16665 Robot Mobility: Air Mobility Project

Chang Shi

November 7, 2019

Q1

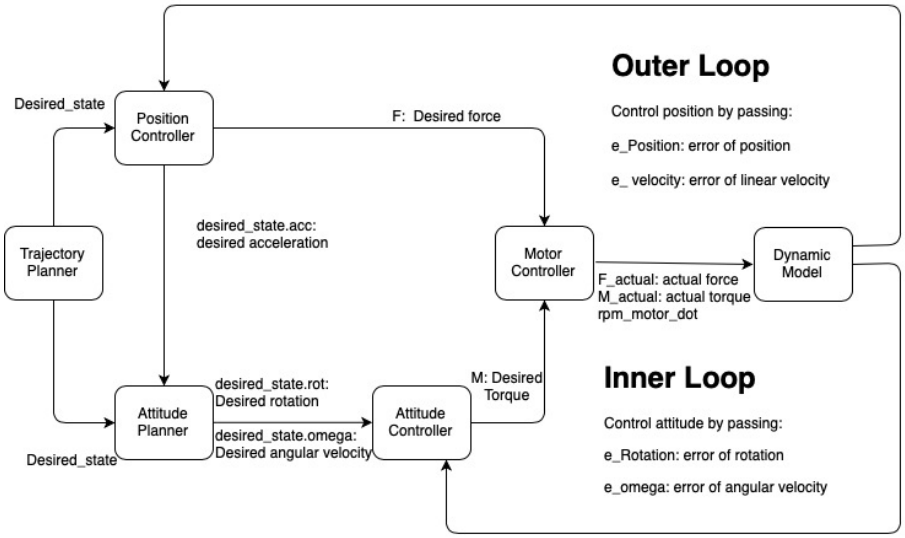


Figure 1: System Diagram

Q2

First set of PD controller:
Position controller(outer loop)

$$Kp1 = 17; Kd1 = 6.6;$$

$$Kp2 = 17; Kd2 = 6.6;$$

$$Kp3 = 20; Kd3 = 9;$$

Attitude control(inner loop)

$$Kpphi = 190; Kdphi = 30;$$

$$Kptheta = 198; Kdtheta = 30;$$

$$Kppsi = 80; Kdpsi = 17.88;$$

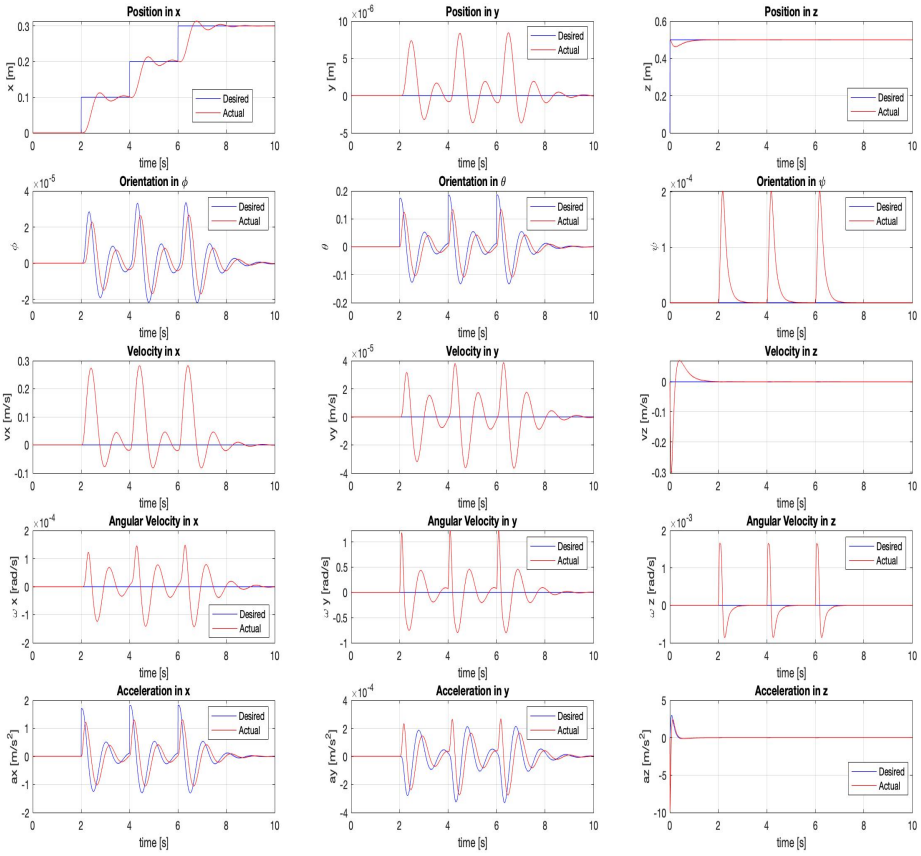


Figure 2: The desired pose and the actual pose

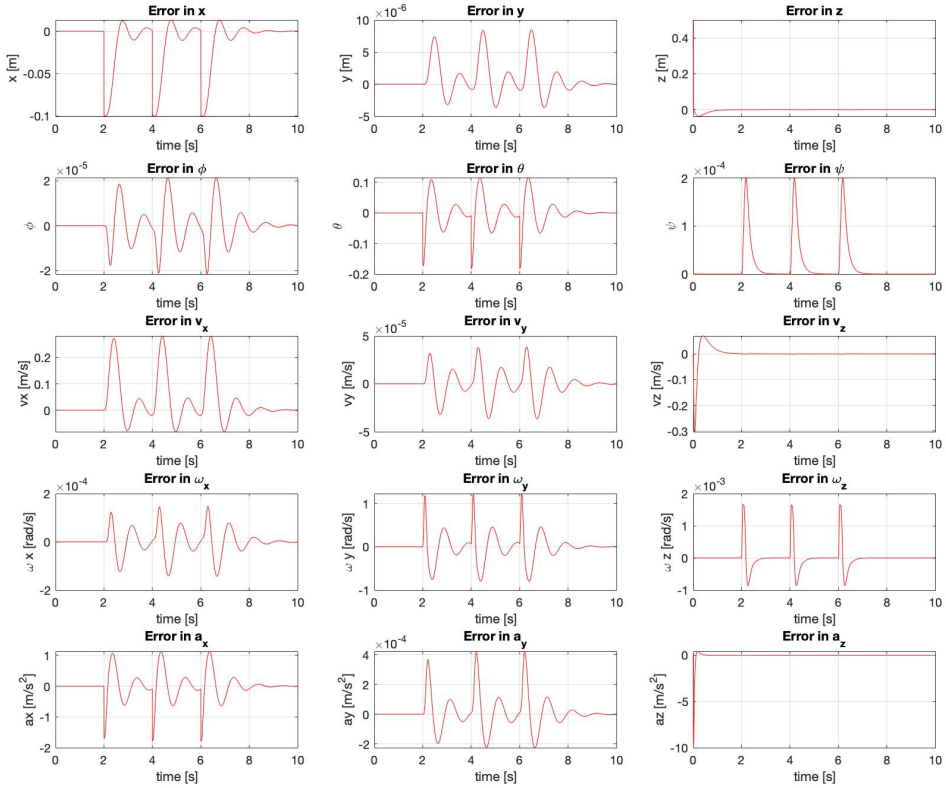


Figure 3: Error between the desired pose and the actual pose

The new waypoints are sent to the system in Trajectory Planner as the desired position in desired_state, then would be used later in position error for position control.

From the error figure, we can see the robot converge to each waypoint after oscillate about it for a period.

Second set of PD controller:
Position controller(outer loop)

$$Kp1 = 27; Kd1 = 4.6;$$

$$Kp2 = 27; Kd2 = 4.6;$$

$$Kp3 = 30; Kd3 = 6;$$

Attitude control(inner loop)

$$Kpphi = 200; Kdphi = 20;$$

$$Kptheta = 218; Kdtheta = 20;$$

$$Kppsi = 100; Kdpsi = 12.88;$$

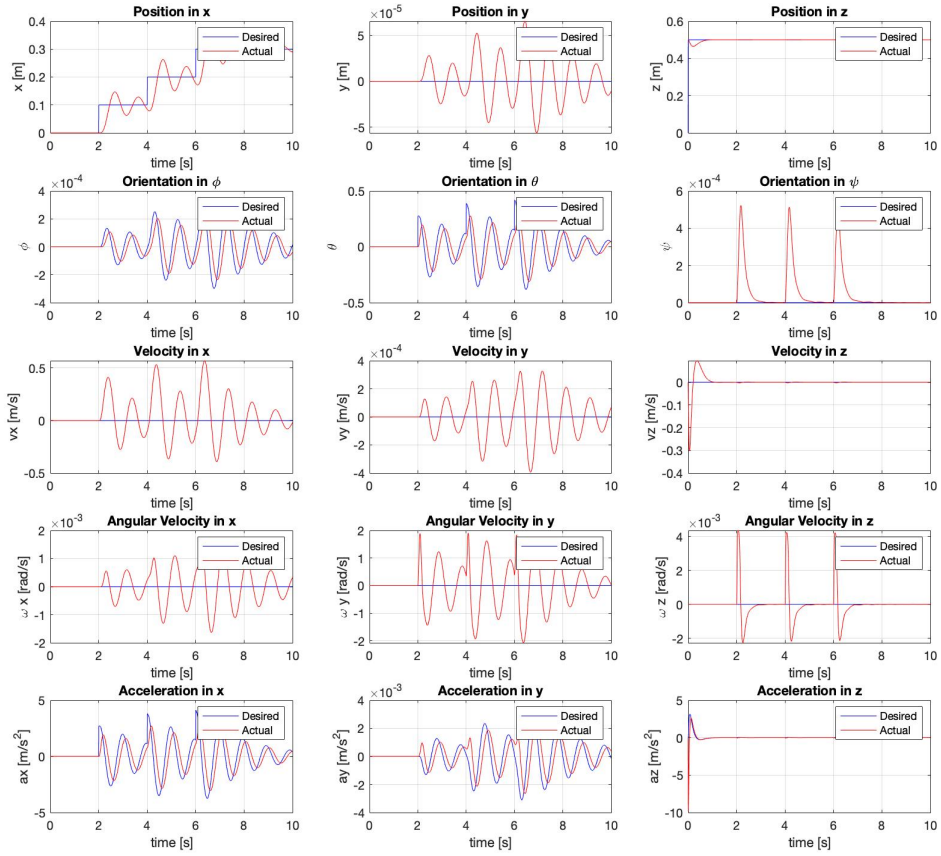


Figure 4: The desired pose and the actual pose

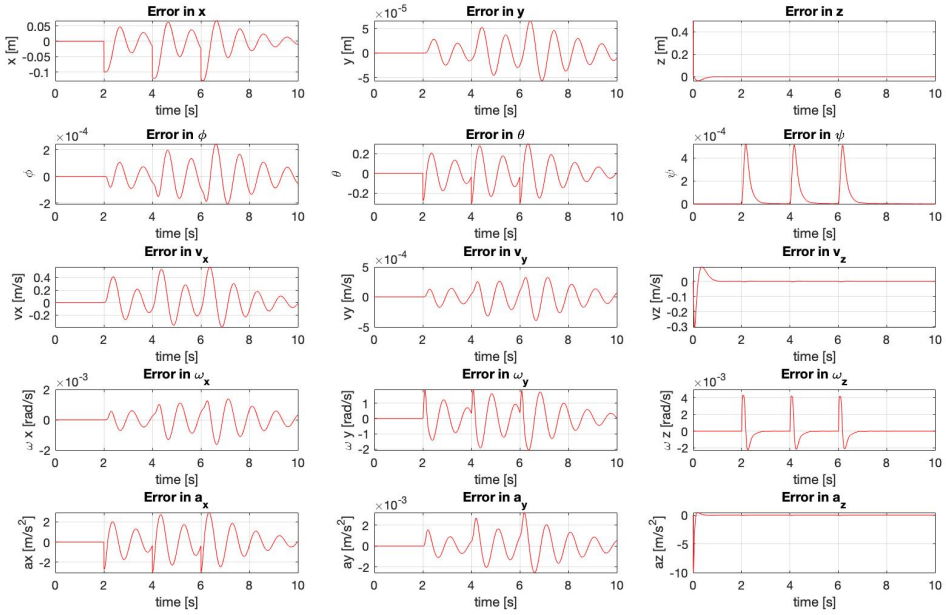


Figure 5: Error between the desired pose and the actual pose

We can see from the figure that basically, the gain of position control the position oscillation, while gain of angle control the attitude oscillate. Increasing the gain of P controller, will make the system reacts faster(rising time smaller), while increasing the gain of D controller, will make the system oscillation smaller(percent overshoot smaller) and damped out faster(settling time smaller).

Q3

First set of PD controller:
Position controller(outer loop)

$$Kp1 = 17; Kd1 = 6.6;$$

$$Kp2 = 17; Kd2 = 6.6;$$

$$Kp3 = 20; Kd3 = 9;$$

Attitude control(inner loop)

$$Kpphi = 190; Kdphi = 30;$$

$$Kptheta = 198; Kdtheta = 30;$$

$$Kppsi = 80; Kdpsi = 17.88;$$

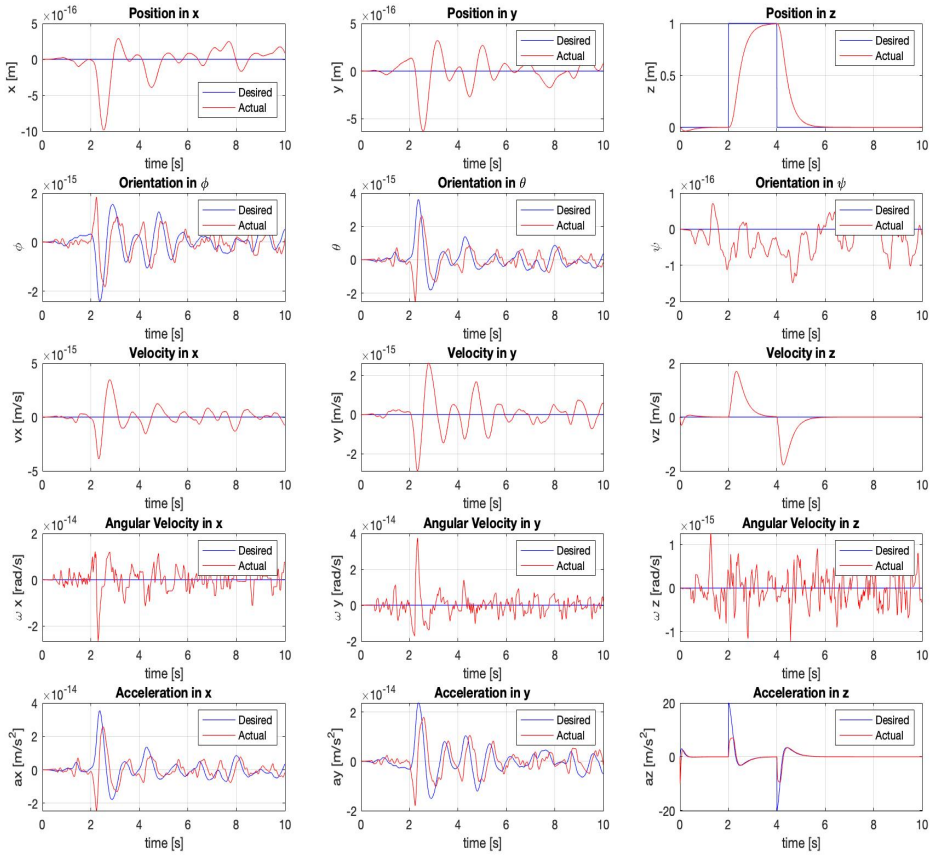


Figure 6: The desired pose and the actual pose

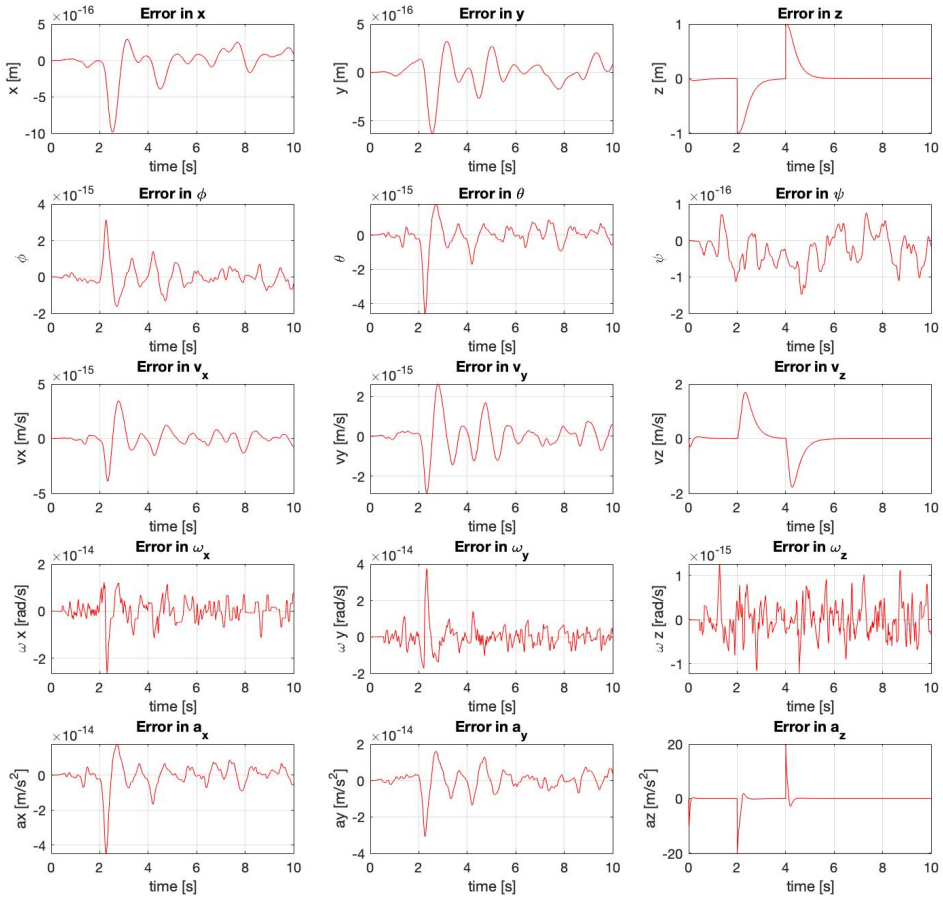


Figure 7: Error between the desired pose and the actual pose

From the error figure, we can see the robot converge to each waypoint after oscillate about it for a period.
Then we increase Kp and Kd of outer loop:

Second set of PD controller:
Position controller(outer loop)

$$Kp1 = 27; Kd1 = 8.6;$$

$$Kp2 = 27; Kd2 = 8.6;$$

$$Kp3 = 30; Kd3 = 12;$$

Attitude control(inner loop)

$$Kpphi = 190; Kdphi = 30;$$

$$Kptheta = 198; Kdtheta = 30;$$

$$Kppsi = 80; Kdpsi = 17.88;$$

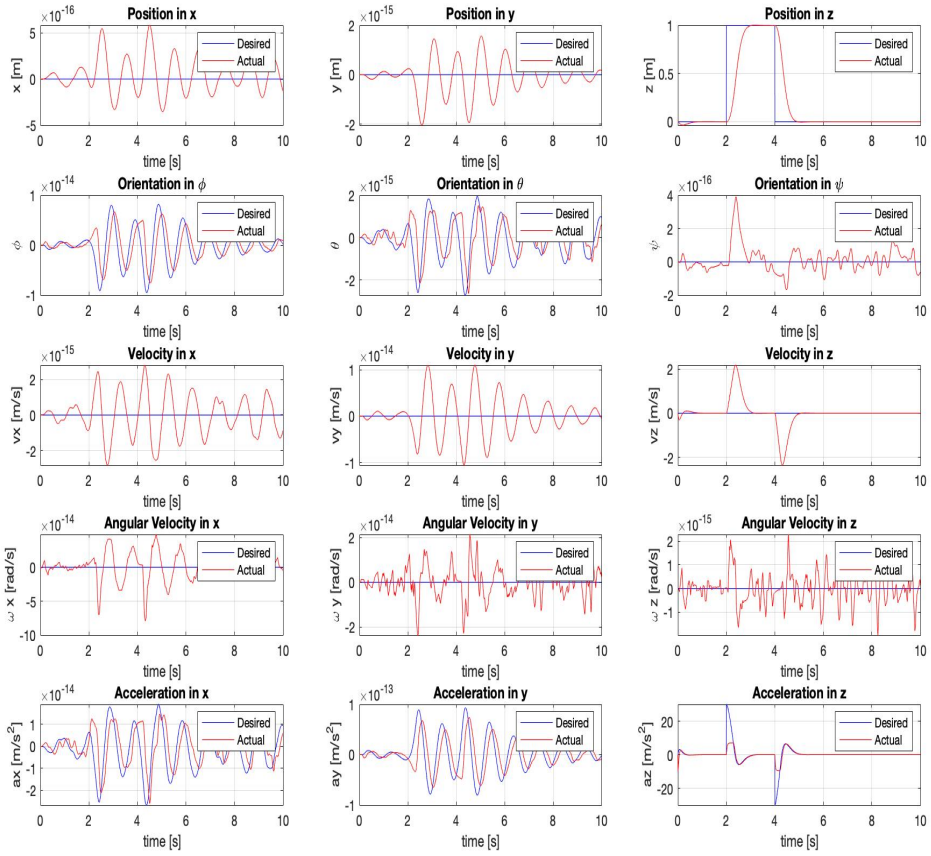


Figure 8: The desired pose and the actual pose

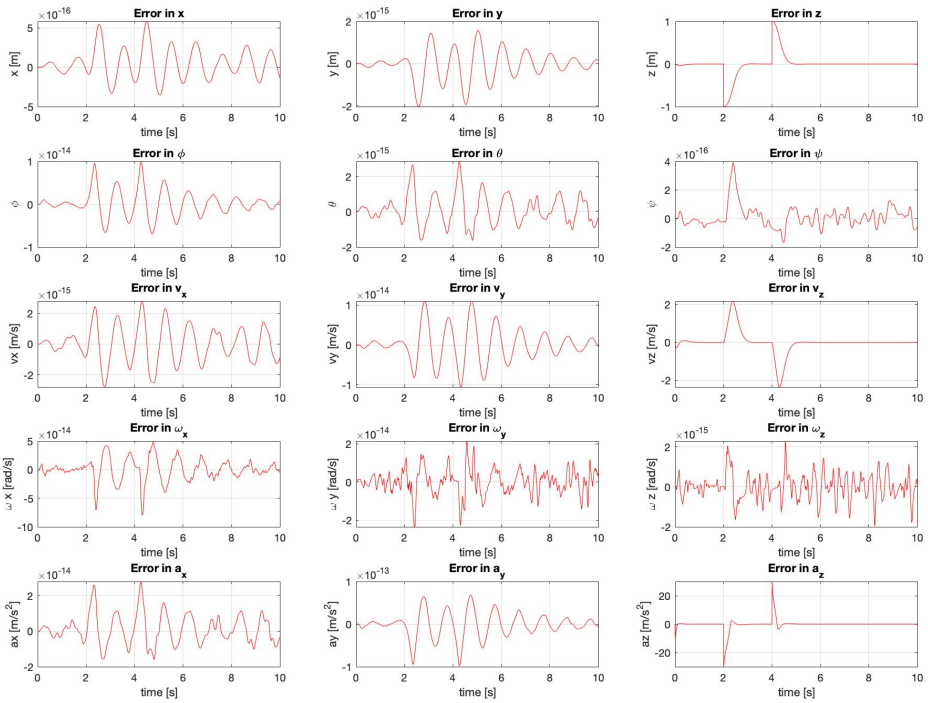


Figure 9: Error between the desired pose and the actual pose

We can see from the graphs that increasing the gains of outer loop, can decrease the rising time, settling time of the position tracking part of the system.

Q4

A state machine is developed by switching mode in the trajectory planner code.

```
if question == 4
    % should add variable 'Mode' to switch, but just hard coded here
    % preparation for question 5

    % idle
    trajectory_state = zeros(15, max_iter);

    % takeoff
    trajectory_state = zeros(15, max_iter);
    desired_height = 5;
    desired_velocity = 1;
```

```
12      for i = 1:max_iter
13          trajectory_state(3, i) = (i-1) * desired_height / ...
                (max_iter - 1);
14          trajectory_state(6, i) = desired_velocity;
15      end
16
17      % hover
18      trajectory_state = zeros(15, max_iter);
19      desired_height = 5;
20      for i = 1:max_iter
21          trajectory_state(3, i) = desired_height;
22      end
23
24      % tracking
25      sample_rate = 1 / time_step;
26      trajectory = waypointTrajectory(waypoints(1:3,:)', ...
27          'TimeOfArrival', waypoint_times, ...
28          'SampleRate', sample_rate);
29
30      trajectory_state = zeros(15, max_iter);
31
32      for i = 1:max_iter
33          [position, orientation, velocity, acceleration, ...
                angularVelocity] = trajectory();
34          trajectory_state(1:3, i) = position;
35          trajectory_state(4:6, i) = velocity;
36          trajectory_state(7:9, i) = quat2eul(orientation);
37          trajectory_state(10:12, i) = angularVelocity;
38          trajectory_state(13:15, i) = acceleration;
39      end
40
41      % landing
42      trajectory_state = zeros(15, max_iter);
43      desired_height = 5;
44      desired_velocity = -1;
45      for i = 1:max_iter
46          trajectory_state(3, i) = desired_height - (i-1) * ...
                desired_height / (max_iter - 1);
47          trajectory_state(6, i) = desired_velocity;
48      end
49  end
```

Q5

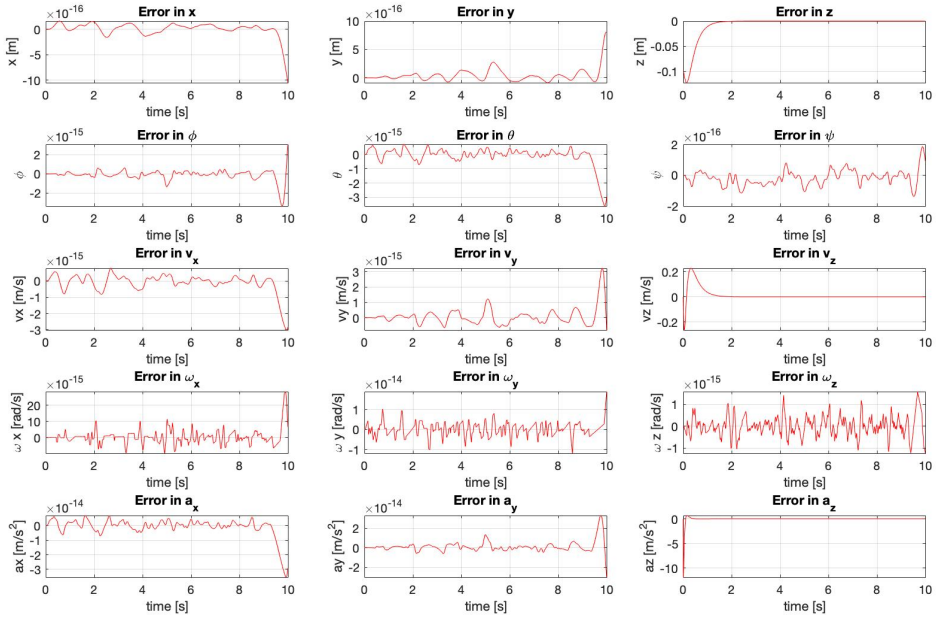


Figure 10: Error between the desired pose and actual pose with 0 degree heading

Since there is no meaning to discuss the disturbance in direction of x and y, the following part only discuss about z direction.

The rise time associated with position z is 1.0290s. The settling time associated with position z is 1.6587s.

The rise time associated with linear velocity v_z is 0.1398s. The settling time associated with linear velocity v_z is 1.7901s.

The steady-state value is 1.1m, and the maximum percent overshoot is 0.

Now provide a waypoint at the same position but with heading of 15 deg. The tracking error is as below:

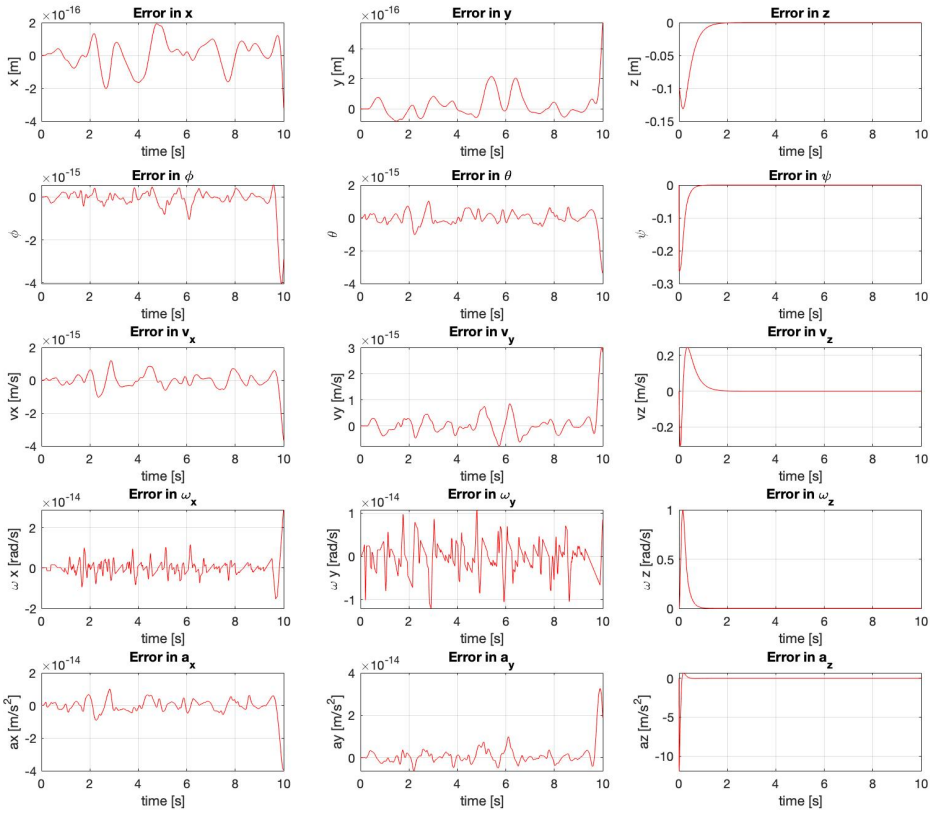


Figure 11: Error between the desired pose and actual pose with 15 degree heading

The similar time domain performance characteristics of the heading controller are as below:

The rise time associated with position z is 1.0750s. The settling time associated with position z is 1.6836s.

The rise time associated with linear velocity v_z is 0.1640s. The settling time associated with linear velocity v_z is 1.7916s.

The steady-state value is 1.1m, and the maximum percent overshoot is 0. Increasing the gain values K_p and K_d of position controller will decrease the rise time and settling time associated with position as well as linear velocity. For example, we increase the gains to

$$Kp1 = 27; Kd1 = 9.6;$$

$$Kp2 = 27; Kd2 = 9.6;$$

$$Kp3 = 30; Kd3 = 23;$$

The rise time associated with position z changes from original value 1.0750s to 0.9755s. The settling time associated with position z changes from original value 1.6836s to 1.5953s.
The rise time associated with linear velocity v_z changes from original value 0.1640s to 0.1438s. The settling time associated with linear velocity v_z changes from original value 1.7916s to 1.7091s.

Q6

Repeat Q2 with LQR, the result is as below:

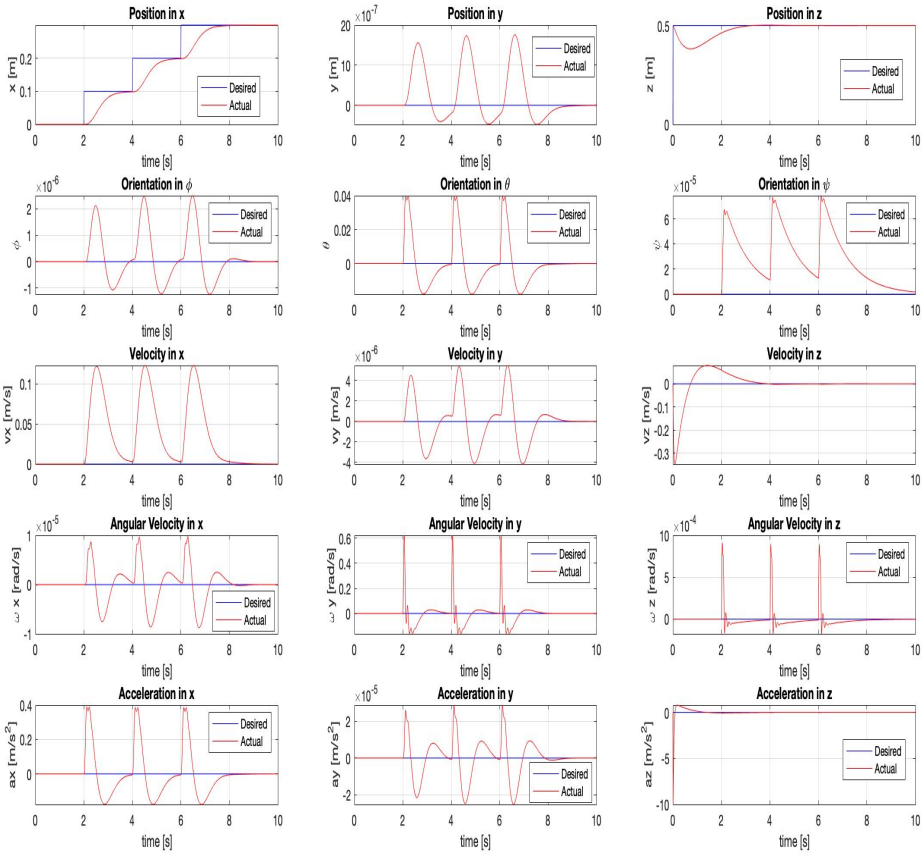


Figure 12: The desired pose and the actual pose for hovering

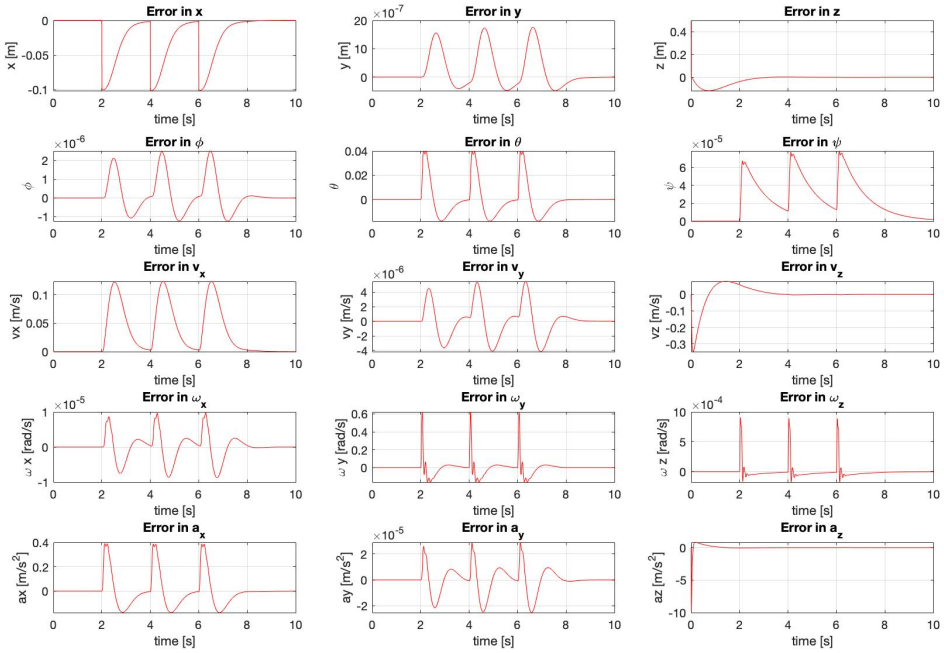


Figure 13: Error between the desired pose and the actual pose for hovering

In general, performance differs in terms of error response characteristics while using PD controller and LQR controller. Here in PD controller, the percent overshoot is around 0, while in LQR the percent overshoot is around 45.5%. And the overall rise time and settling time for both position and linear velocity in LQR is larger that those in PD controller.

Repeat Q3 with LQR, the result is as below:

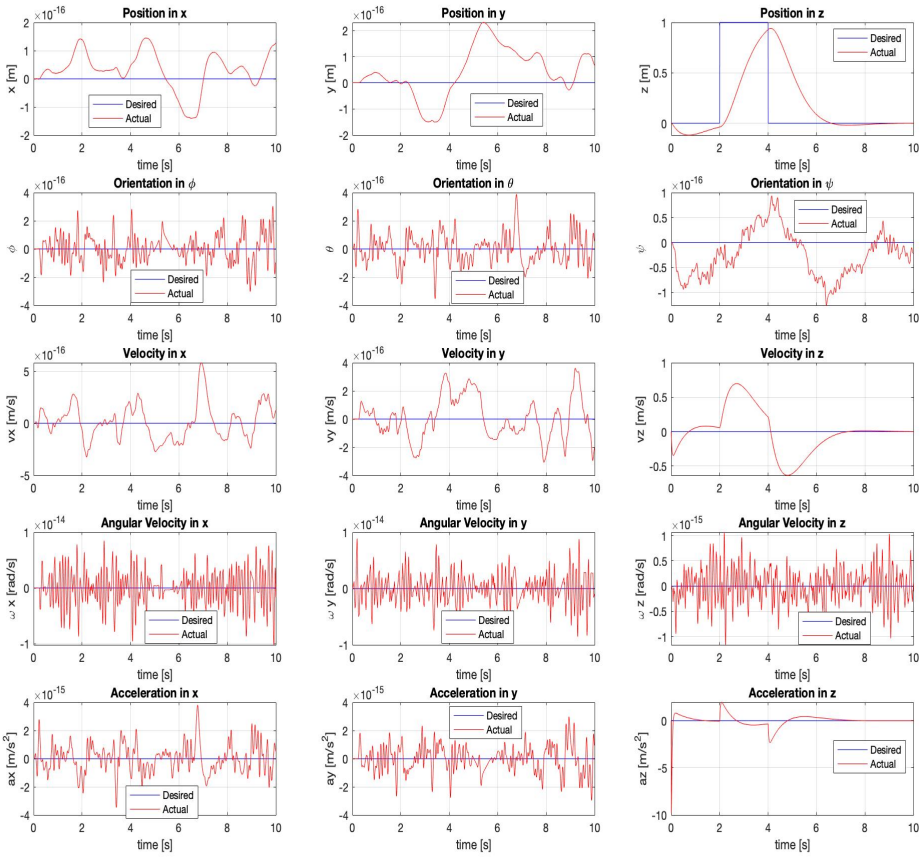


Figure 14: The desired pose and the actual pose for line-tracking

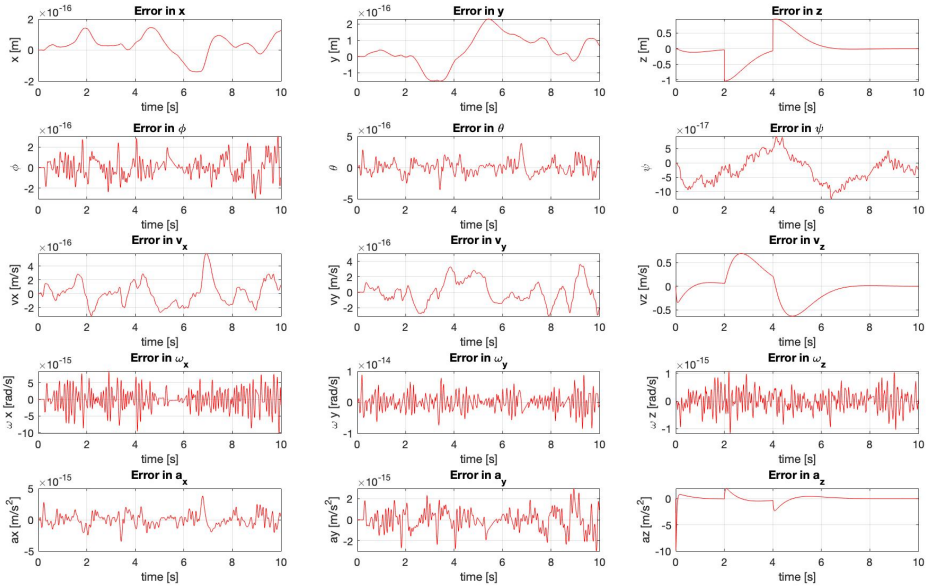


Figure 15: Error between the desired pose and the actual pose for line-tracking

In general, performance differs in terms of error response characteristics while using PD controller and LQR controller. Here in PD controller, the percent overshoot is around 0, while in LQR the has larger undershoot. And the overall rise time and settling time for both position and linear velocity in LQR is larger that those in PD controller.

Repeat Q5 with LQR, the result is as below:

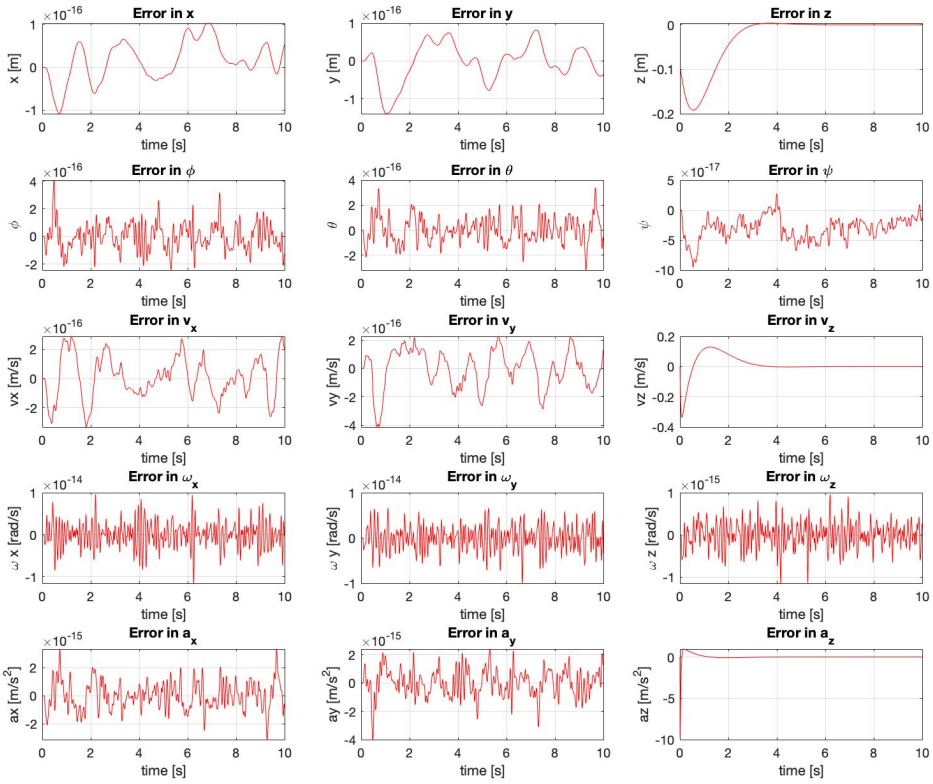


Figure 16: Error between the desired pose and the actual pose with 0 degree heading

Same as Q5, we only discuss characteristics associated with z direction.
The rise time associated with position z is 2.6237s. The settling time associated with position z is 4.7409s.

The rise time associated with linear velocity v_z is 0.5475s. The settling time associated with linear velocity v_z is 3.4867s.

The steady-state value is 1.1m, and the maximum percent overshoot is 37.02%.

Now provide a waypoint at the same position but with heading of 15 deg. The tracking error is as below:

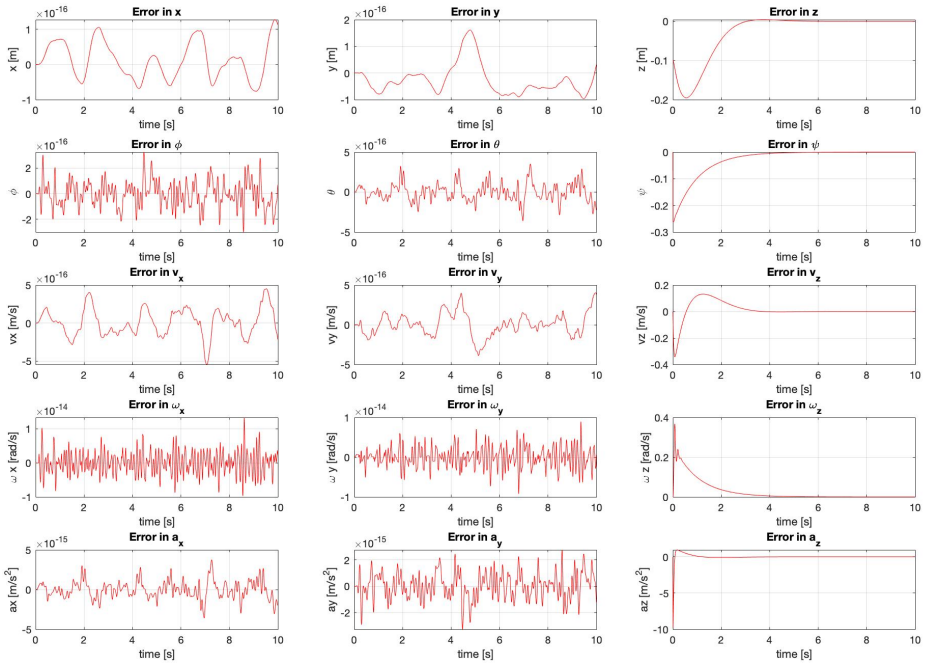


Figure 17: Error between the desired pose and the actual pose with 0 degree heading

The similar time domain performance characteristics of the heading controller are as below:

The rise time associated with position z is 2.6357s. The settling time associated with position z is 4.7476s.

The rise time associated with linear velocity v_z is 0.5543s. The settling time associated with linear velocity v_z is 3.4941s.

The steady-state value is 1.1m, and the maximum percent overshoot is 37.70%. Increasing Q and decreasing R of position controller will decrease the rise time and settling time associated with position as well as linear velocity.
In general, performance differs in terms of error response characteristics while using PD controller and LQR controller. Here in PD controller, the percent overshoot is 0, while in LQR the percent overshoot is around 37%. And the overall rise time and settling time for both position and linear velocity in LQR is larger that those in PD controller.

Q7

Choosing 7 time scaling for the robot to rise from [0,0,1] to [0,0,10], we get performance plots as below:

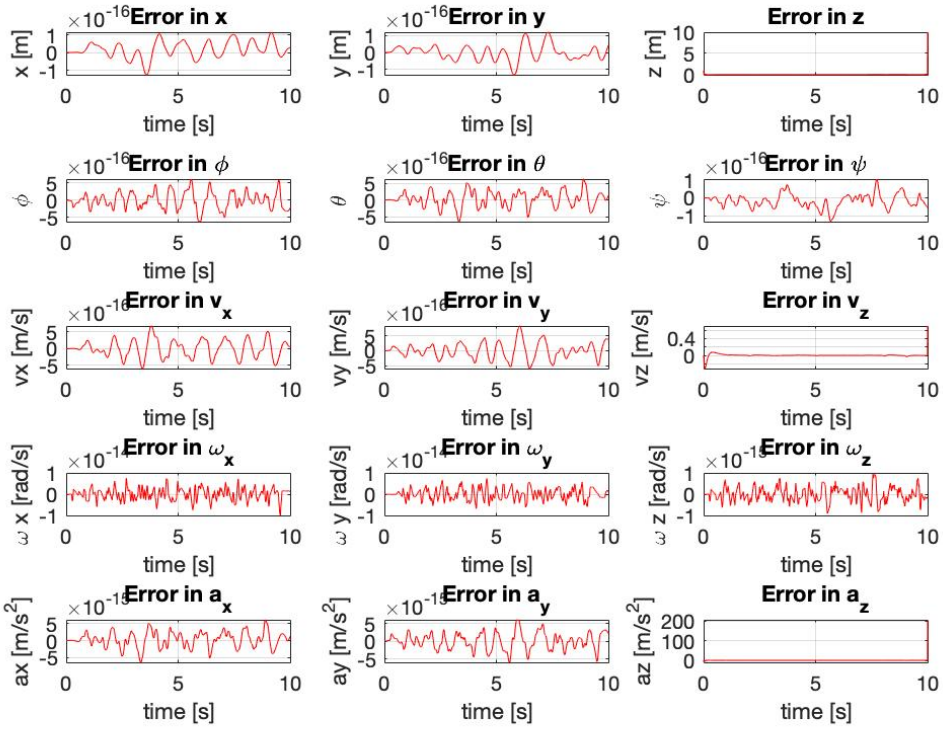


Figure 18: Error with time scaling of 1s

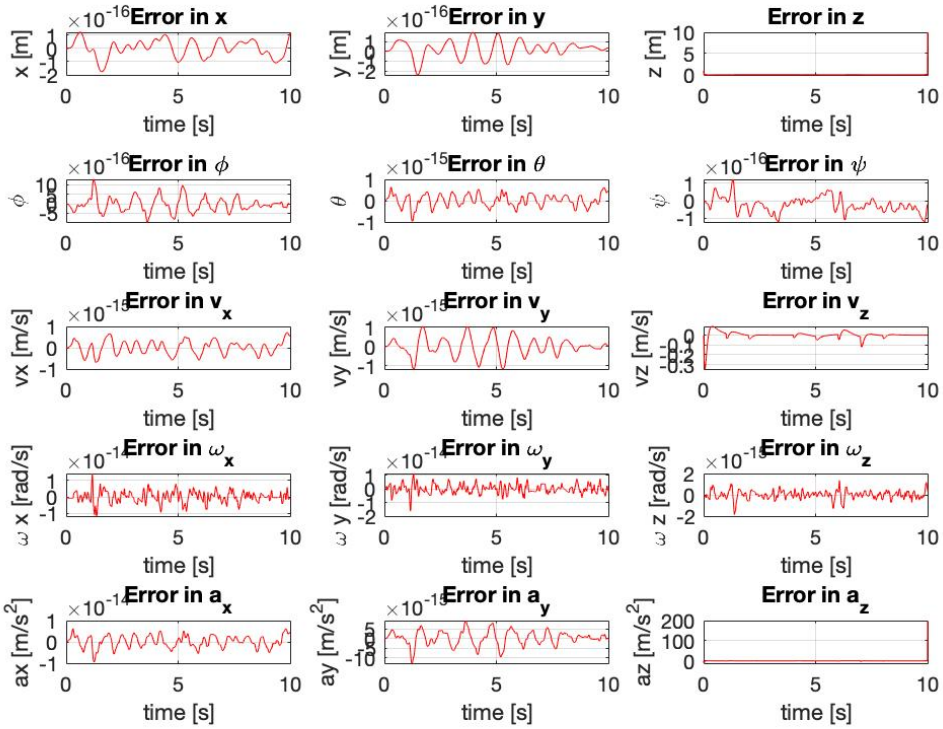


Figure 19: Error with time scaling of 2s

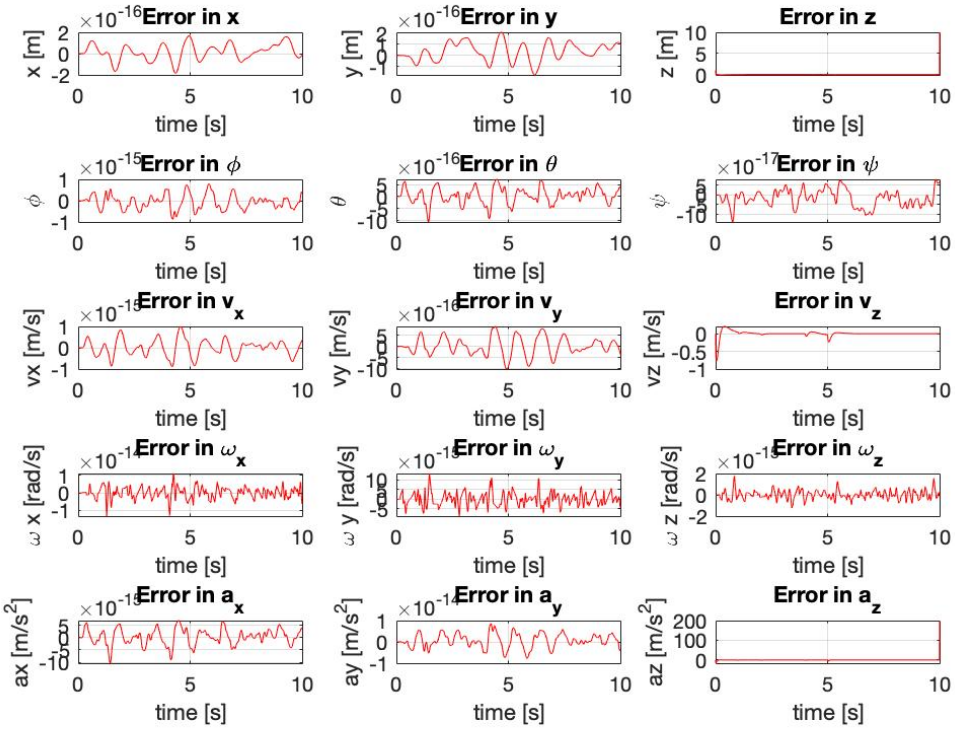


Figure 20: Error with time scaling of 3s

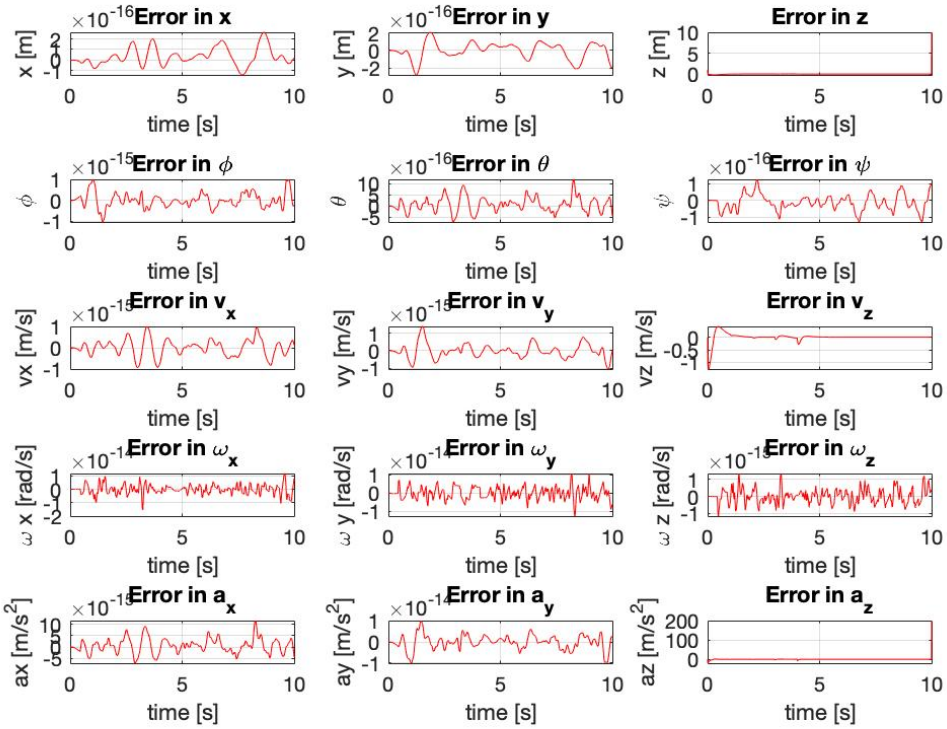


Figure 21: Error with time scaling of 4s

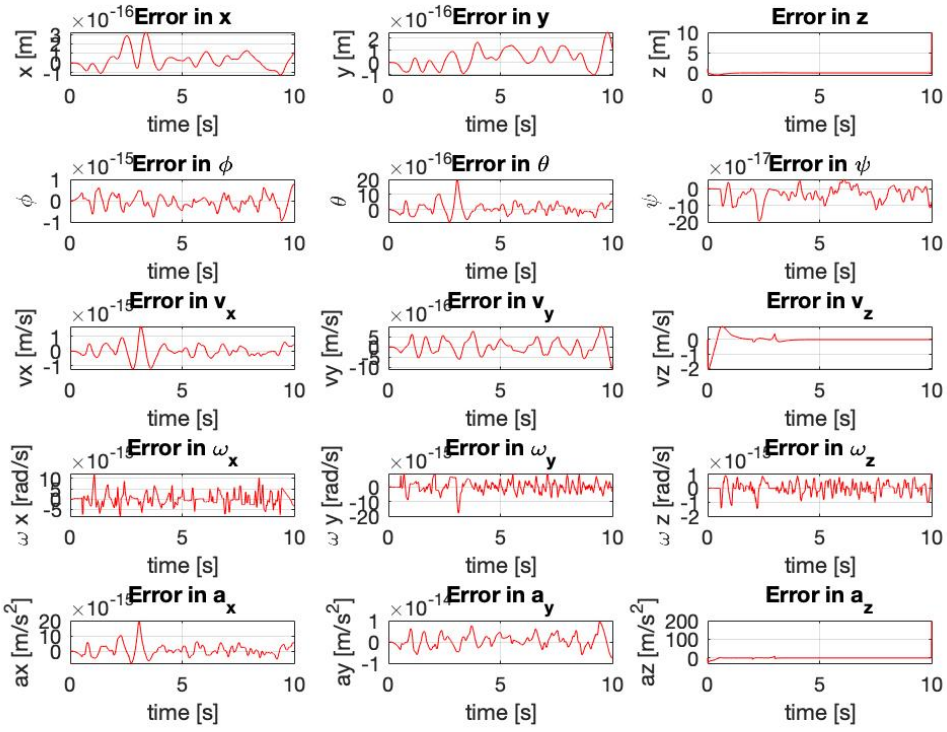


Figure 22: Error with time scaling of 5s

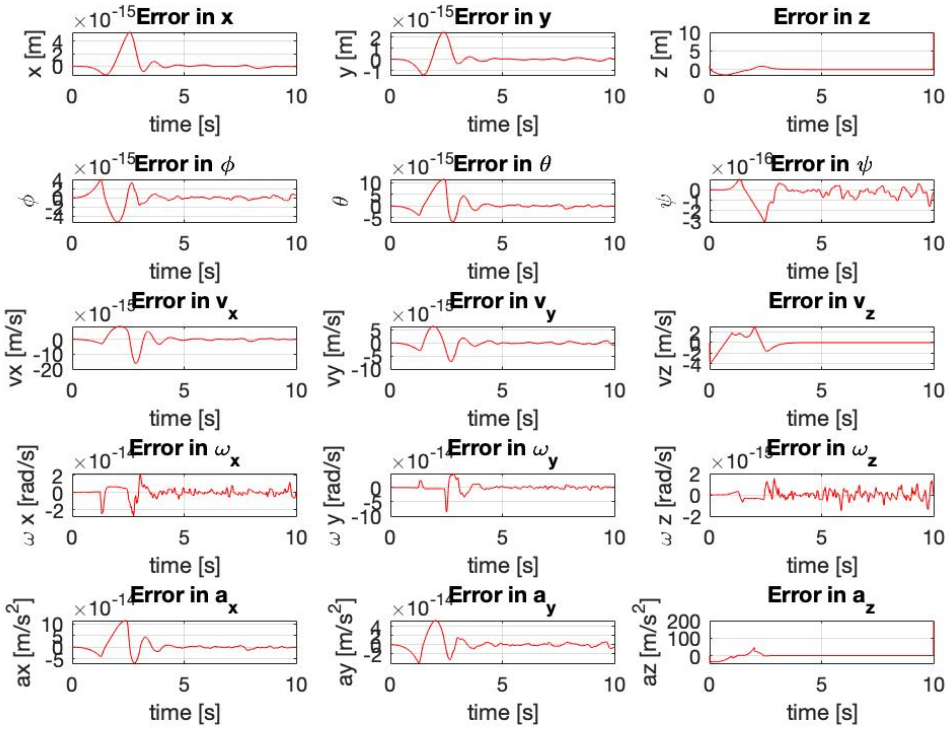


Figure 23: Error with time scaling of 8s

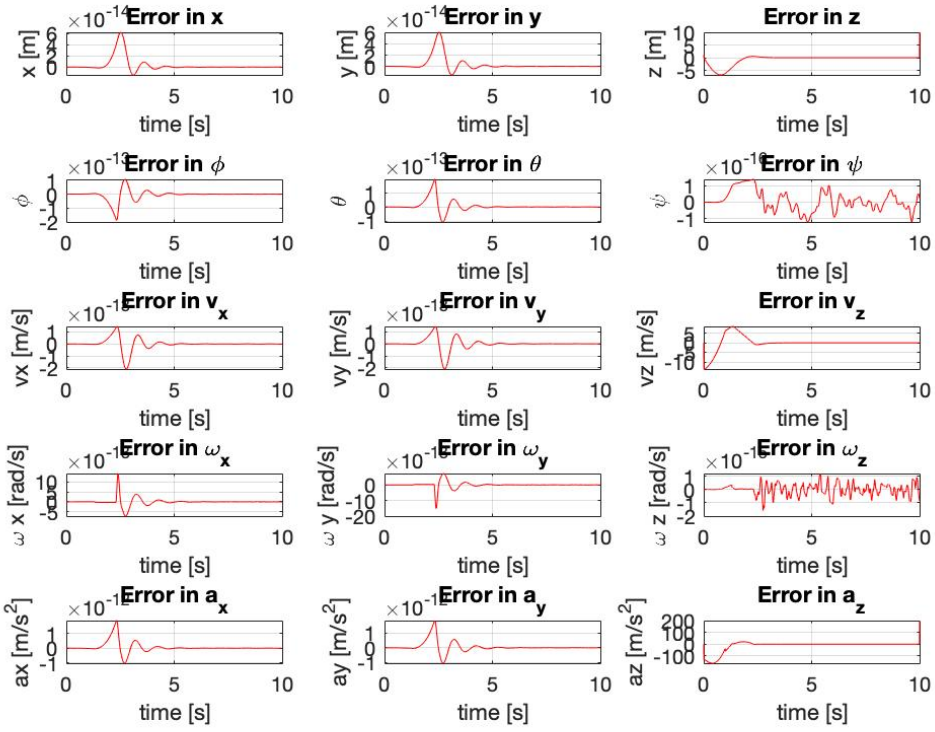
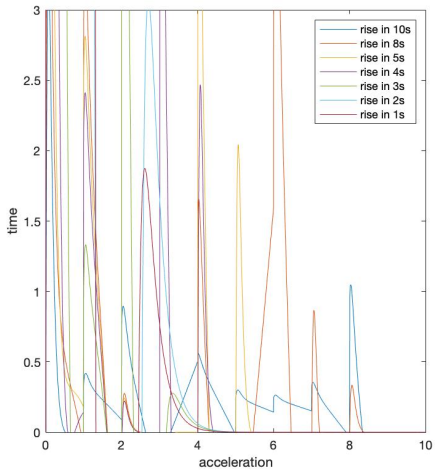
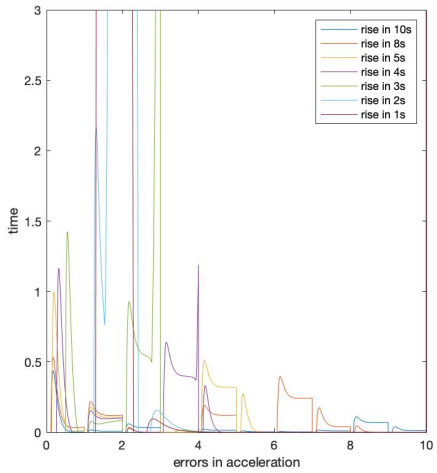


Figure 24: Error with time scaling of 10s

Then looking from the figure below:





We know, with time scaling of 10s, and way points as below:

$$waypoints = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1.1 & 1.4 & 2 & 2.6 & 3.5 & 4.6 & 5.9 & 7.4 & 9 & 10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$waypoint\_times = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9v10 \end{bmatrix}$$

we can limit the acceleration always smaller than 3m/s.

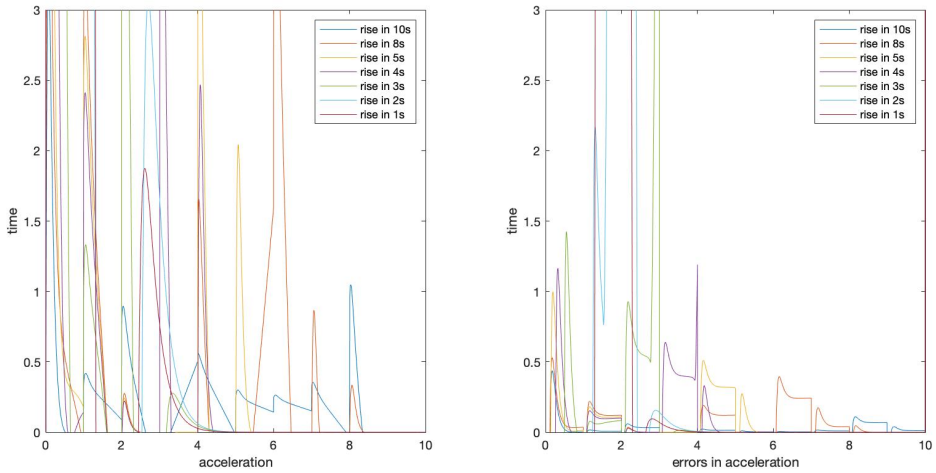


Figure 25: Error between the desired pose and the actual pose with 0 degree heading

Yes, the robot track the trajectory as expected, but there is substantial error arising from the source of max_rpm limit of motor.Increase the desired acceleration bound, we find that when the bound reaches 6.962 m/s, the system begin to exhibit degraded tracking accuracy.

This empirical observation can be explained by motor response. Each motor in physical world has a parameter "Maximum RPM" which limit the thrust it can provide. Thus, there will be a maximum acceleration limit. While we keep increase the desired acceleration bound, once it exceed the maximum acceleration limit of the motor, the system cannot track the trajectory with good performance, since the force that the motor provides is not enough to keep the system catching up the trajectory.

The performance improves if we artificially increase the motor gain configuration parameter thereby effectively upgrading the robot motors.

Q8

In this question, for clearer view, I will first give plots separately for phase 1 trajectory and phase 2 trajectory, then at last give the united plots that including phase 1,2 and 3 (The whole process includes 1 elliptical loop of acceleration, followed by 2 elliptical loops with constant speed, then 1 elliptical loop at last while the robot slows down).

The pose and error plots for phase one trajectory(Go along the elliptical with acceleration)

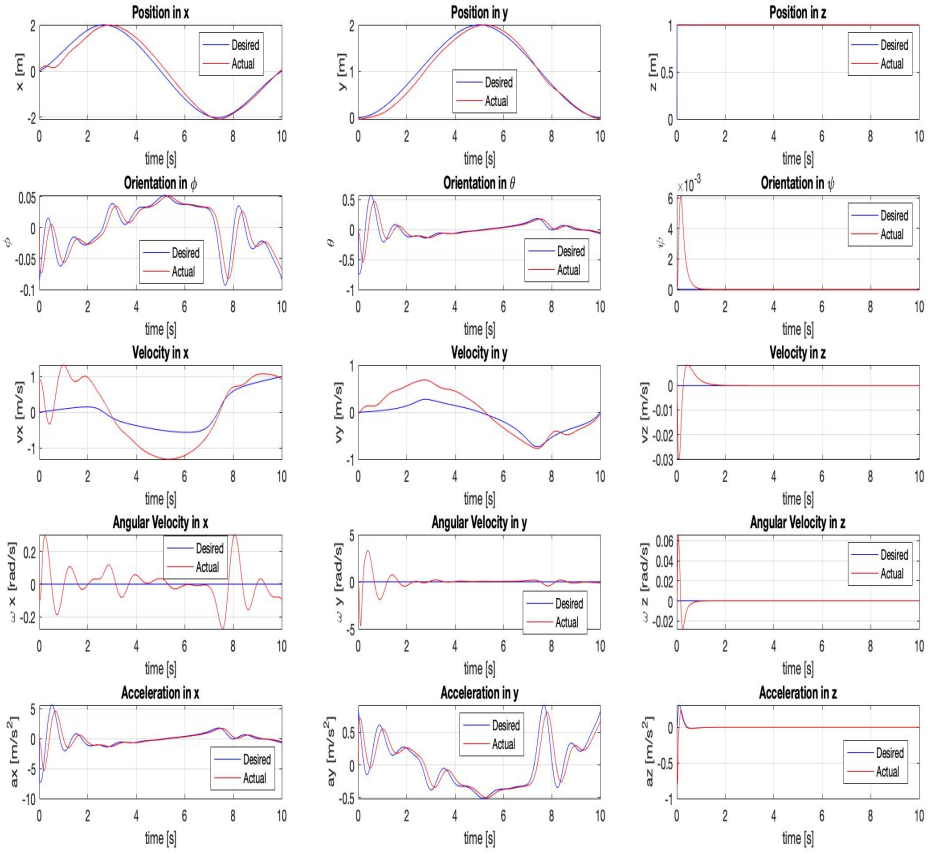


Figure 26: The desired pose and the actual pose for Elliptical Trajectory

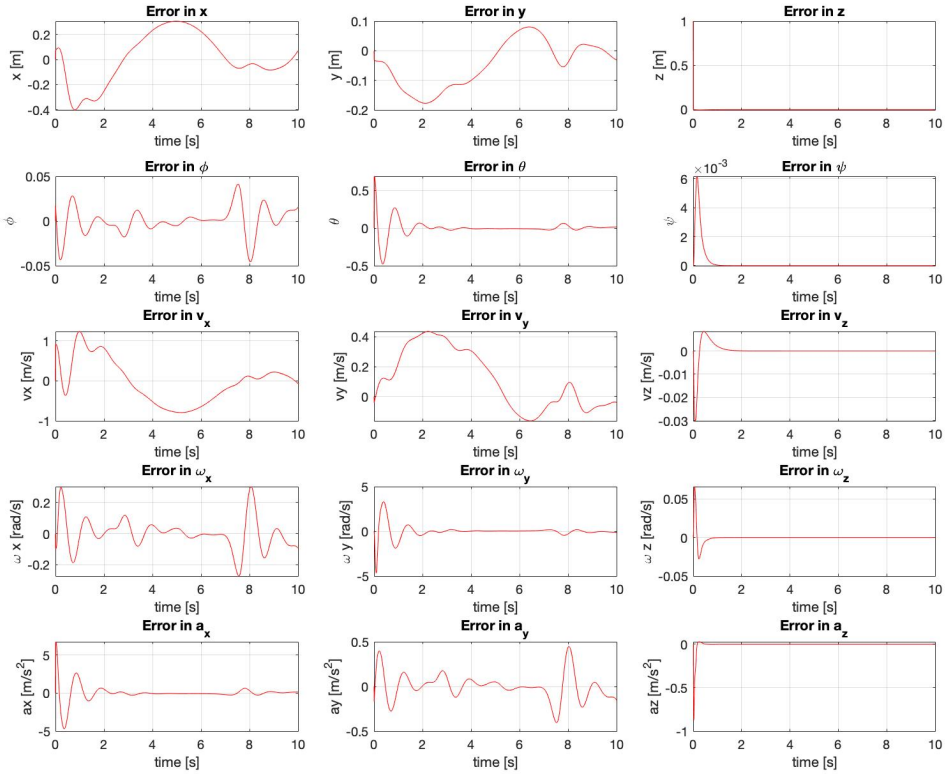


Figure 27: Error between the desired pose and the actual pose for Elliptical Trajectory

The pose and error plots for phase two trajectory(Go along the elliptical with constant speed 1m/s)

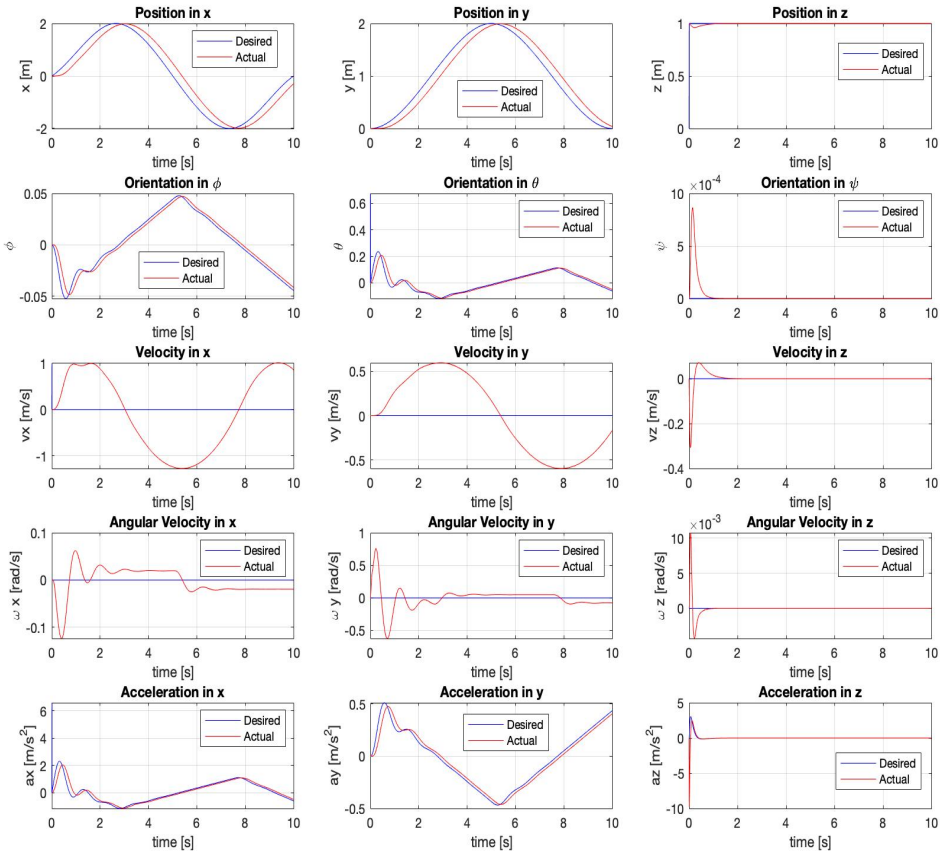


Figure 28: The desired pose and the actual pose for Elliptical Trajectory

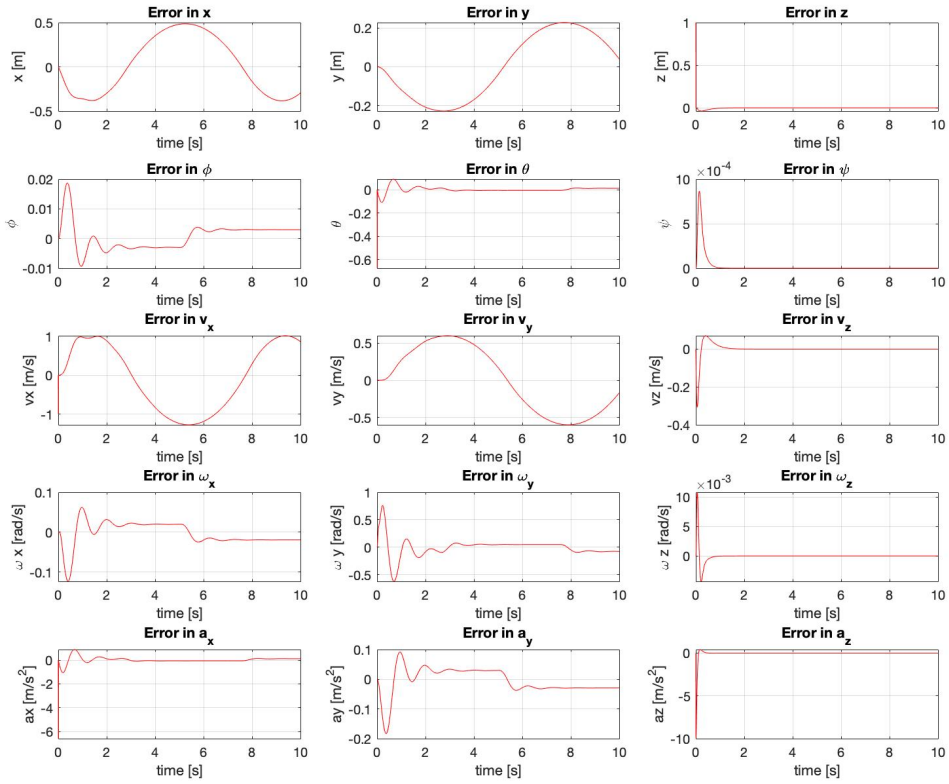


Figure 29: Error between the desired pose and the actual pose for Elliptical Trajectory

The pose and error plots for the whole process

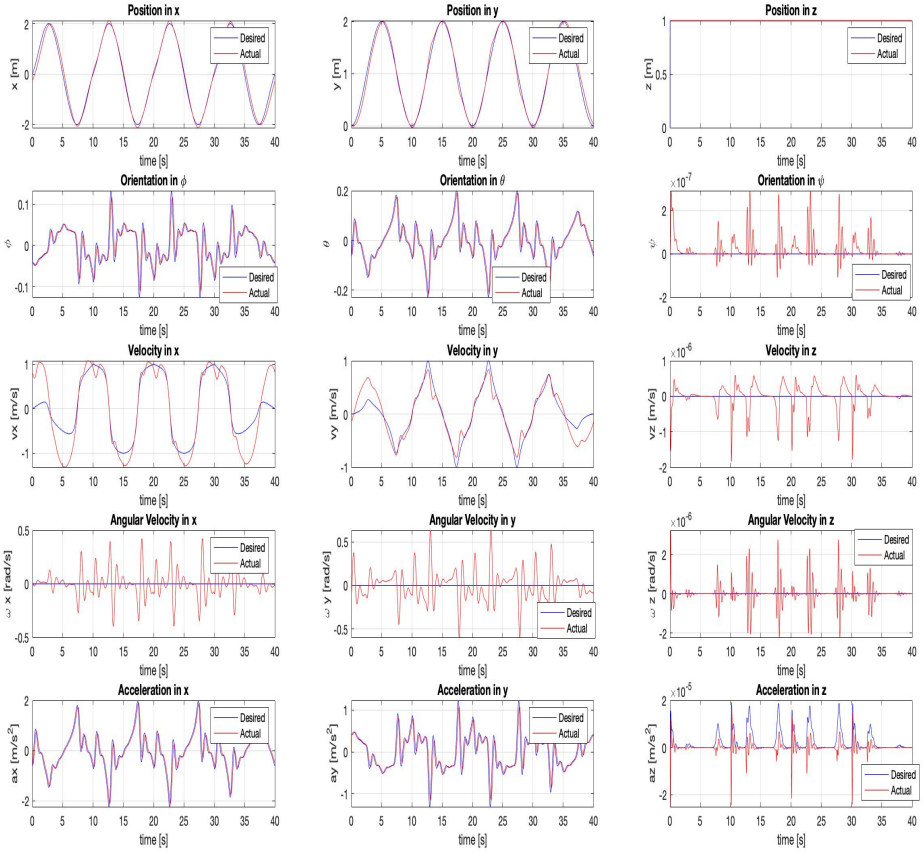


Figure 30: The desired pose and the actual pose for Elliptical Trajectory

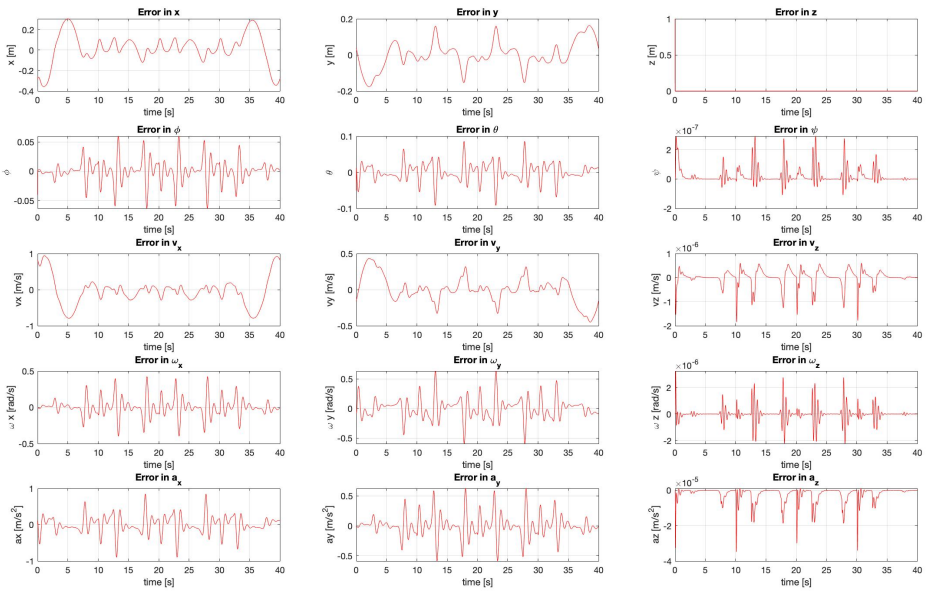


Figure 31: Error between the desired pose and the actual pose for Elliptical Trajectory

And here is the cumulative error distribution plot:

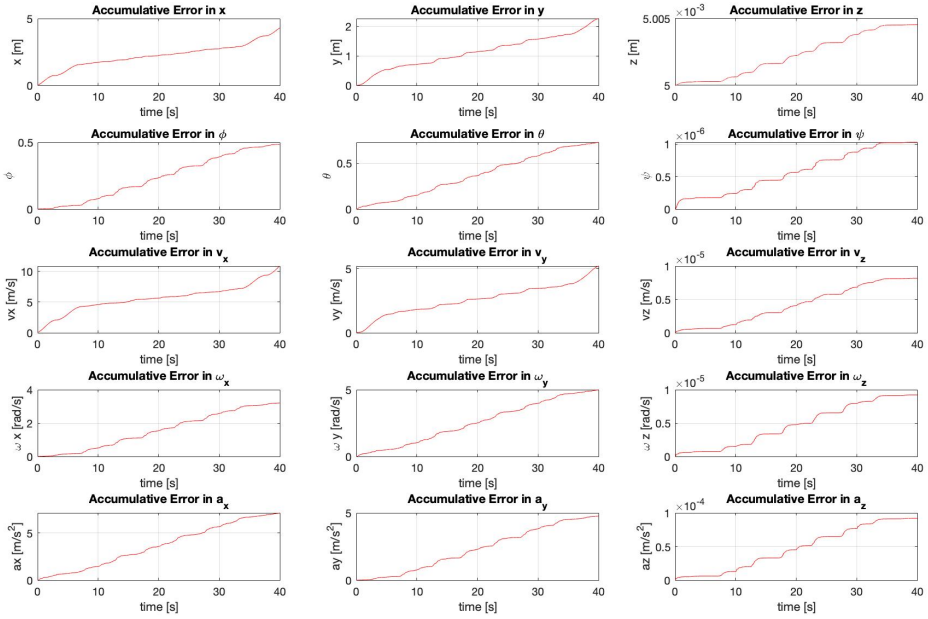


Figure 32: The cumulative error distribution

From the overall error plot and cumulative error distribution, we can find the tracking performance(error) are in the same pattern during the acceleration process and deceleration process, since they are like inverse process of each other. And it's apparent that the tracking error in x, y, ϕ,and θ during the process with constant velocity 1m/s is smaller than those during process with changing velocity, while the tracking error in z and ψ during the process with constant velocity 1m/s is larger than those during process with changing velocity.

Q9

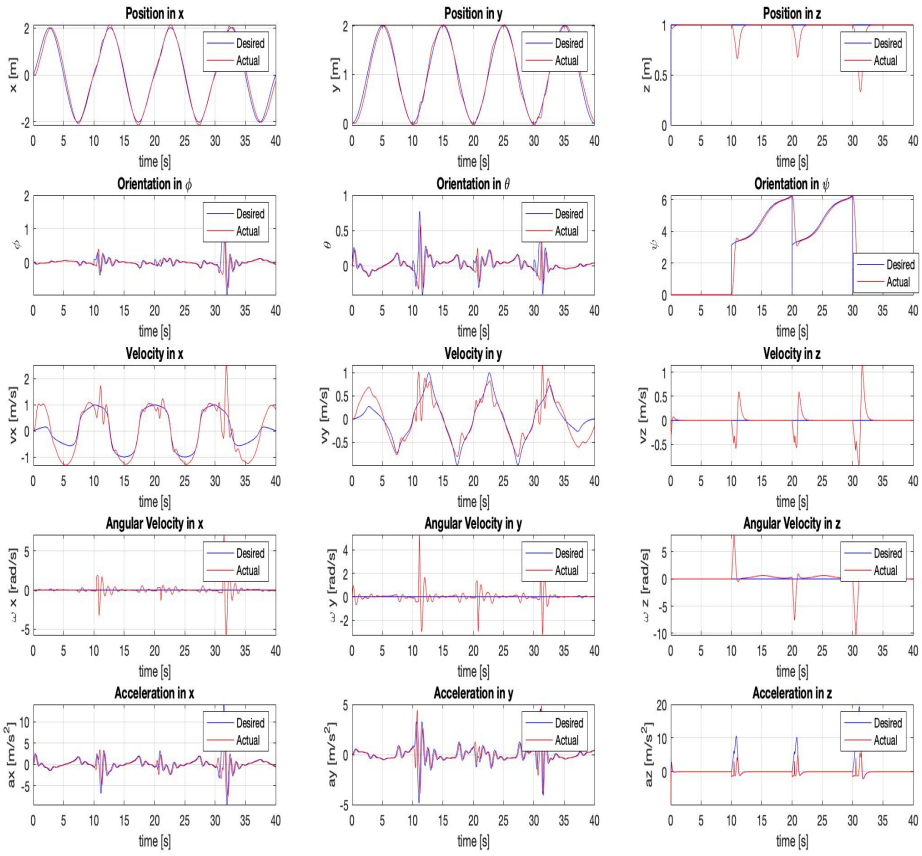


Figure 33: The desired pose and the actual pose for Elliptical Trajectory

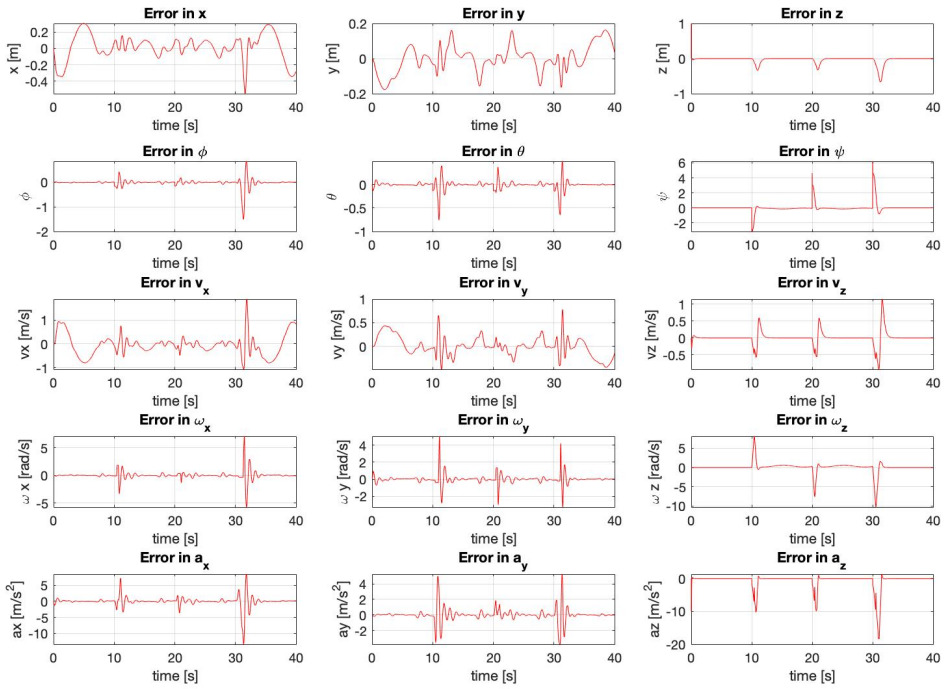


Figure 34: Error between the desired pose and the actual pose for Elliptical Trajectory

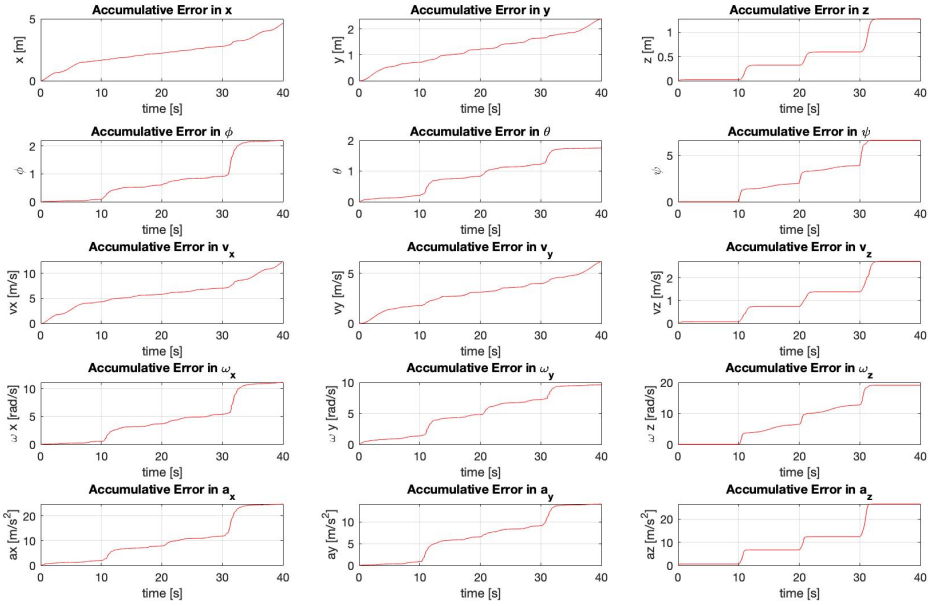


Figure 35: The cumulative error distribution

We can see from the figures that introducing time-varying heading commands at high-speeds, largely increases the oscillation and error in z direction, thus leading to worse tracking performance.