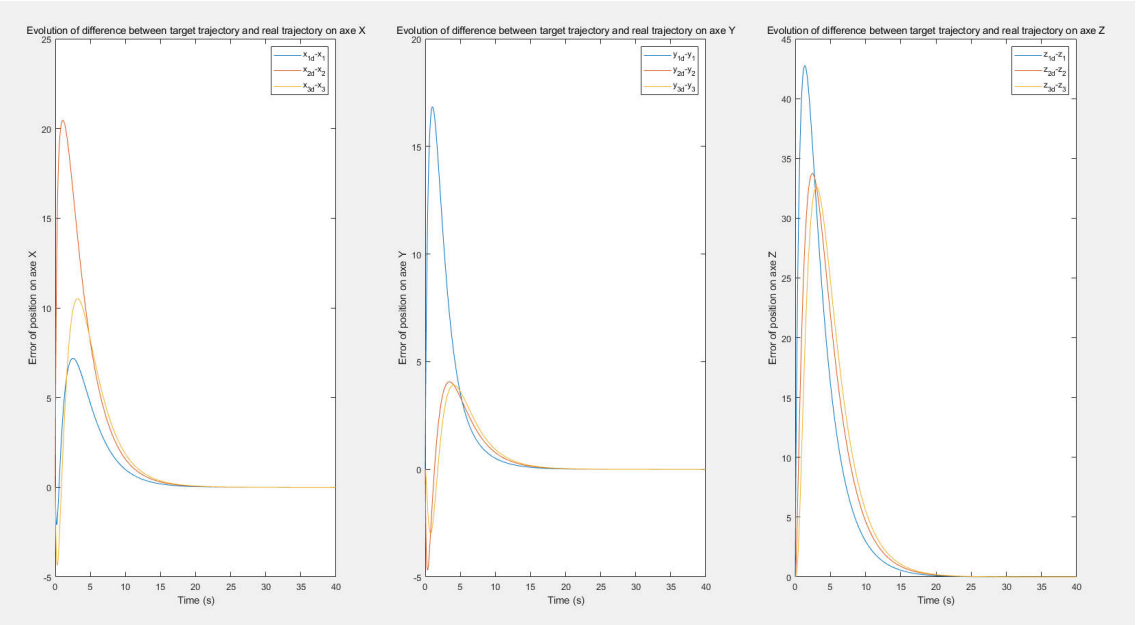

Figure 31: Evolution of the error between the actual distance and the desired distance of three drones

Analysing from Figure 23 to Figure 30, we can see that our drone control is correctly realized when the target trajectory is a point. The actual trajectory converges to the target trajectory and all the errors converge to zero. We get all drones followed and their distance maintained. So as we indicated before, the controller$\dot{x_i} = u_i$ is applicable only when the destination is a fixed point rather than a varying trajectory.

When the trajectory is a moving trajectory over time, we need to modify the controller like including a new term to control the acceleration in addition to the original one, and the control equation will be roughly like:

$$k_1 \times \dot{x_i} + k_2 \times \ddot{x_i} = u_i \tag{71}$$

9 Conclusion

In this project, the first part, drone dynamics modelling and Backstepping control is well done by us. But for the second part of the multi-intelligence body control, just two days before the deadline of the project, when writing the project report, we found the problem of choosing the wrong controller model. Due to the time limit, we could only figure out the problem, analyse the problem and try to give a solution from the theoretical point of view, but did not complete the modification of this part of the code, that is a pity. But we still learn a lot from this project, many thanks to the course ARS5, real useful and inspiring.