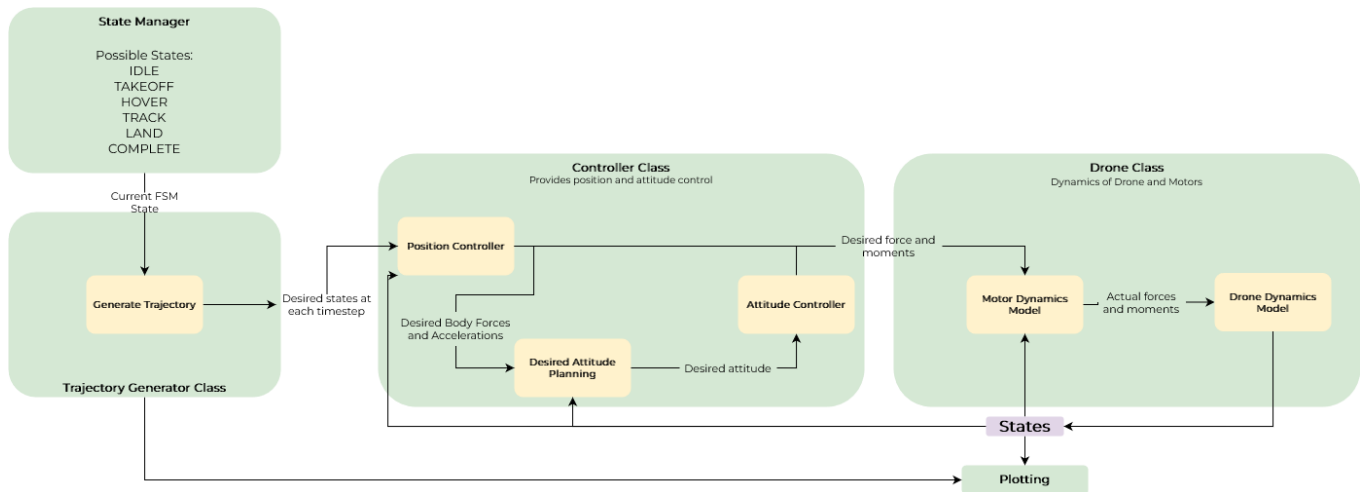# 16-665 Robot Mobility, Air Mobility Project

Praveen Venkatesh (pvenkat2)

Question 1

The system architecture is as follows:

Brief description of each block:

- State manager:

  - Has 6 possible states - IDLE, TAKEOFF, HOVER, TRACK, LAND, COMPLETE
  - The statemanager handles everything that is to do with the state of the finite state machine. It also handles whenever a given FSM state is complete and goes to the next state using an error based threshold on the desired trajectory location.

- Trajectory Generator class:

  - The trajectory generator generates the trajectory for each part of the flight phase and defines the state vector of the drone at each timestep in the duration of flight of the drone.

- Controller Class:

  - The controller handles both the position and attitude control laws of the drone.

- Drone Class:

- This class encapsulates the dynamics of the motors and the drone itself. The state of the drone is also saved in this class.

Some notes:

- The entire codebase has been significantly altered and organized into classes and separate files as to make the project easy to execute and develop.
- The execution method is retained so that the TA's can execute the code in the same manner as was provided
  - The execution method is `python main_q.py <question_number>`
- When running the code, the written program automatically generates all plots and saves the flight parameters to the `outputs/` directory that will be created in the root folder for ease of access.
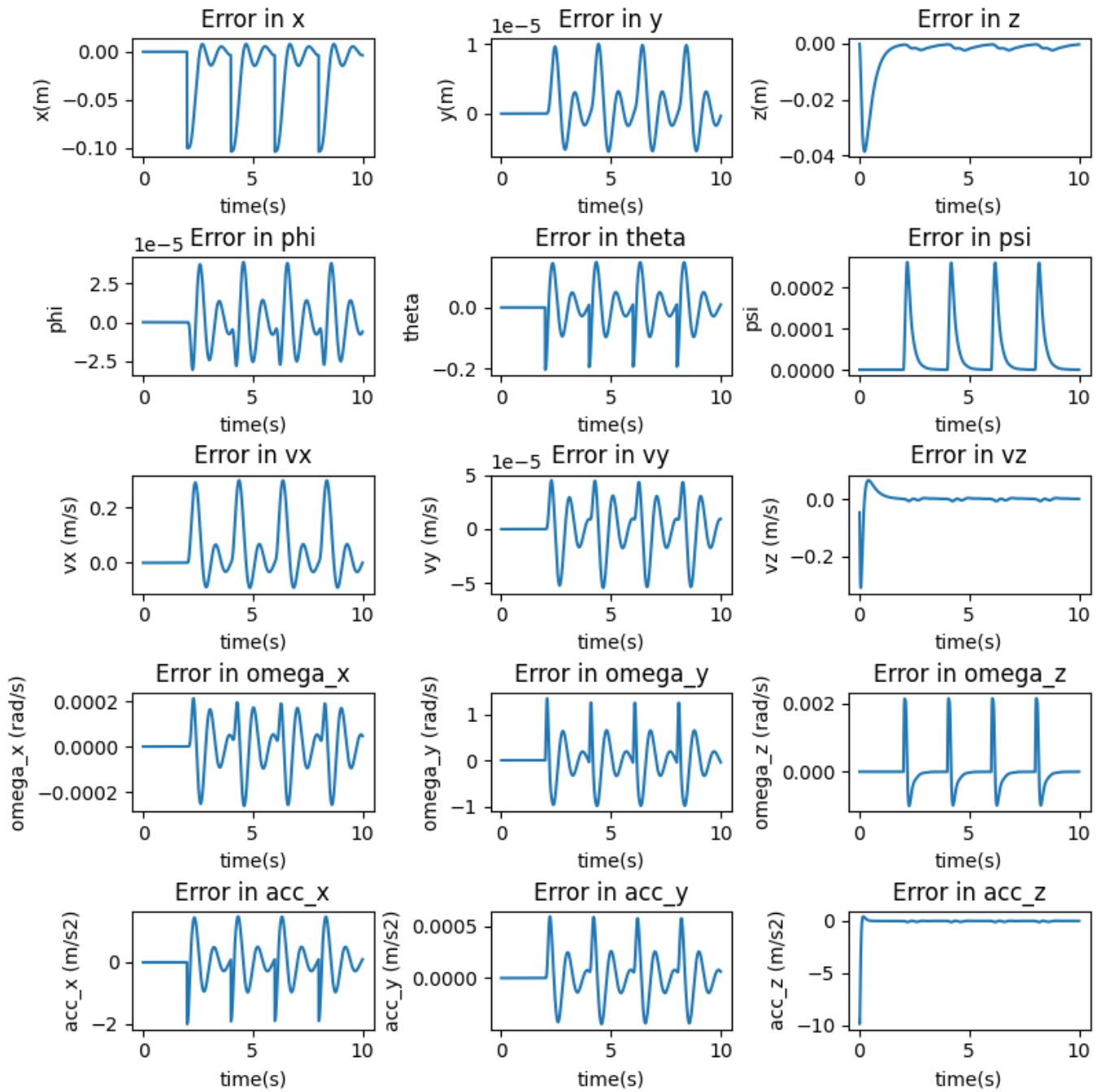
Question 2

```
Gains:
kXatt = [Kp_phi, Kp_theta, Kp_psi]
kXpos = [Kp_x, Kp_y, Kp_z]
"kpatt": [  190,    190,    70],
"kdatt": [   30,     30,    18],
"kppos": [   20,     20,    18],
"kdpos": [    8,      8,     9],
```

The actual poses of the drone are as follows:
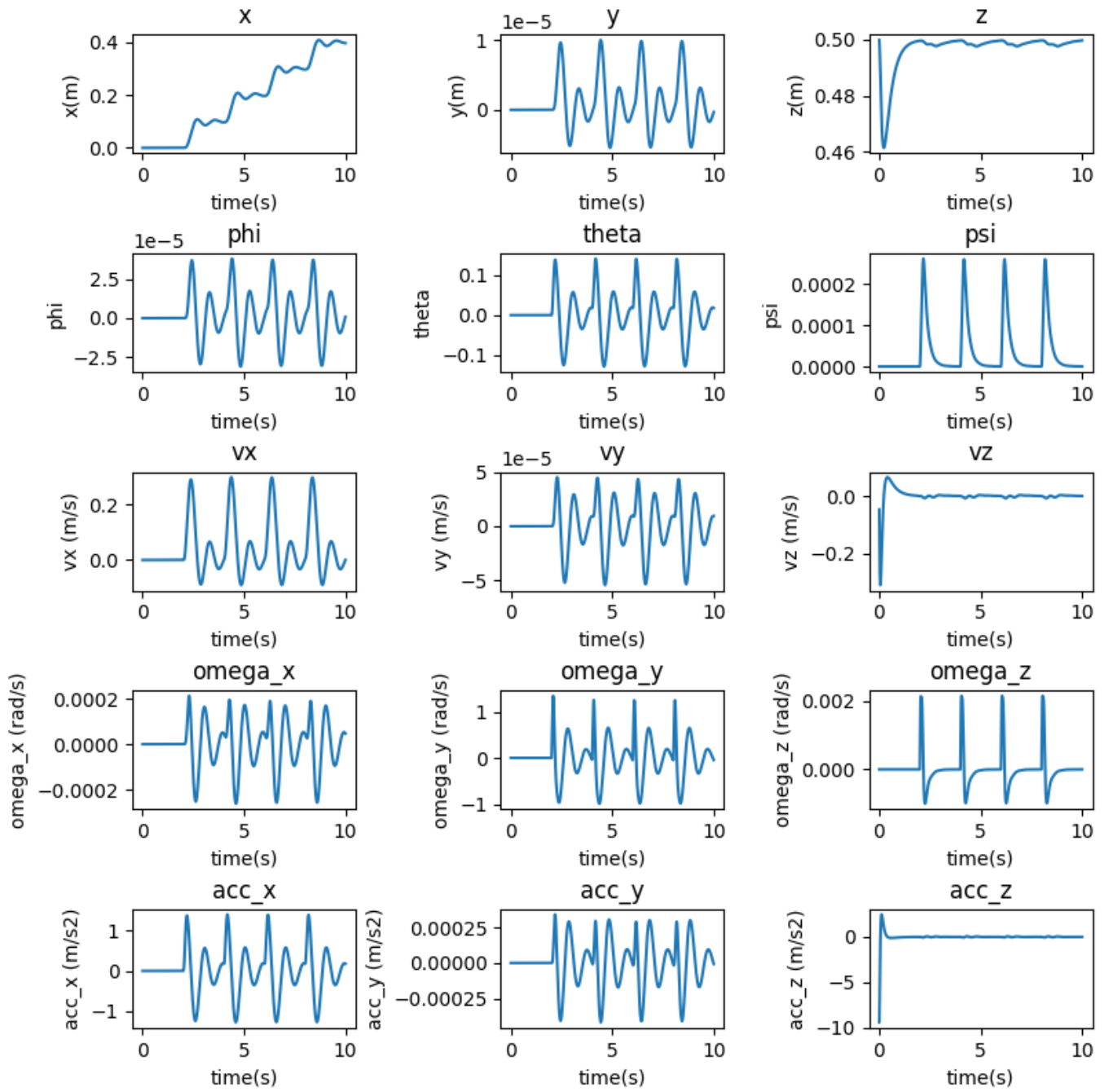
The error plots for the requested scenario are as follows:

- We can see that the drone converges to each waypoint sufficiently before the next waypoint is fed into the system. The new waypoints are fed into the system at $t = 2s, 4s, 6s, 8s$.

- The system oscillates about the given waypoints since we are using a PD controller and damping is not sufficient to completely remove all oscillations (underdamped system).

Ensuring that the performance is physically plausible using the thrust to weight plot for this scenario:

We can see that the performance is well within physical limits.

The trajectory in 3D can be seen like this:

Position
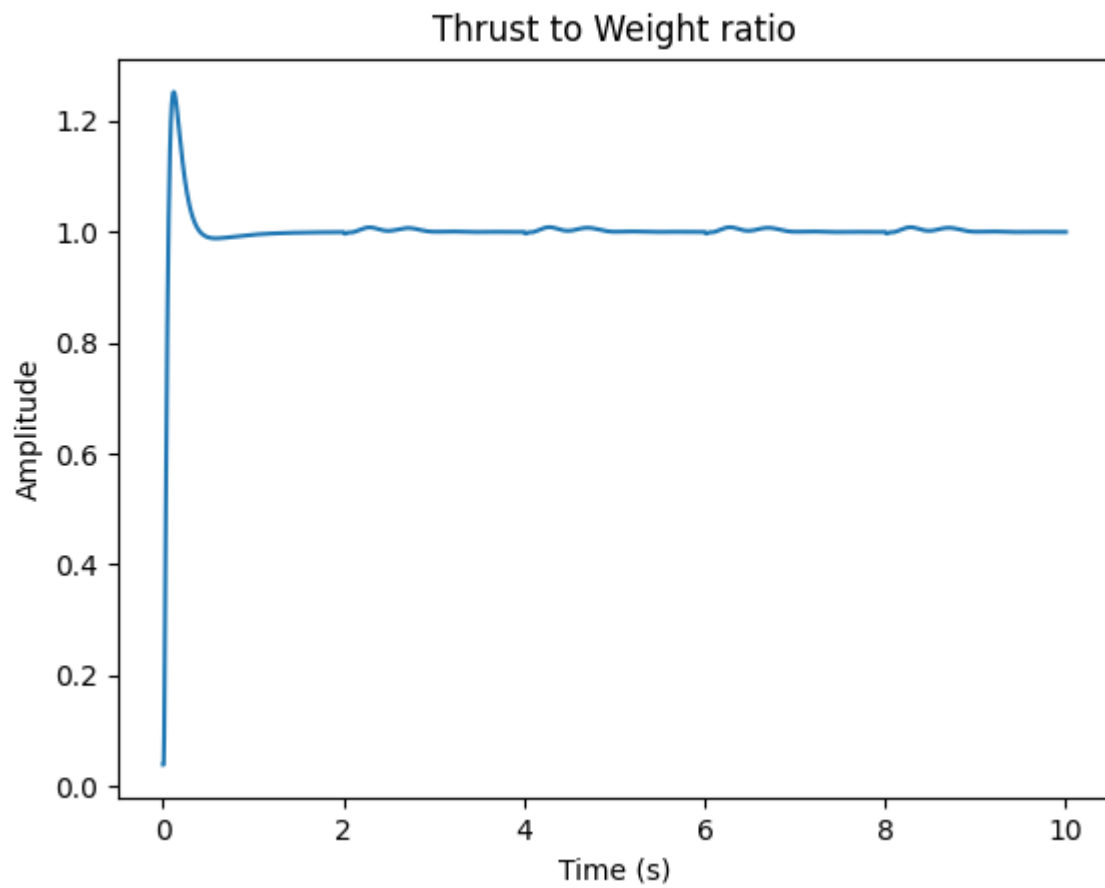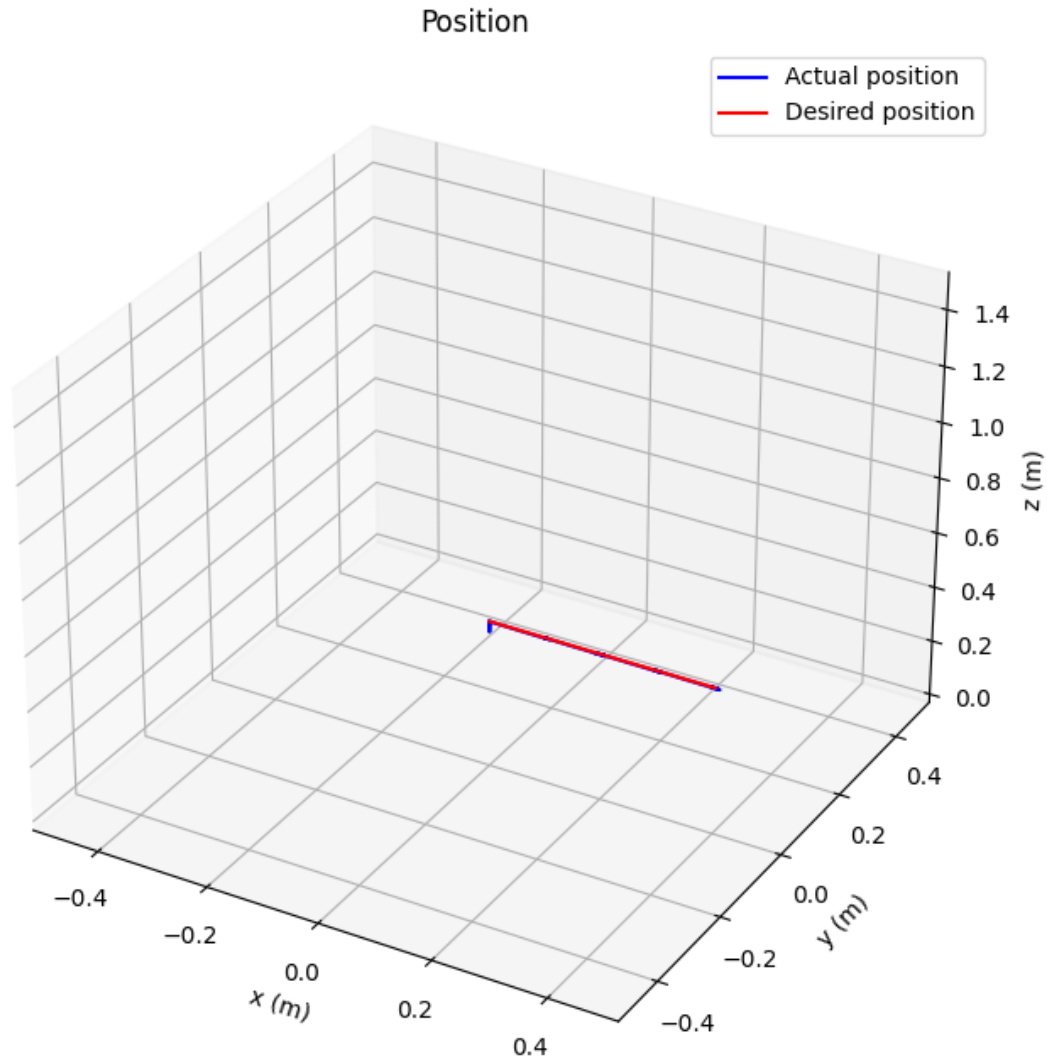
Disclaimer: I have changed the initial state of the drone such that it starts at hover. Previously the drone was given the starting location of z = 0 which causes a sudden jerk in the desired trajectory which I have removed.

Variational study on PD Controller gains

Modifying Kd_positions (Outer Loop)

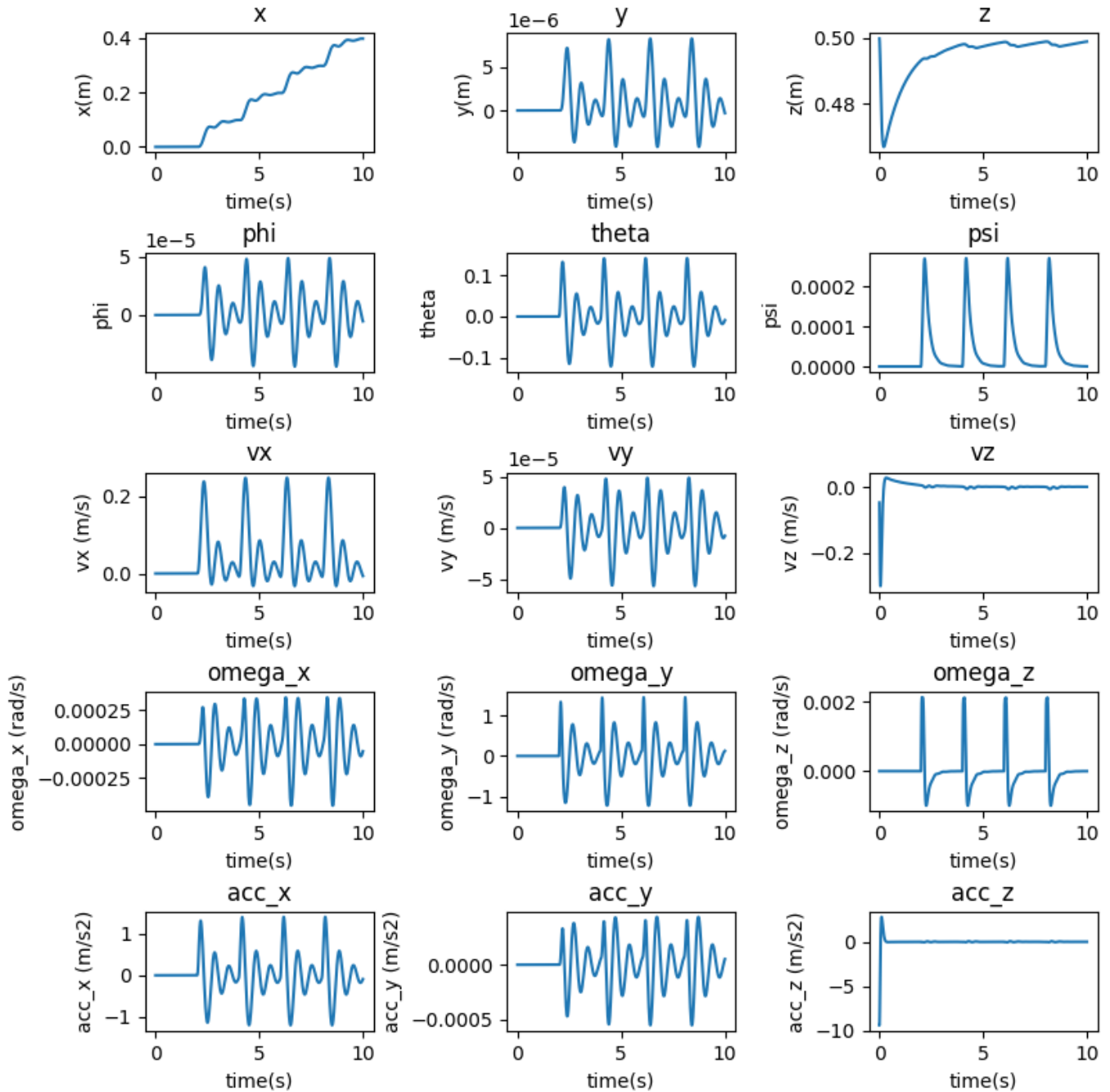Here the Kd of the outer loop has been increased.

```
Gains:
kXatt = [Kp_phi, Kp_theta, Kp_psi]
```
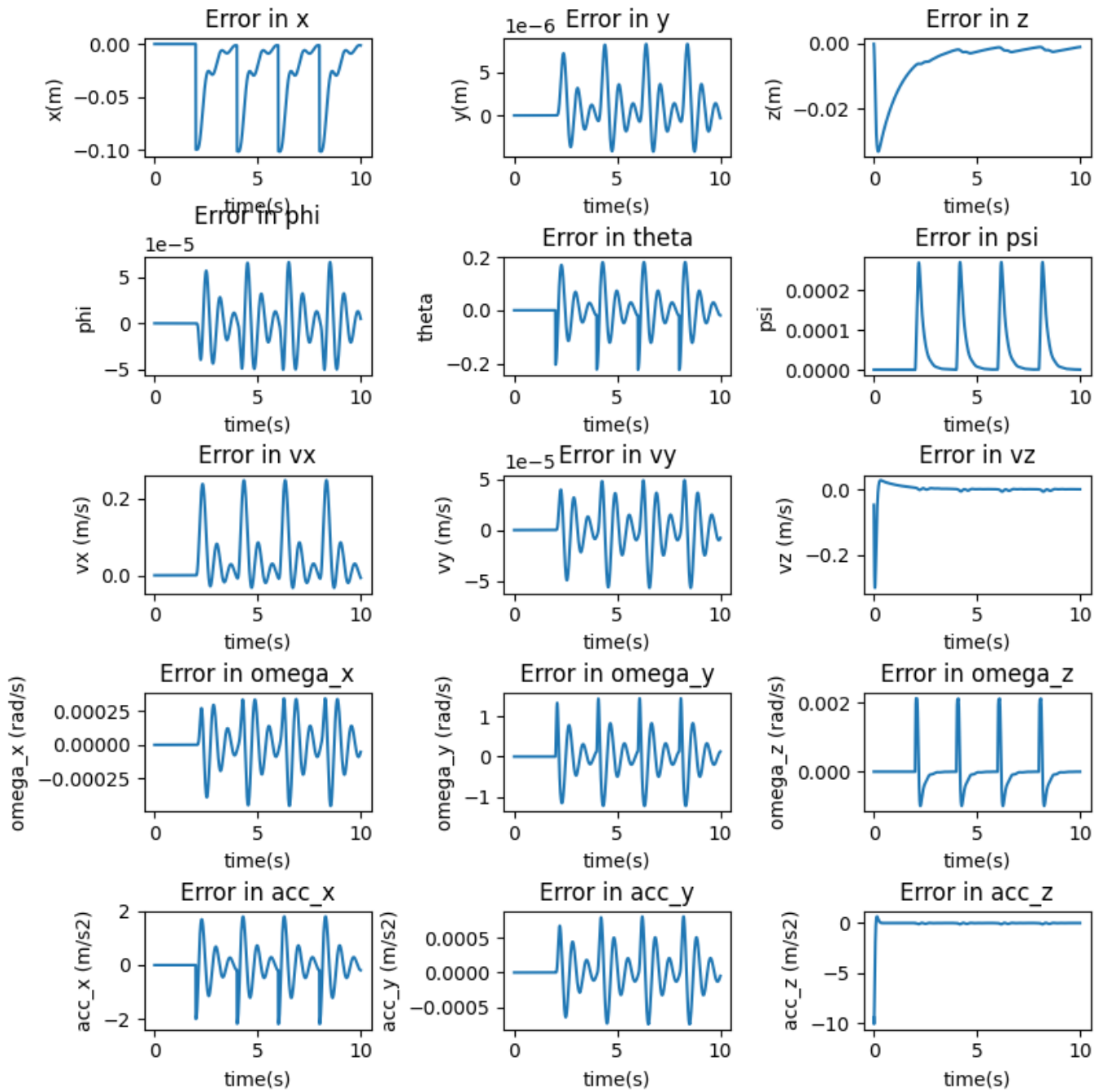
```
kXpos = [Kp_x, Kp_y, Kp_z]
params['kpatt'] = [190, 190, 70]
params['kdatt'] = [30, 30, 18]
params['kppos'] = [20, 20, 10]
params['kdpos'] = [12, 12, 12]
```

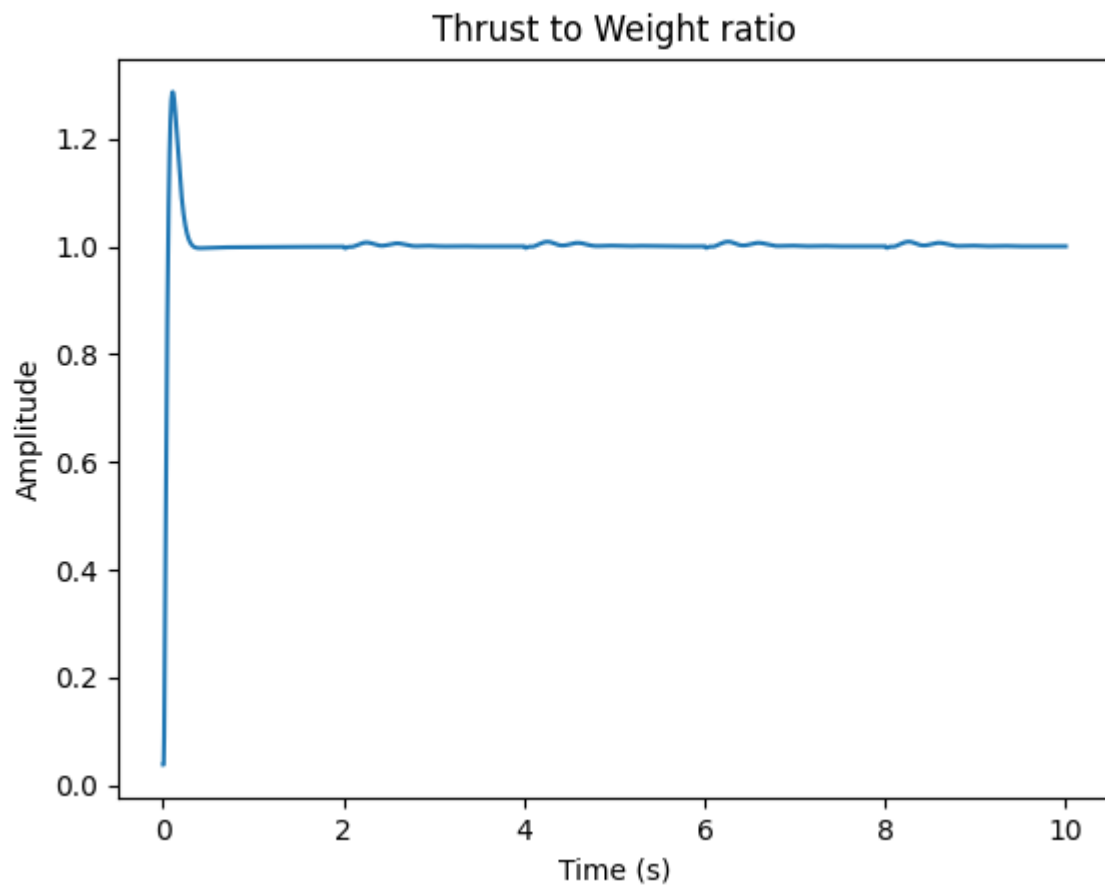The actual poses of the drone are as follows:

The error plots for the requested scenario are as follows:

- We can see that the overshoot has been significantly reduced and the drone reaches the desired x-location slowly compared to the previous case.
- We can also notice that the convergence time for z coordinate is slightly worse as the damping has been increased.

Ensuring that the performance is physically plausible using the thrust to weight plot for this scenario:

We can see that the performance is well within physical limits.

Modifying Kp_positions (Outer Loop)

Here, the Kp of the outer loop has been increased.

```
Gains:
kXatt = [Kp_phi, Kp_theta, Kp_psi]
kXpos = [Kp_x, Kp_y, Kp_z]
params['kpatt'] = [190, 190, 20]
params['kdatt'] = [30, 30, 18]
params['kppos'] = [25, 25, 10]
params['kdpos'] = [8, 8, 9]
```
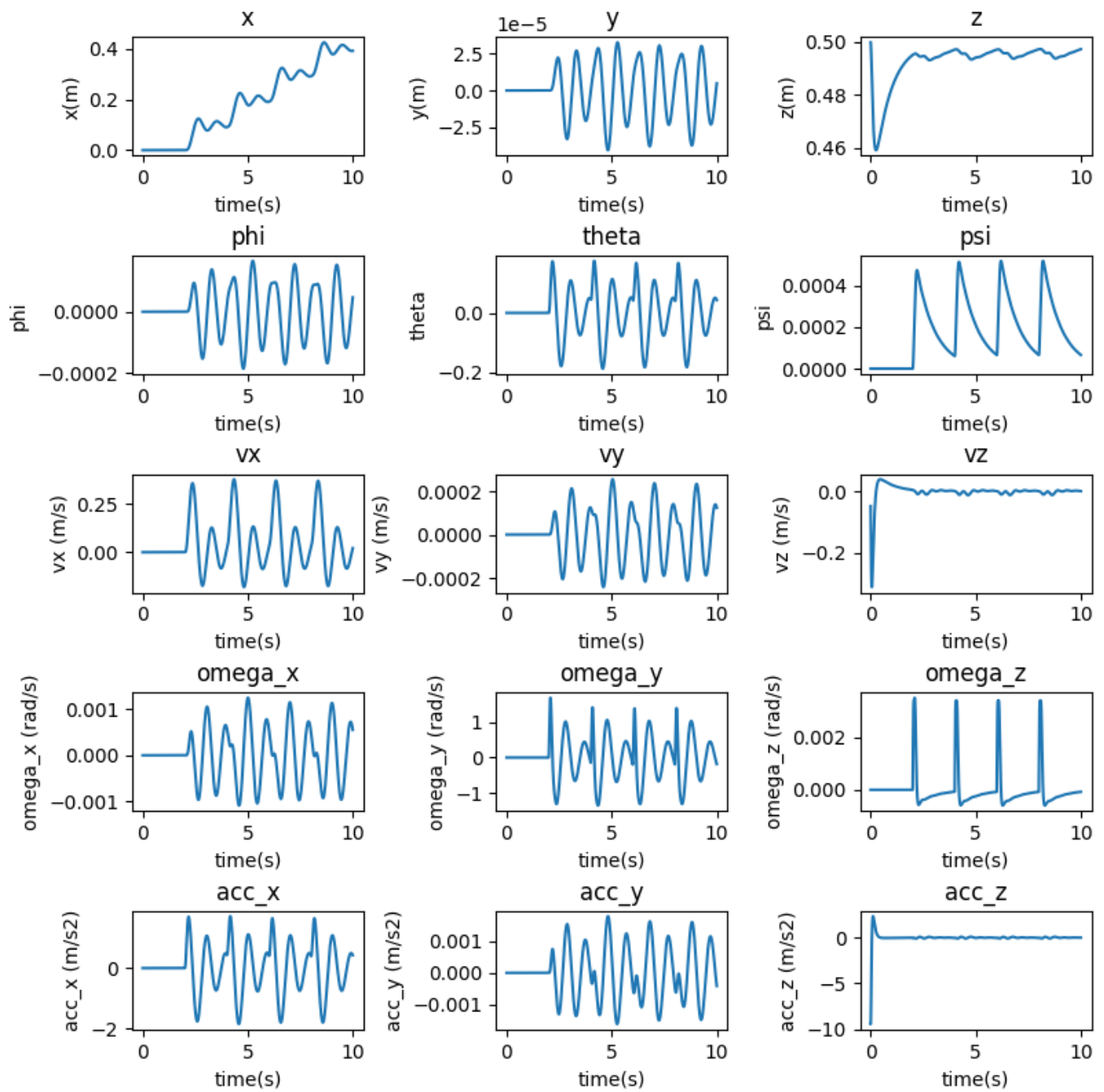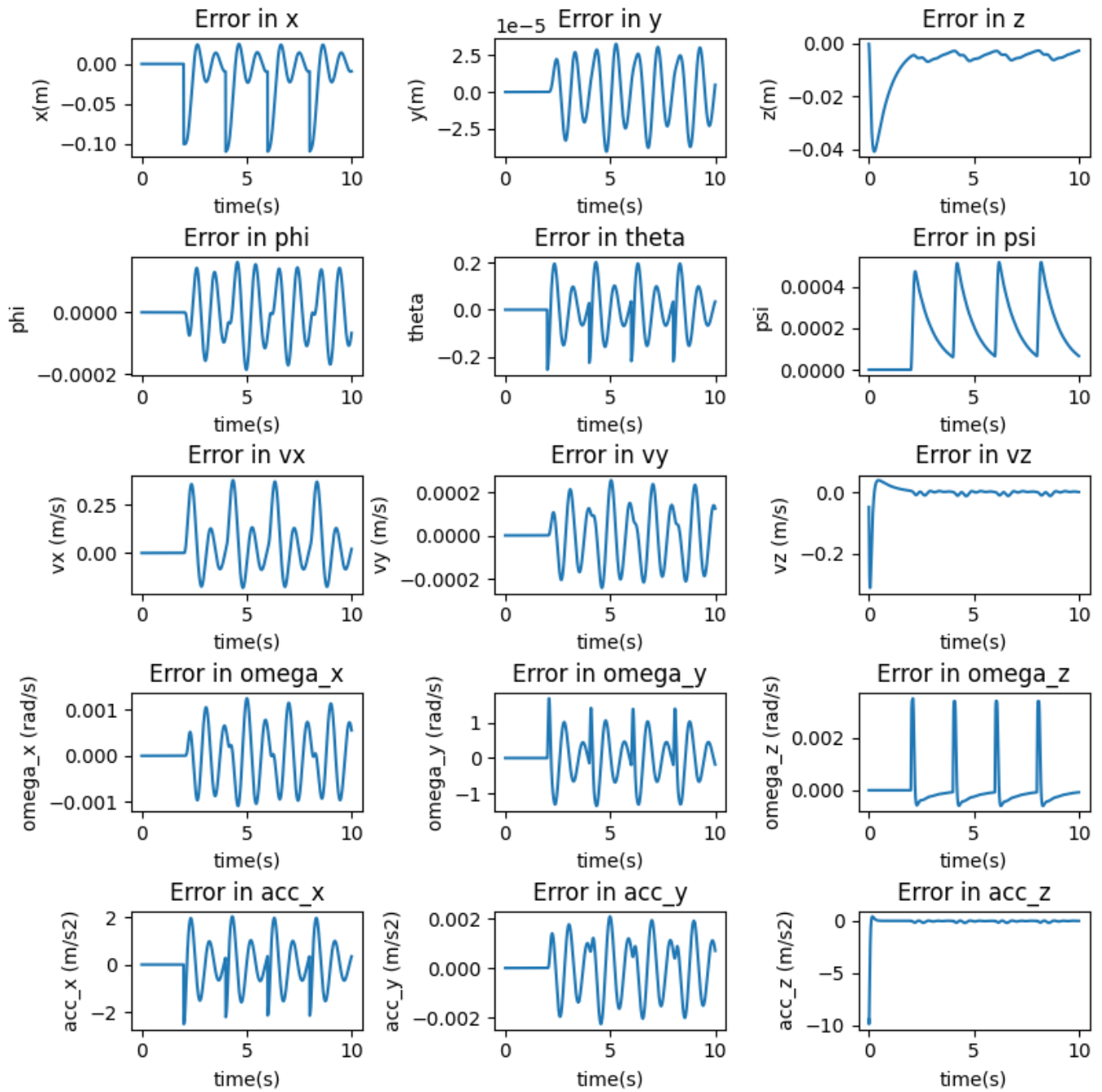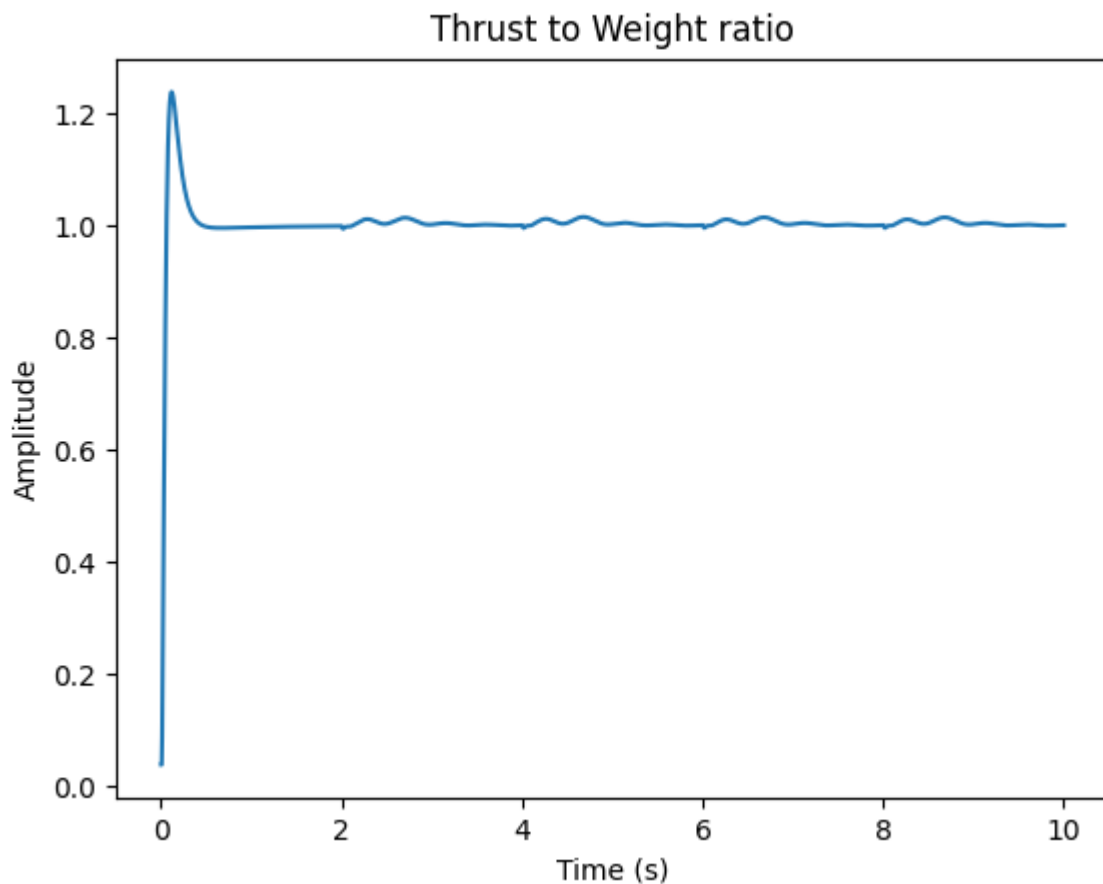
The actual poses of the drone are as follows:

The error plots for the requested scenario are as follows:

- We can see that the overshoot has been increased and the drone reaches the desired x-location much quicker than previously. This is because the proportional gain has been increased. This behaviour is as expected.

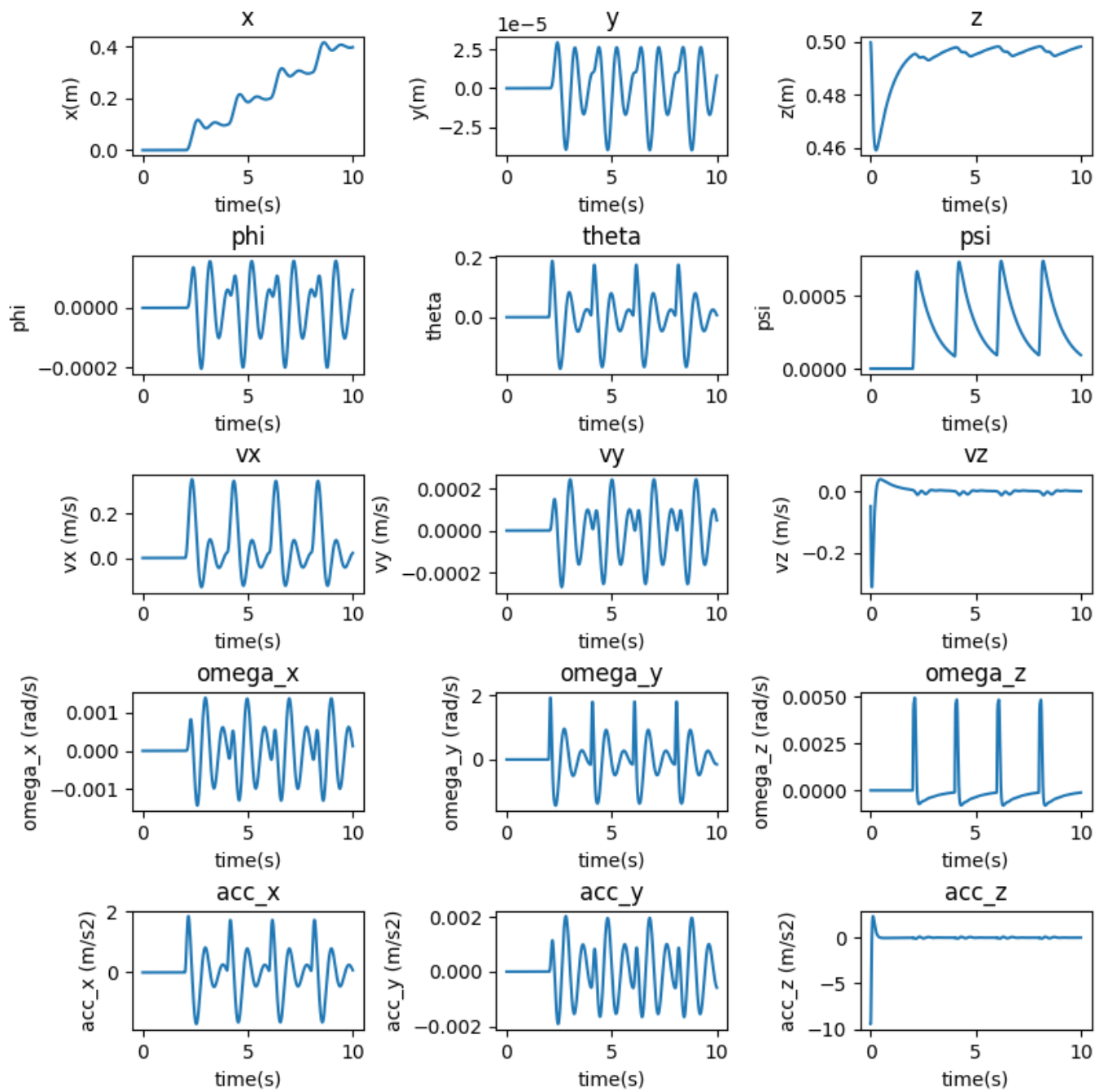Ensuring that the performance is physically plausible using the thrust to weight plot for this scenario:

Thrust to Weight ratio

We can see that the performance is well within physical limits.
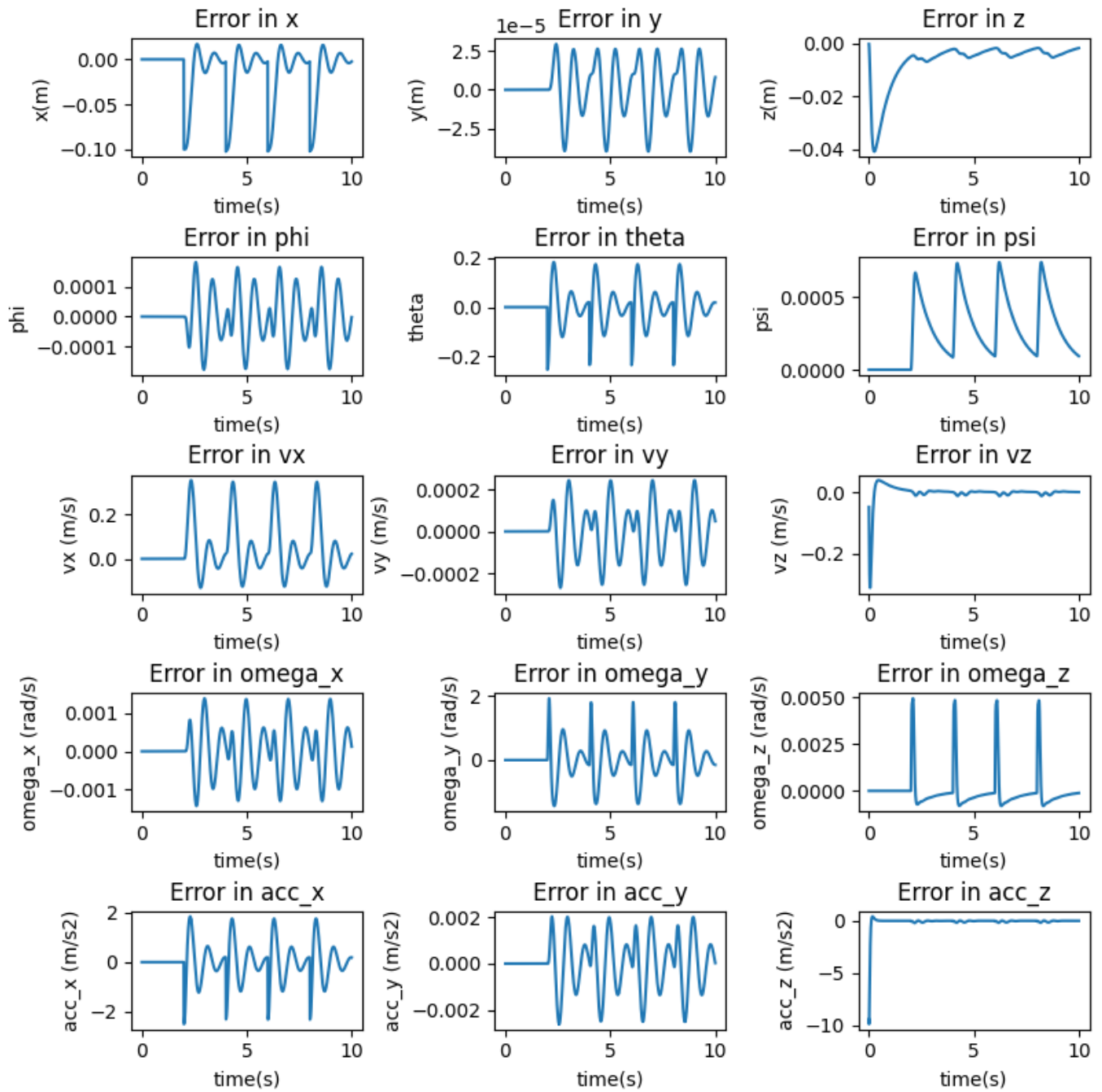
Modifying Kp_attitude (Inner Loop)

Here the Kp of the attitude controller has been increased

```
Gains:
kXatt = [Kp_phi, Kp_theta, Kp_psi]
kXpos = [Kp_x, Kp_y, Kp_z]
params['kpatt'] = [220, 220, 20]
params['kdatt'] = [30, 30, 18]
params['kppos'] = [25, 25, 10]
params['kdpos'] = [8, 8, 9]
```
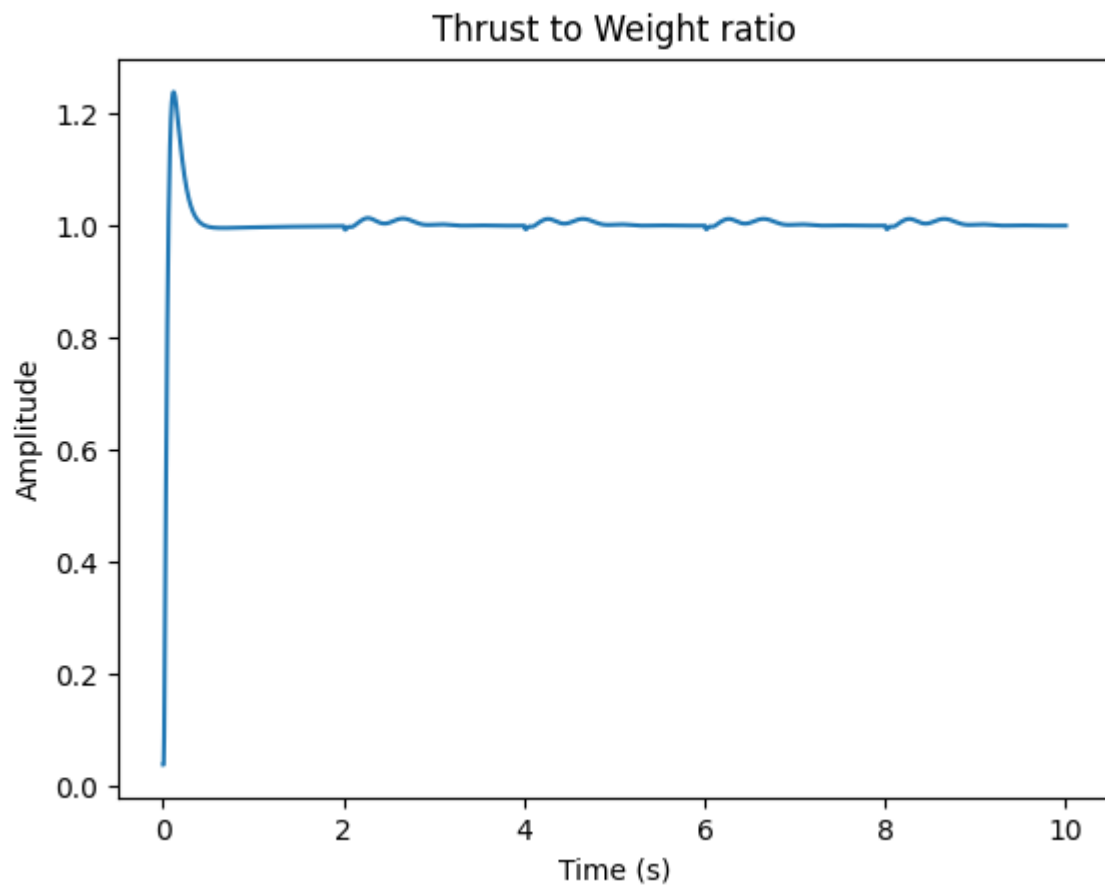
The actual poses of the drone are as follows:

The error plots for the requested scenario are as follows:

- We can see that the overshoot in the attitude has been increased significantly. However, we also notice that the drone reaches its desired attitude in a quicker fashion.

Ensuring that the performance is physically plausible using the thrust to weight plot for this scenario:

Thrust to Weight ratio

We can see that the performance is well within physical limits.

Modifying Kd_attitude (Inner Loop)

Here the Kd of the attitude controller has been increased

```
Gains:
kXatt = [Kp_phi, Kp_theta, Kp_psi]
kXpos = [Kp_x, Kp_y, Kp_z]
params['kpatt'] = [190, 190, 20]
params['kdatt'] = [35, 35, 20]
params['kppos'] = [25, 25, 10]
params['kdpos'] = [8, 8, 9]
```
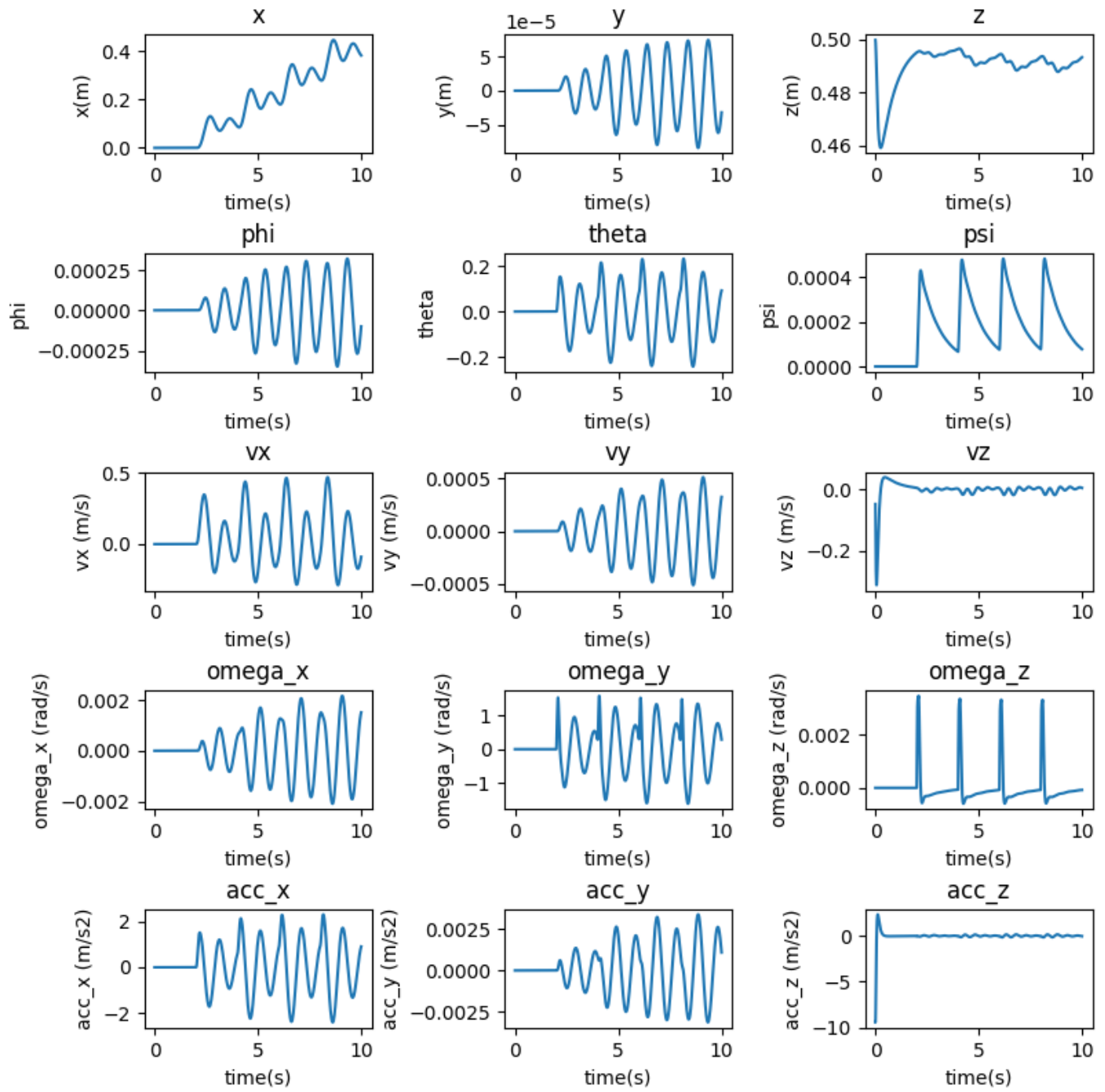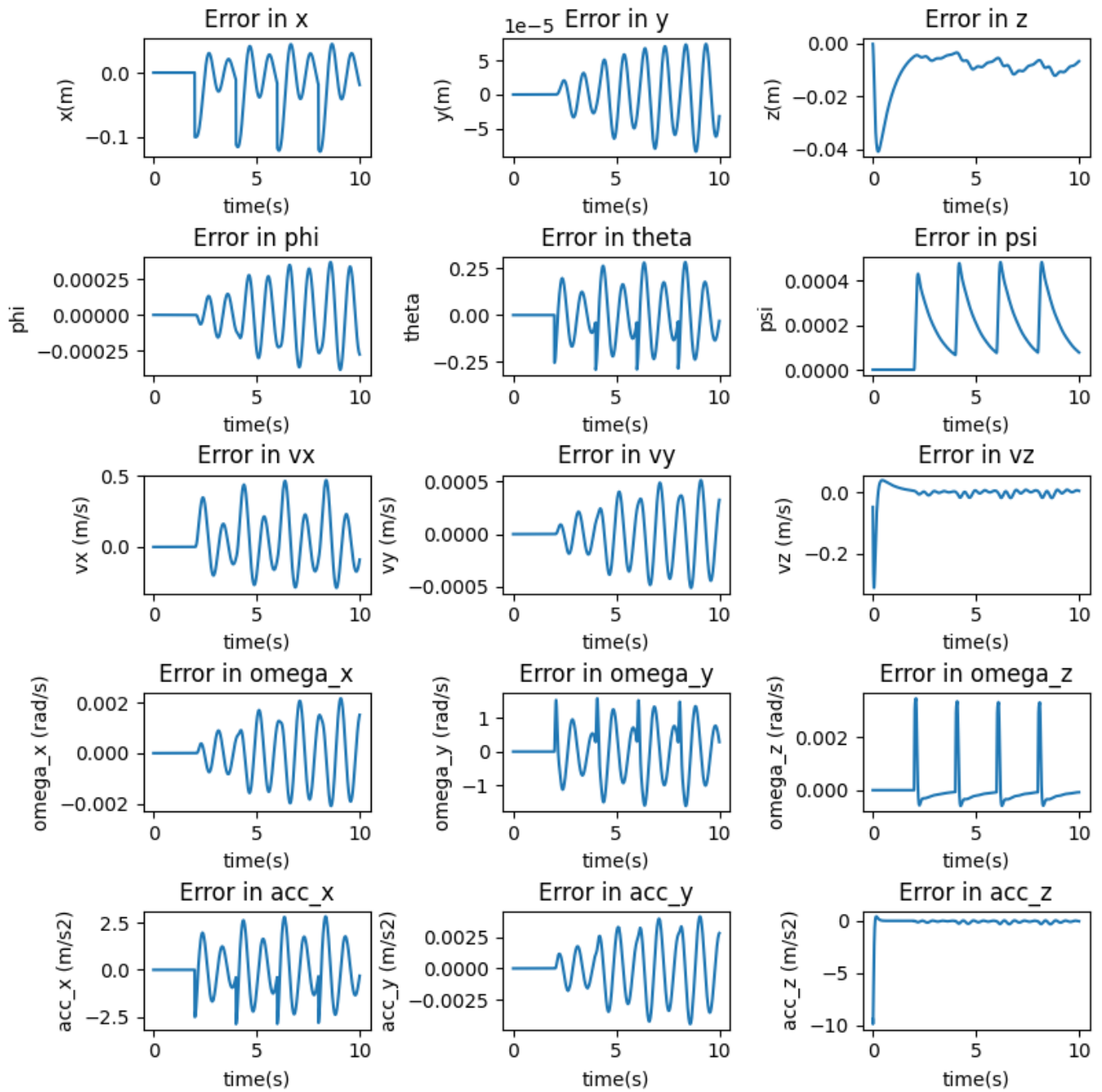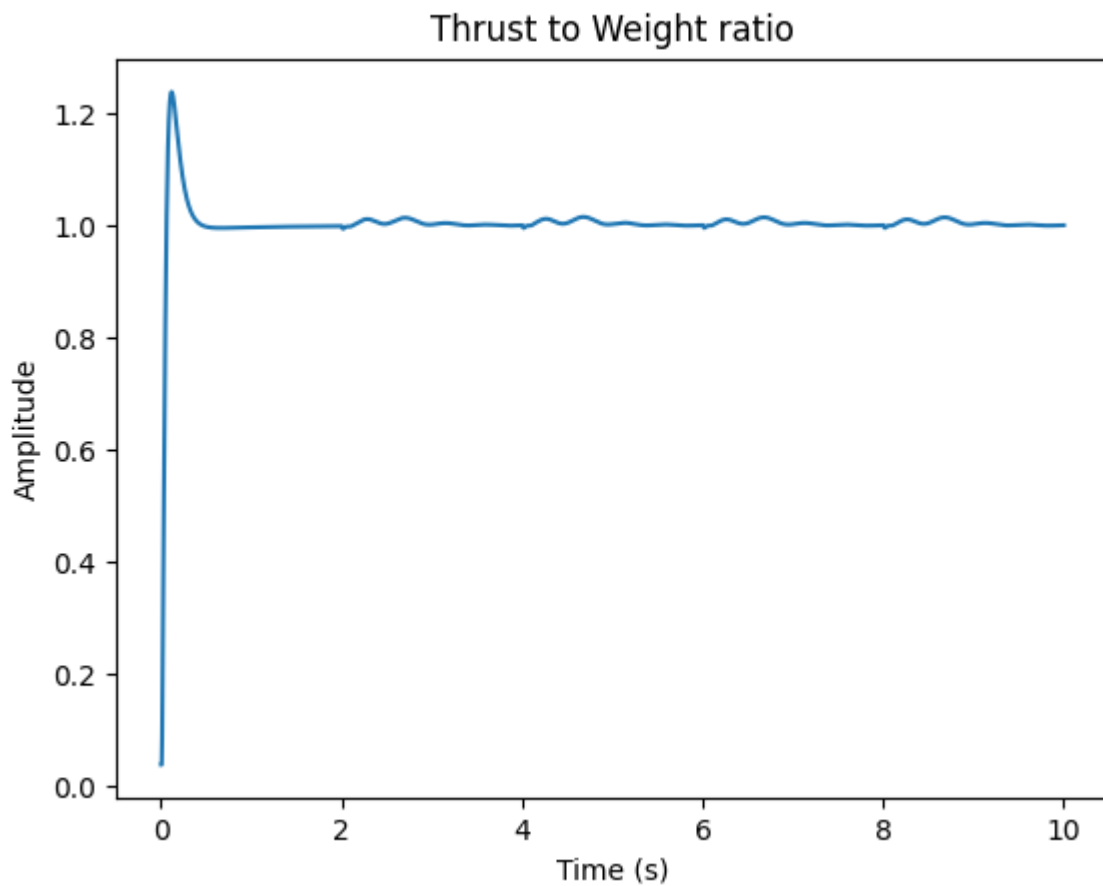
The actual poses of the drone are as follows:

The error plots for the requested scenario are as follows:

- We can see that the overshoot in the attitude has been decreased, and the drone is smoother in its response. It takes more time for the drone to reach its desired attitude.

Ensuring that the performance is physically plausible using the thrust to weight plot for this scenario:

Thrust to Weight ratio

We can see that the performance is well within physical limits.

Question 3

```
Gains:
KXatt = [Kp_phi, Kp_theta, Kp_psi]
KXpos = [Kp_x, Kp_y, Kp_z]
"kpatt": [  190,    190,    70],
"kdatt": [   30,     30,    18],
"kppos": [   20,     20,    18],
"kdpos": [    8,      8,     9],
```
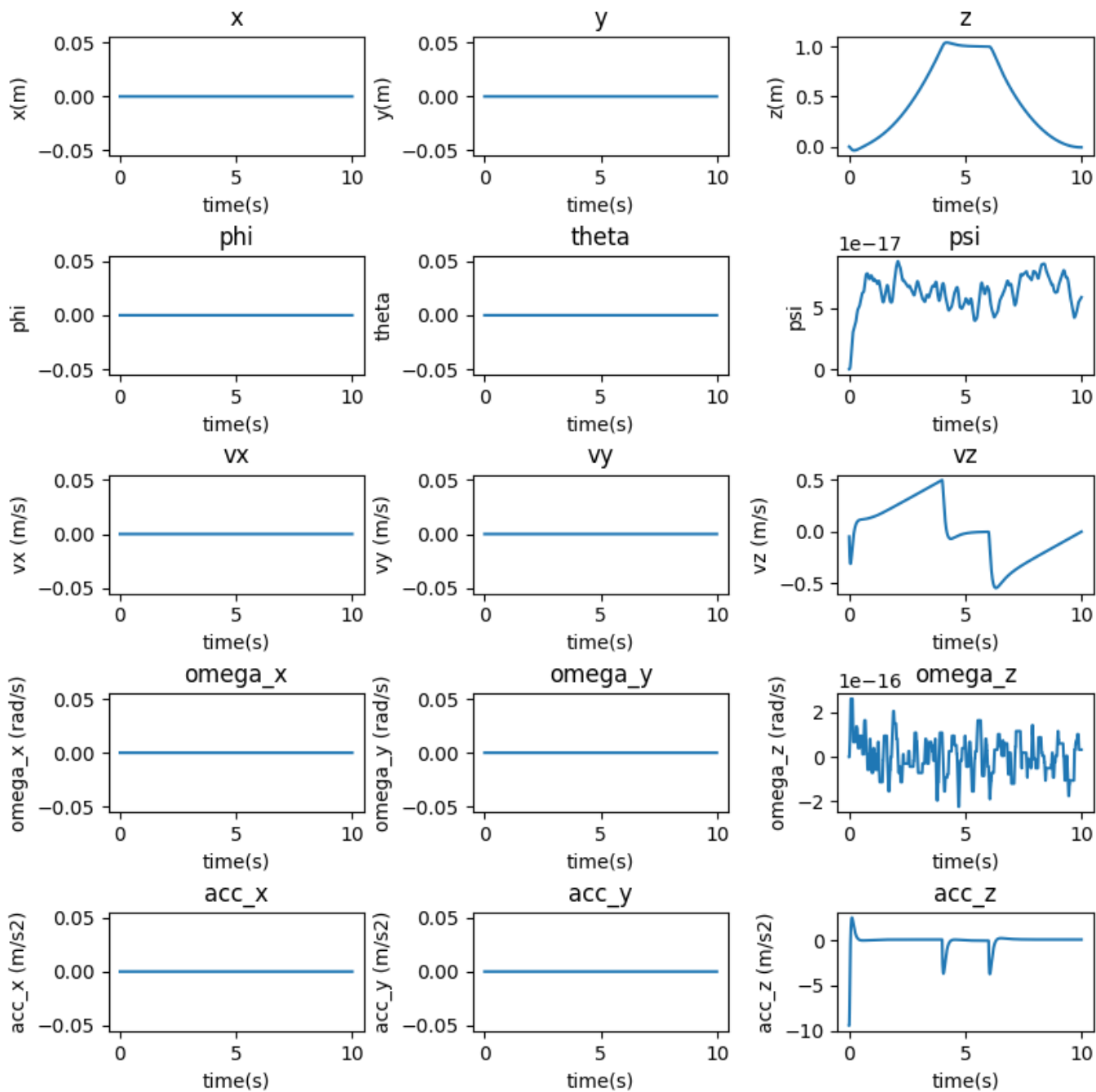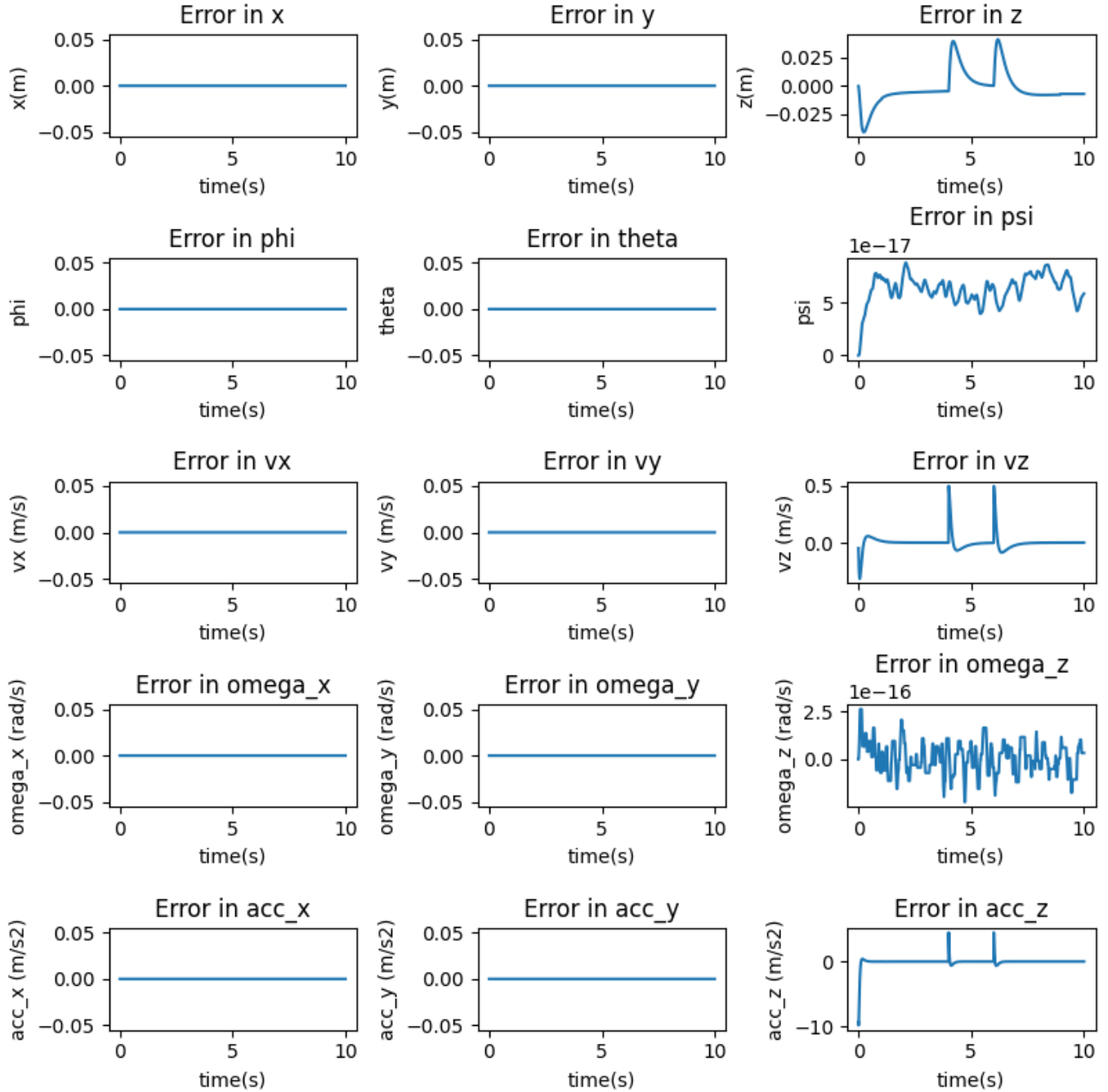
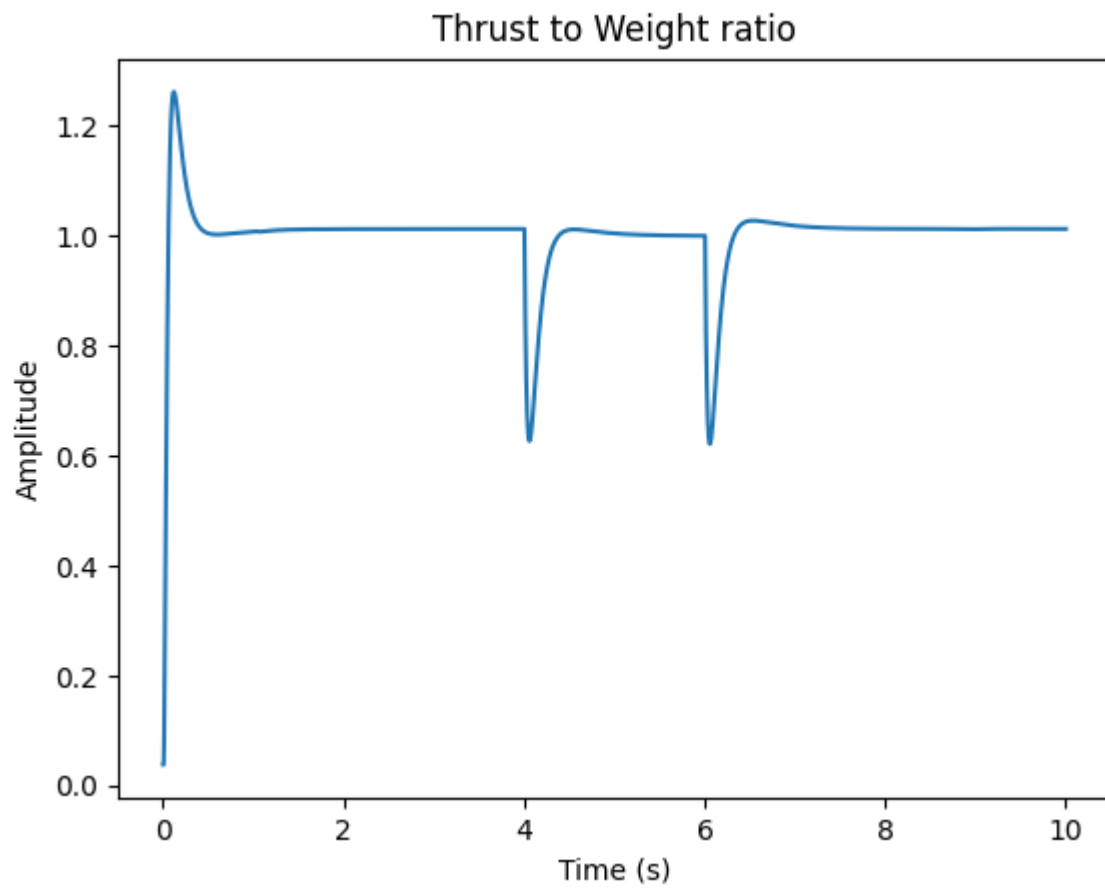The actual poses of the drone are as follows:

The error plots for the requested scenario are as follows:

- The notion of convergence to a waypoint gets tricky in this scenario as the waypoints are provided in an extremely dense manner. The robot sufficiently converges to the desired trajectory as can be seen in the error plots.
- There is no oscillation between waypoints as the provided trajectory is extremely smooth and dense (timestep level trajectory). This means that there is no room for oscillation except at sudden jumps in the input trajectory.
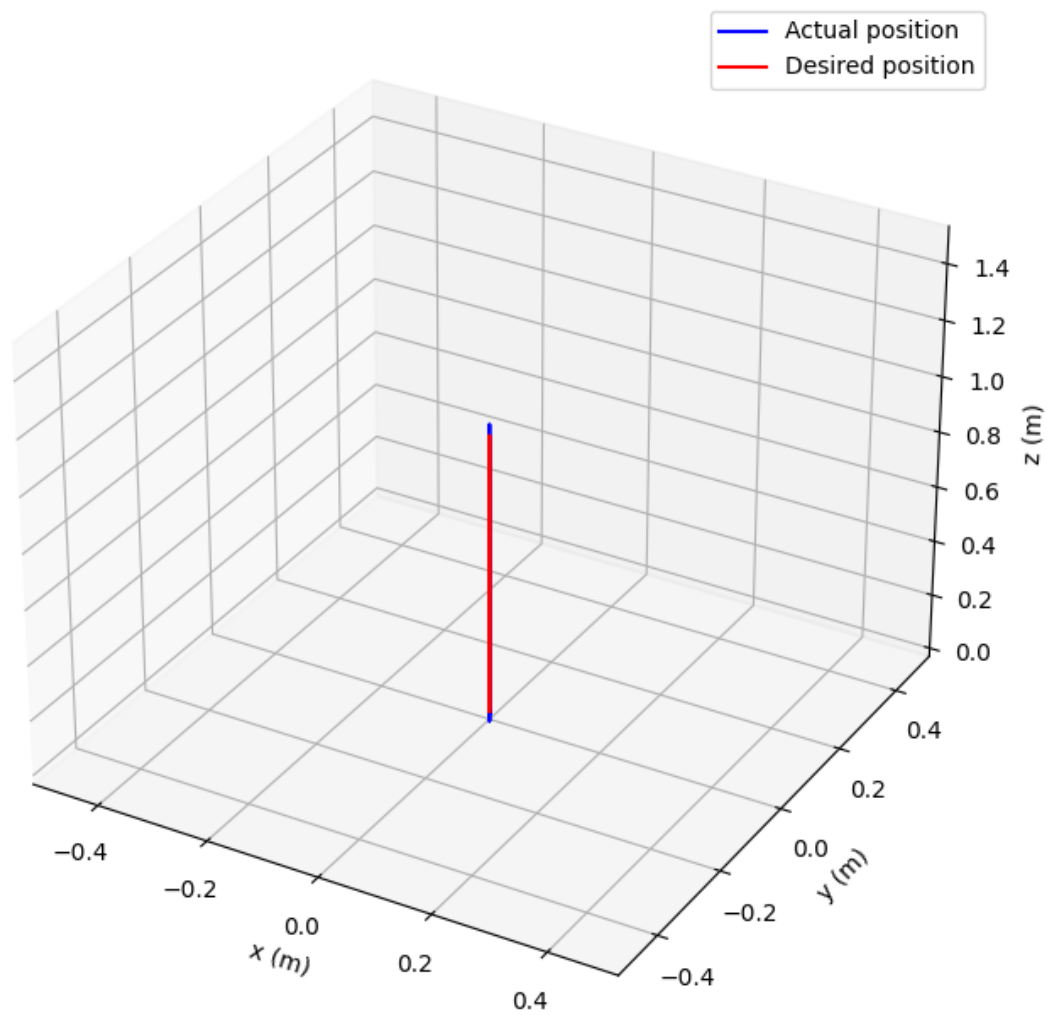
Ensuring that the performance is physically plausible using the thrust to weight plot for this scenario:

Thrust to Weight ratio

We can see that the performance is well within physical limits.

The trajectory in 3D can be seen like this:

Position

Ablation study

Modifying Kd_positions (Outer Loop)

Here the Kd of the outer loop has been increased.

```
Gains:
KXatt = [Kp_phi, Kp_theta, Kp_psi]
KXpos = [Kp_x, Kp_y, Kp_z]
params['kpatt'] = [190, 190, 70]
params['kdatt'] = [30, 30, 18]
params['kppos'] = [20, 20, 10]
params['kdpos'] = [12, 12, 12]
```
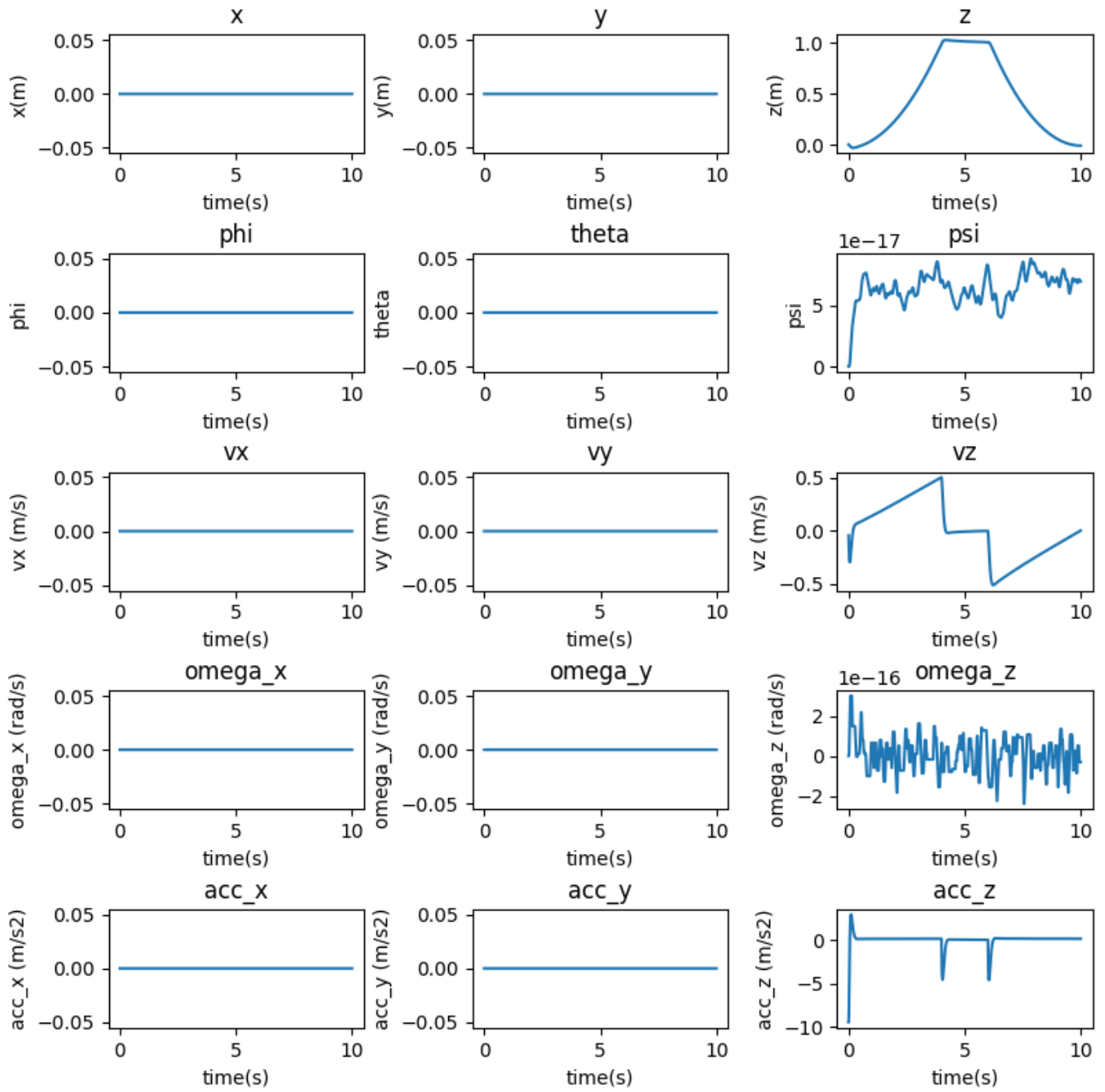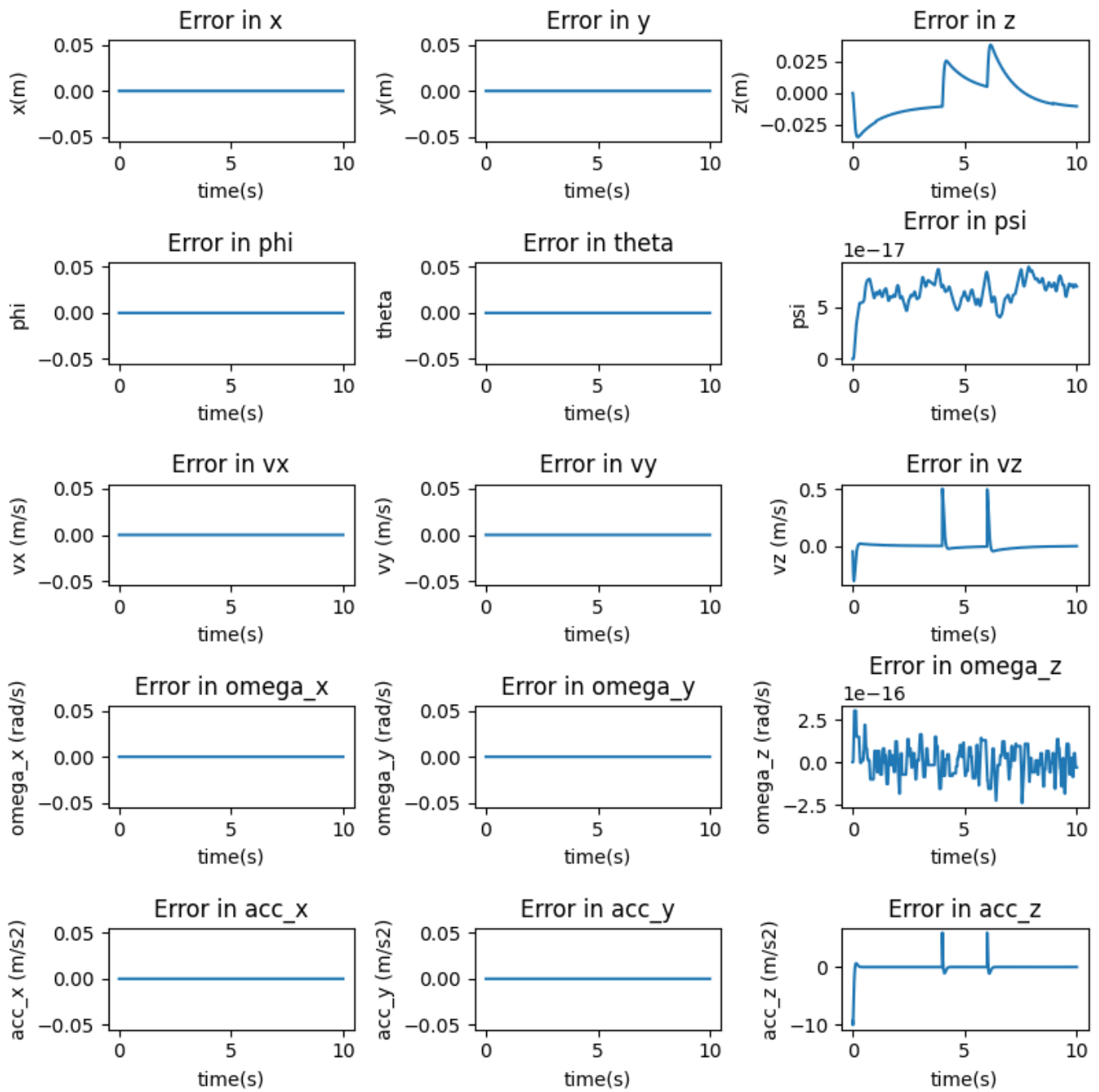
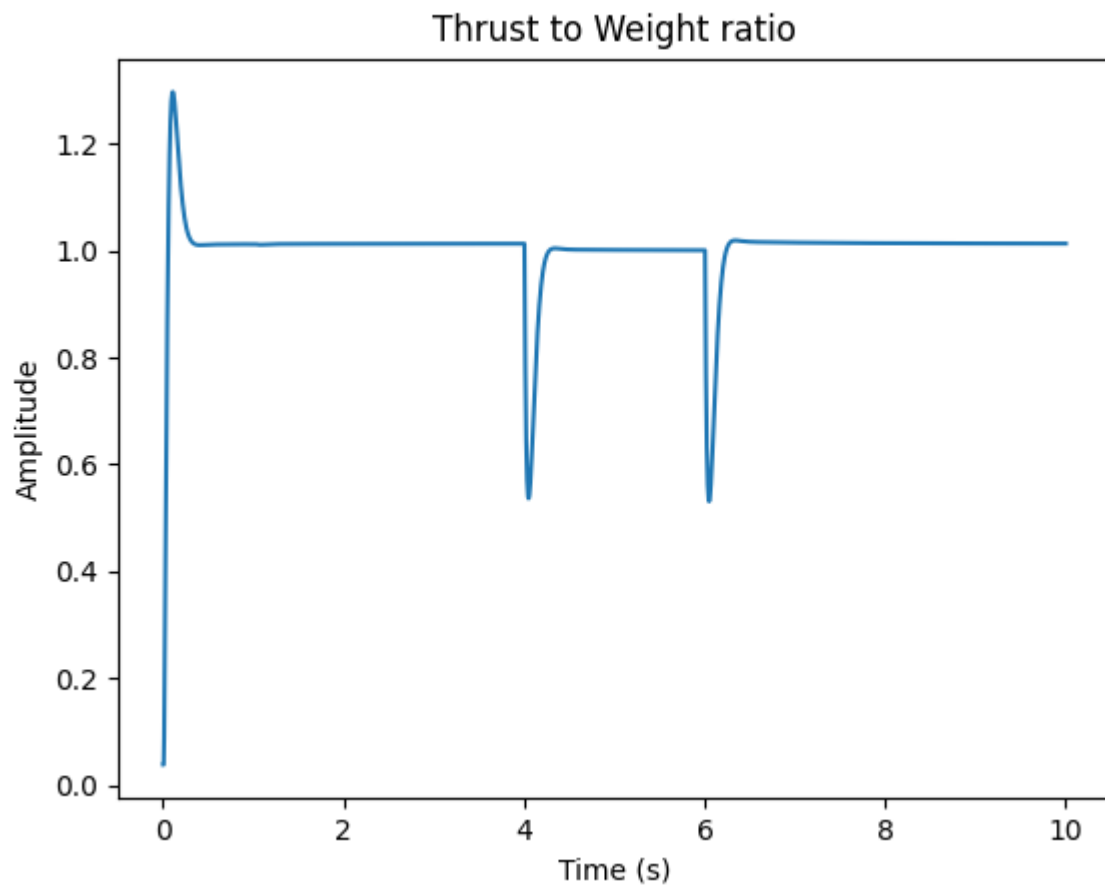The actual poses of the drone are as follows:

The error plots for the requested scenario are as follows:

- We can see that the system reaches the steady state value in a much more delayed fashion - it takes a lot more time for the drone to converge to the desired height. This can more readily be seen in the error plots.

Ensuring that the performance is physically plausible using the thrust to weight plot for this scenario:

Thrust to Weight ratio

We can see that the performance is well within physical limits.

Modifying Kp_positions (Outer Loop)

Here, the Kp of the outer loop has been increased.

```
Gains:
KXatt = [Kp_phi, Kp_theta, Kp_psi]
KXpos = [Kp_x, Kp_y, Kp_z]
params['kpatt'] = [190, 190, 20]
params['kdatt'] = [30, 30, 18]
params['kppos'] = [25, 25, 20]
params['kdpos'] = [8, 8, 9]
```
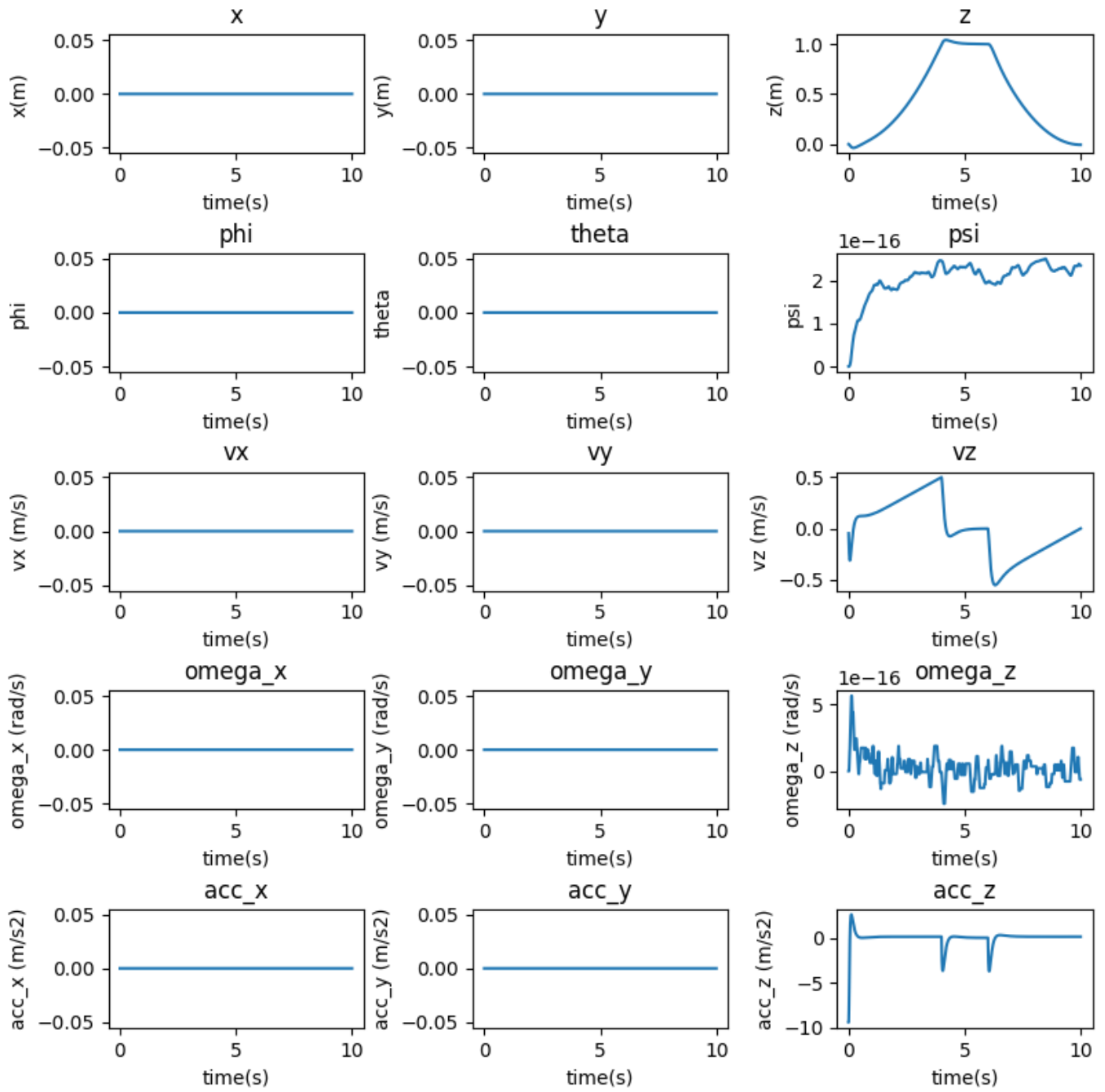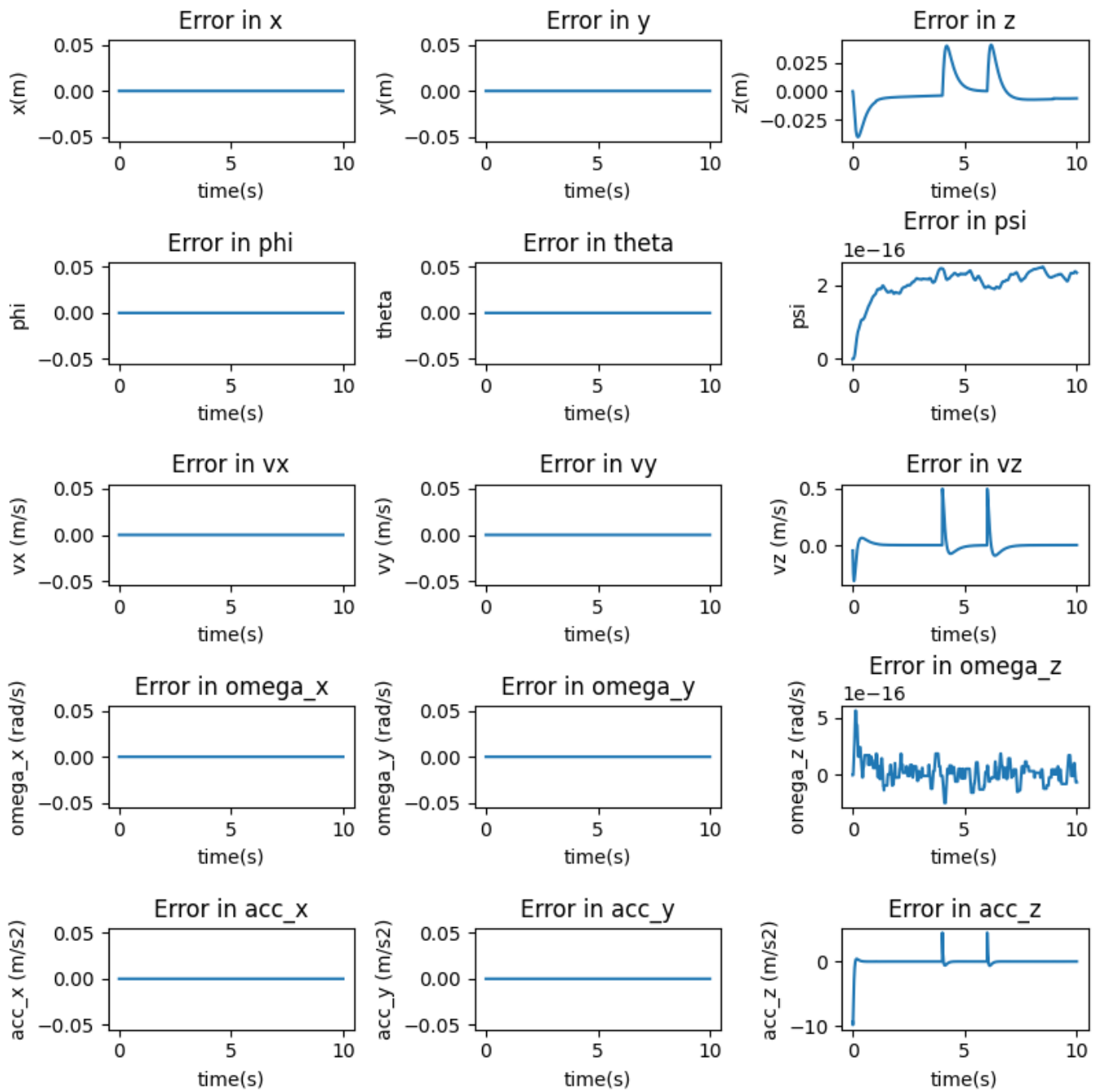
The actual poses of the drone are as follows:

The error plots for the requested scenario are as follows:

- We can see that the overshoot has been increased and the drone reaches the desired x-location much quicker than previously. This is as expected when we increased the Positional Kp value.

Ensuring that the performance is physically plausible using the thrust to weight plot for this scenario:
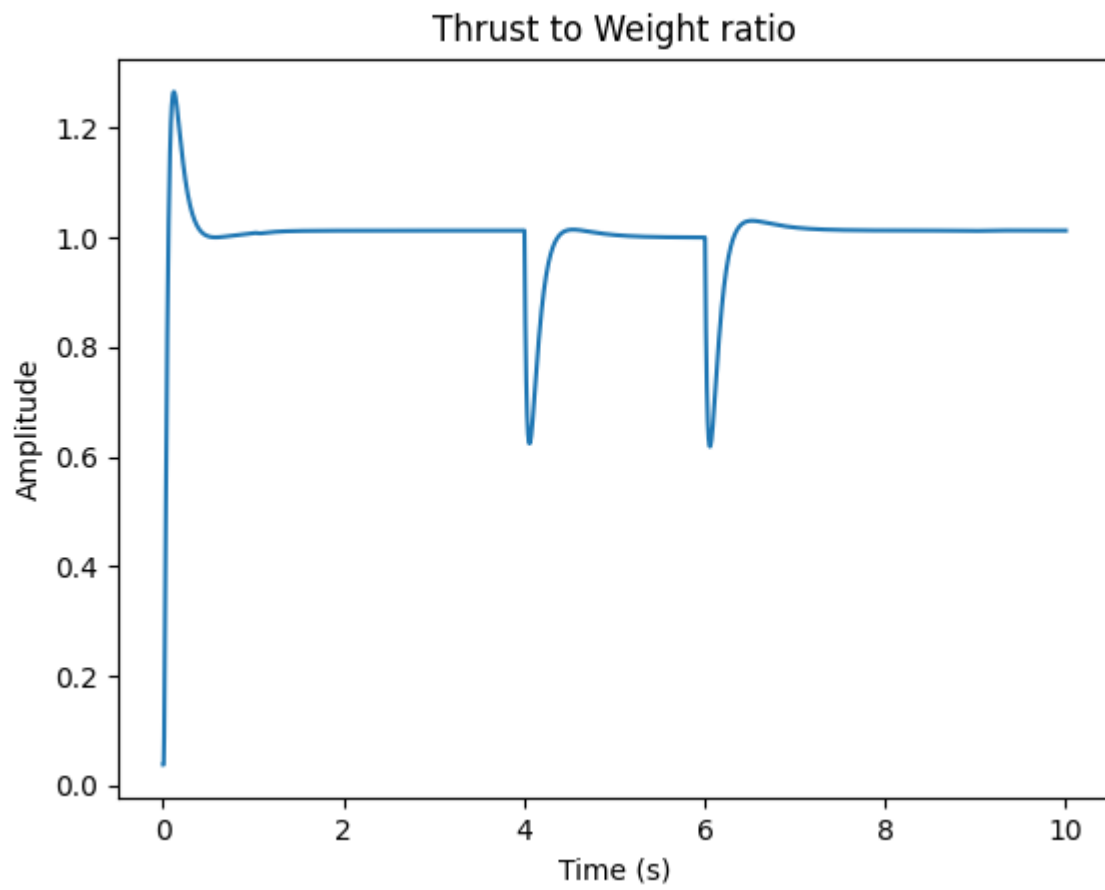
We can see that the performance is well within physical limits.

Question 4

In this question, we develop a state machine using the following method:

- A separate class called StateManger manages the state for the entire drone mission.
- The StateManager maintains 1 of 7 possible states that is:
  - IDLE: The system begins in this state generating no control inputs.
  - TAKEOFF: The system begins to takeoff and goes to a desired hover location at 0.5m height.
  - HOVER1: The system maintains its location for a short duration of 3 seconds.
  - TRACK: The system tracks a given trajectory.
  - HOVER2: The system maintains its location for a short duration of 3 seconds.
  - LAND: The system begins its descent from its current location and reaches the ground.
  - COMPLETE: The mission has been executed successfully.
- The StateManager maintains an internal Finite State Machine architecture that allows the system to go to the next state only if the trajectory has been completely tracked for a given state and the drone has converged to the given location of the trajectory.
- The StateManager moves the state of the system by incrementing the current state to the next state once the current state has completed execution.
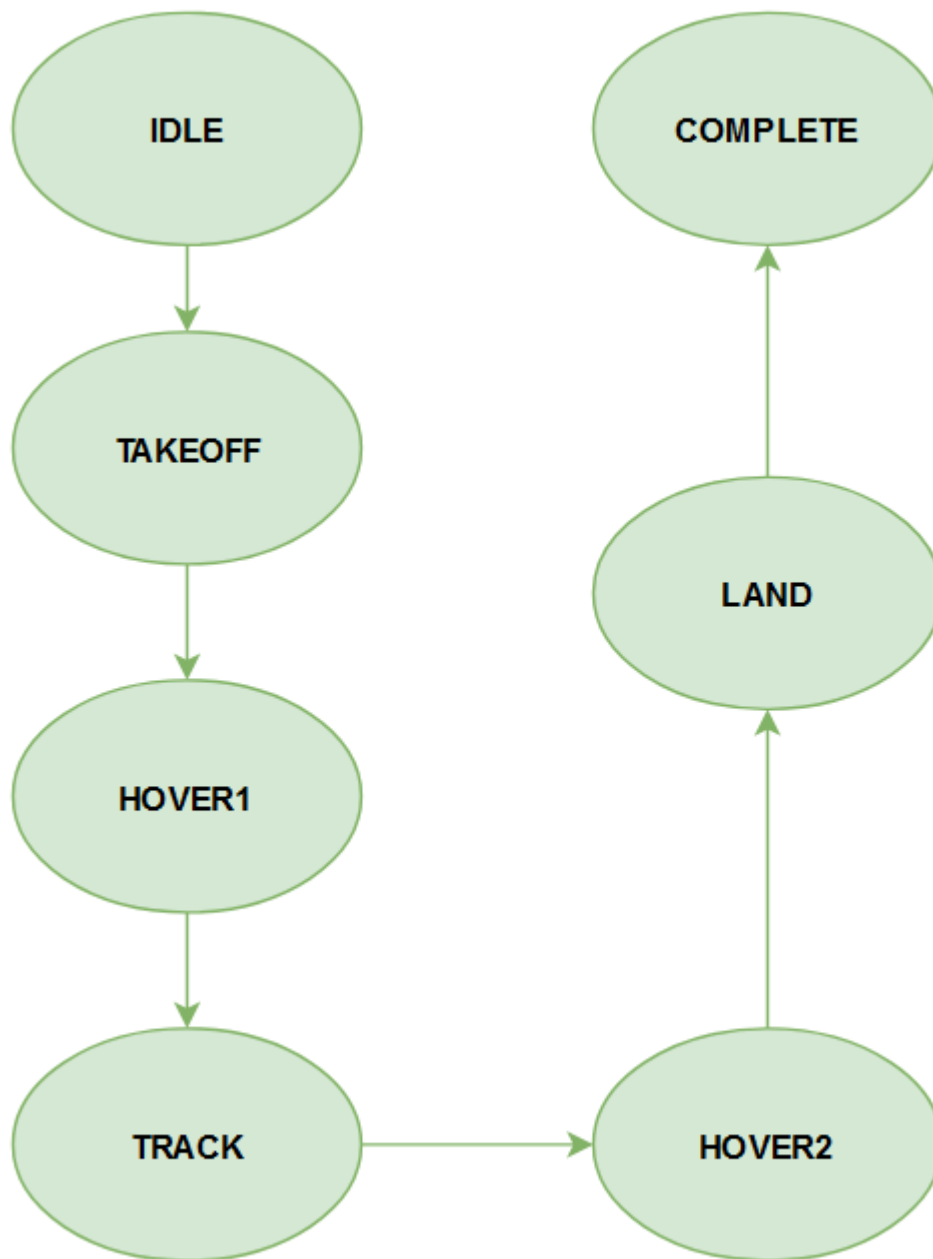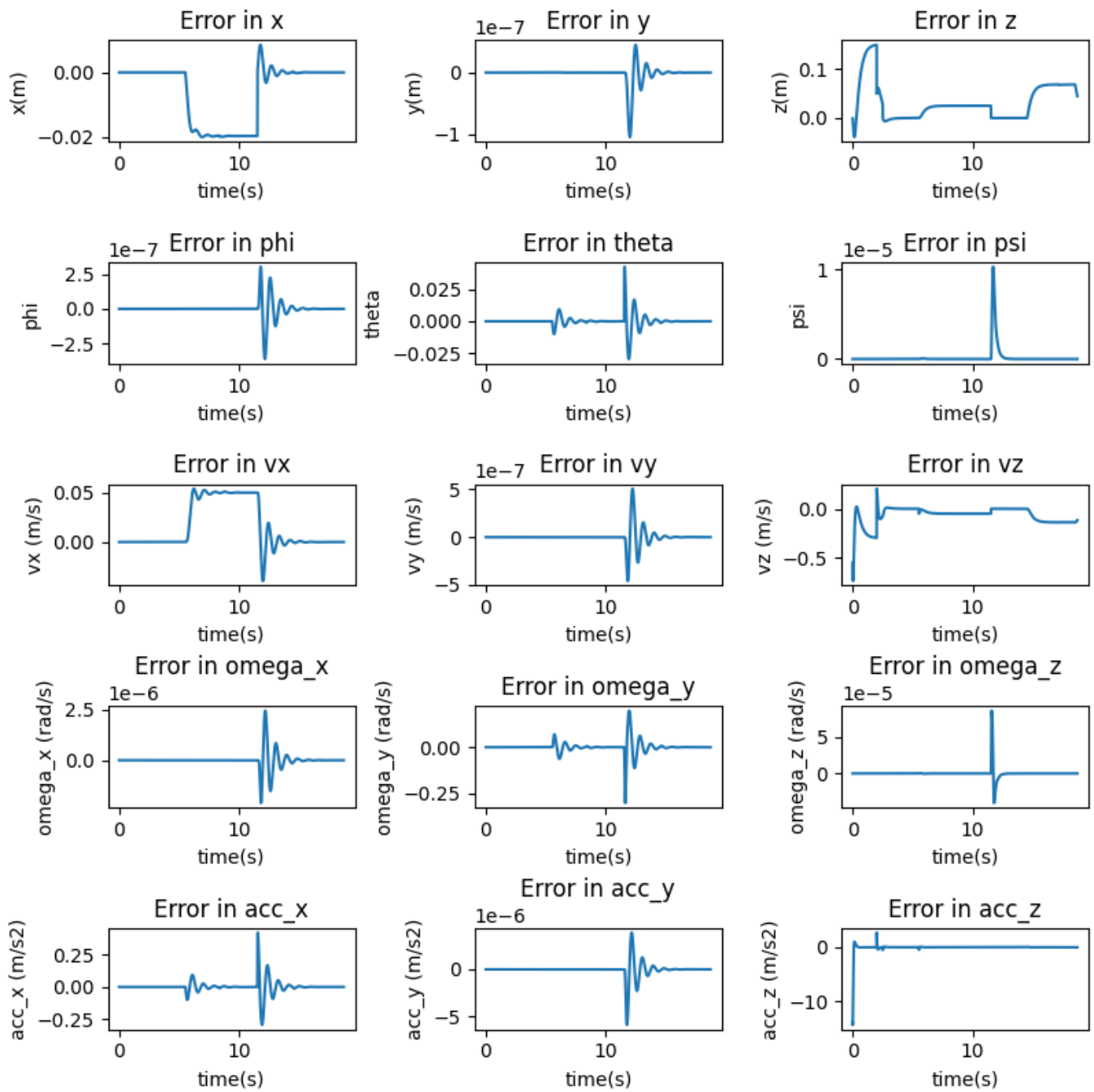
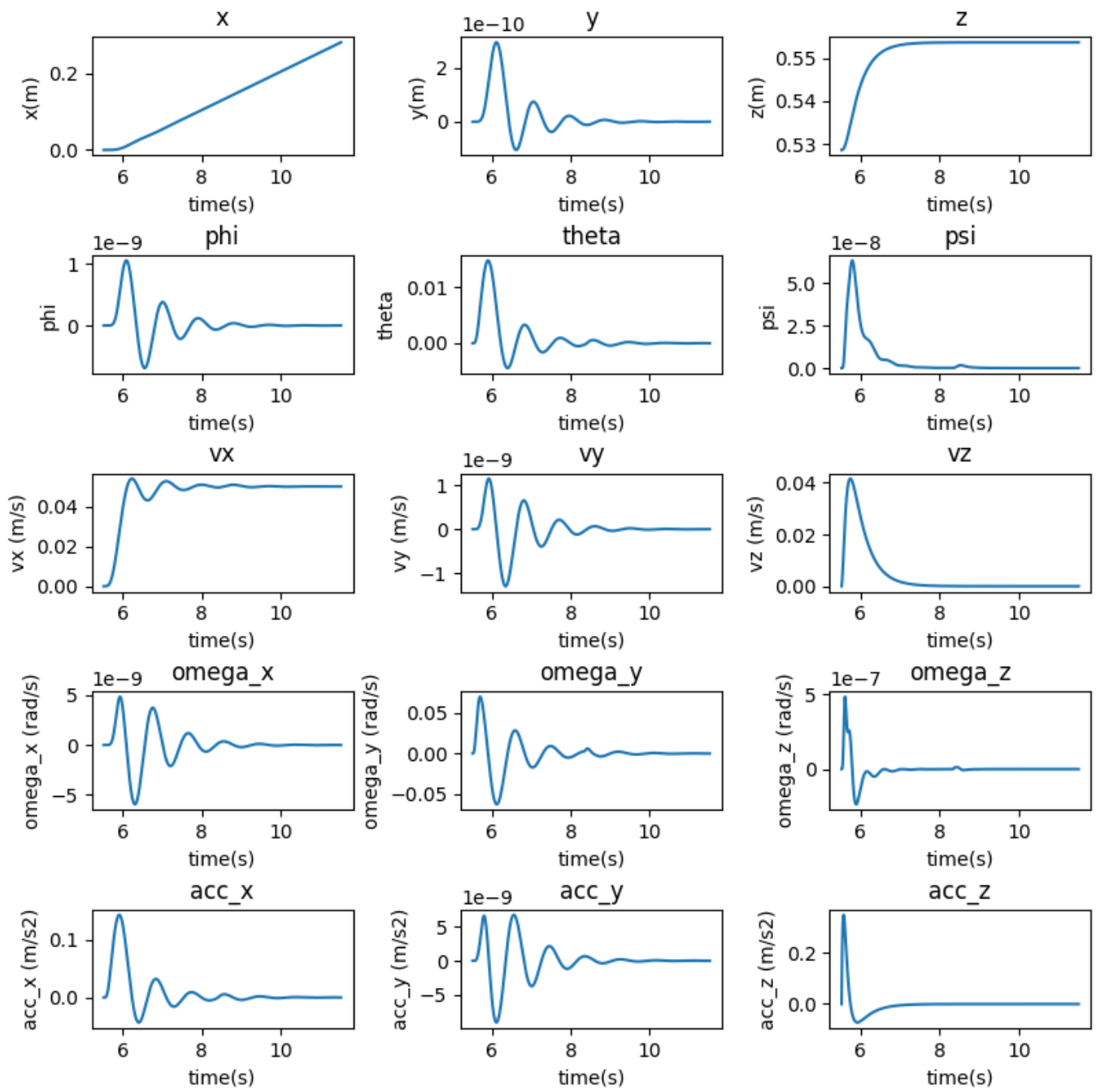The trajectory for a simple line track is shown below:

Position
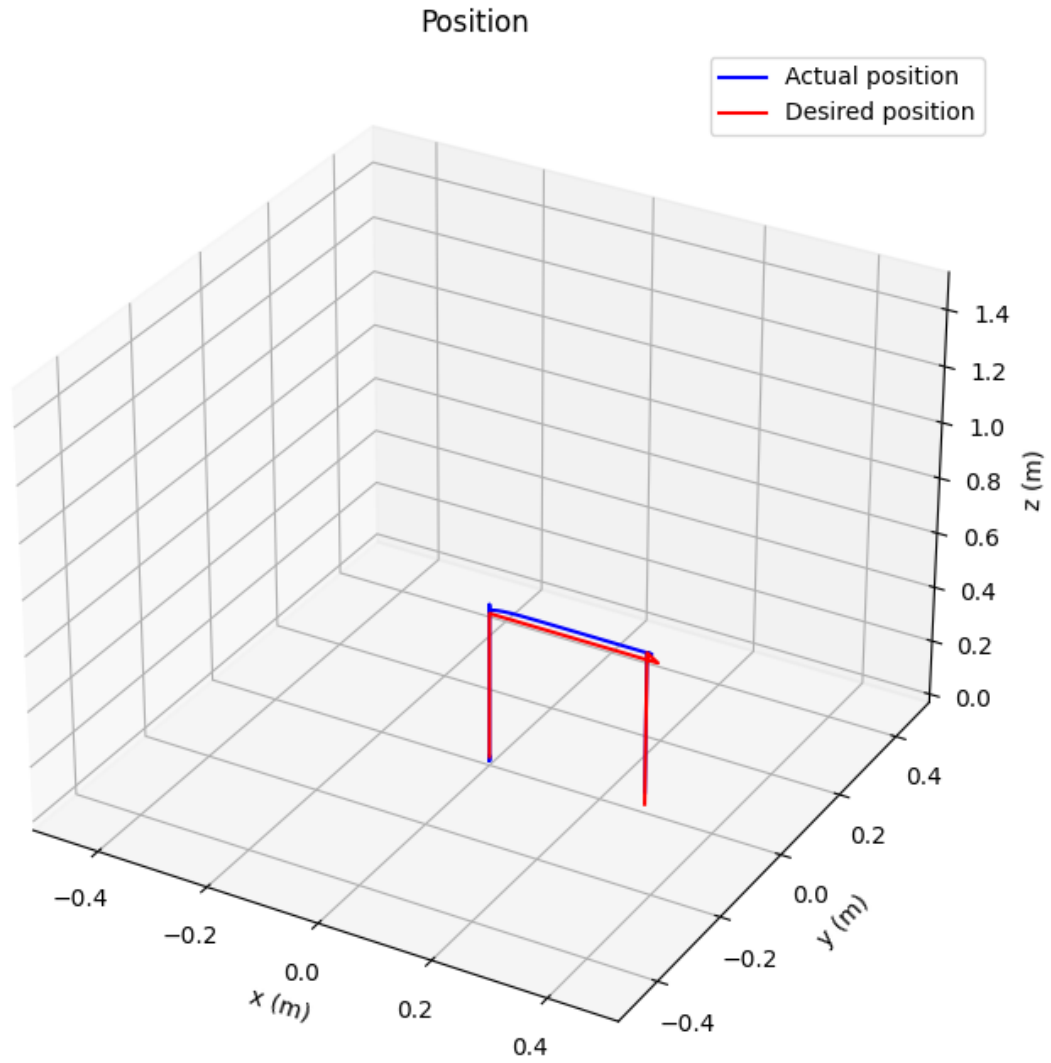
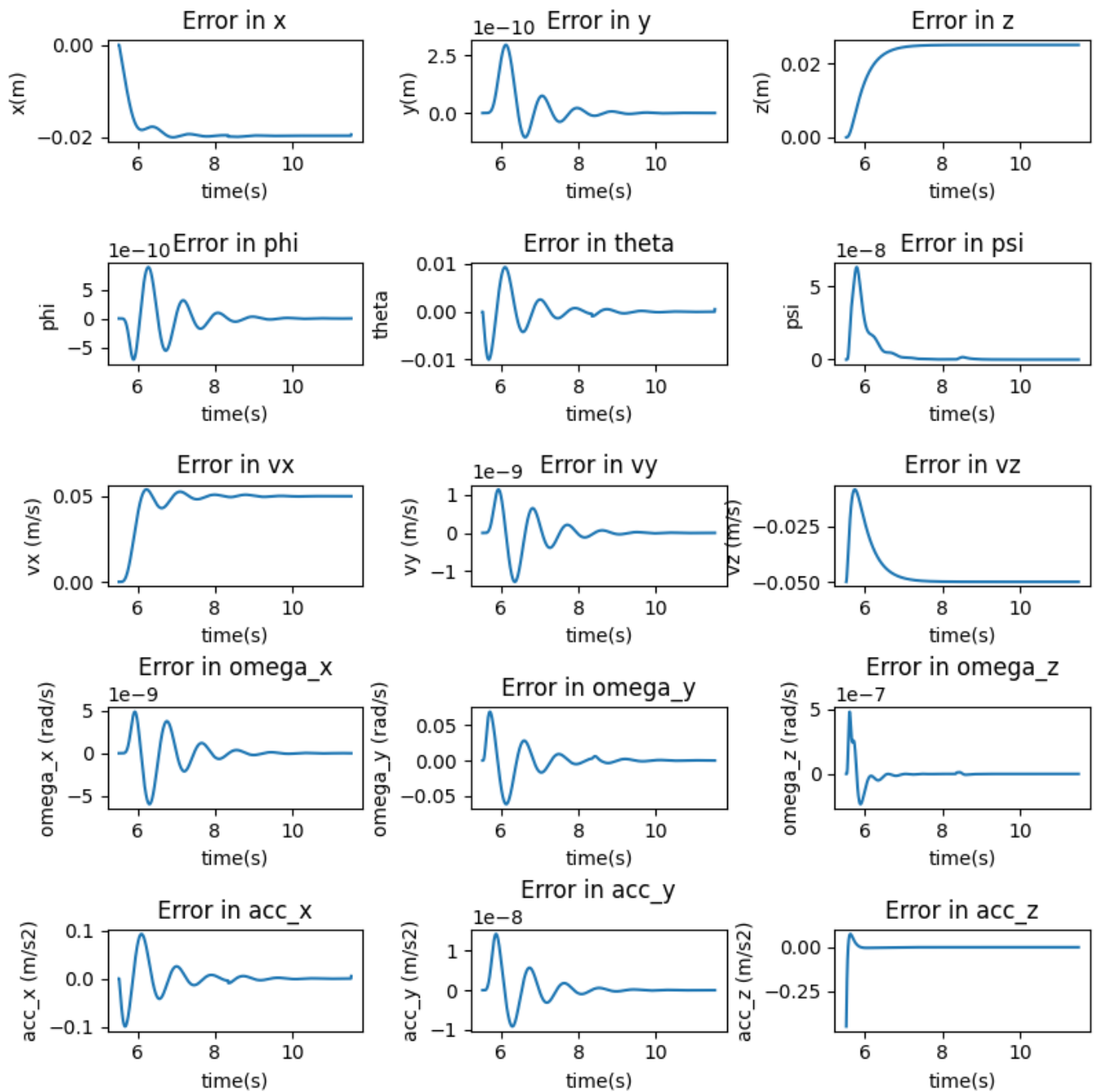The state plots for the entire duration are:
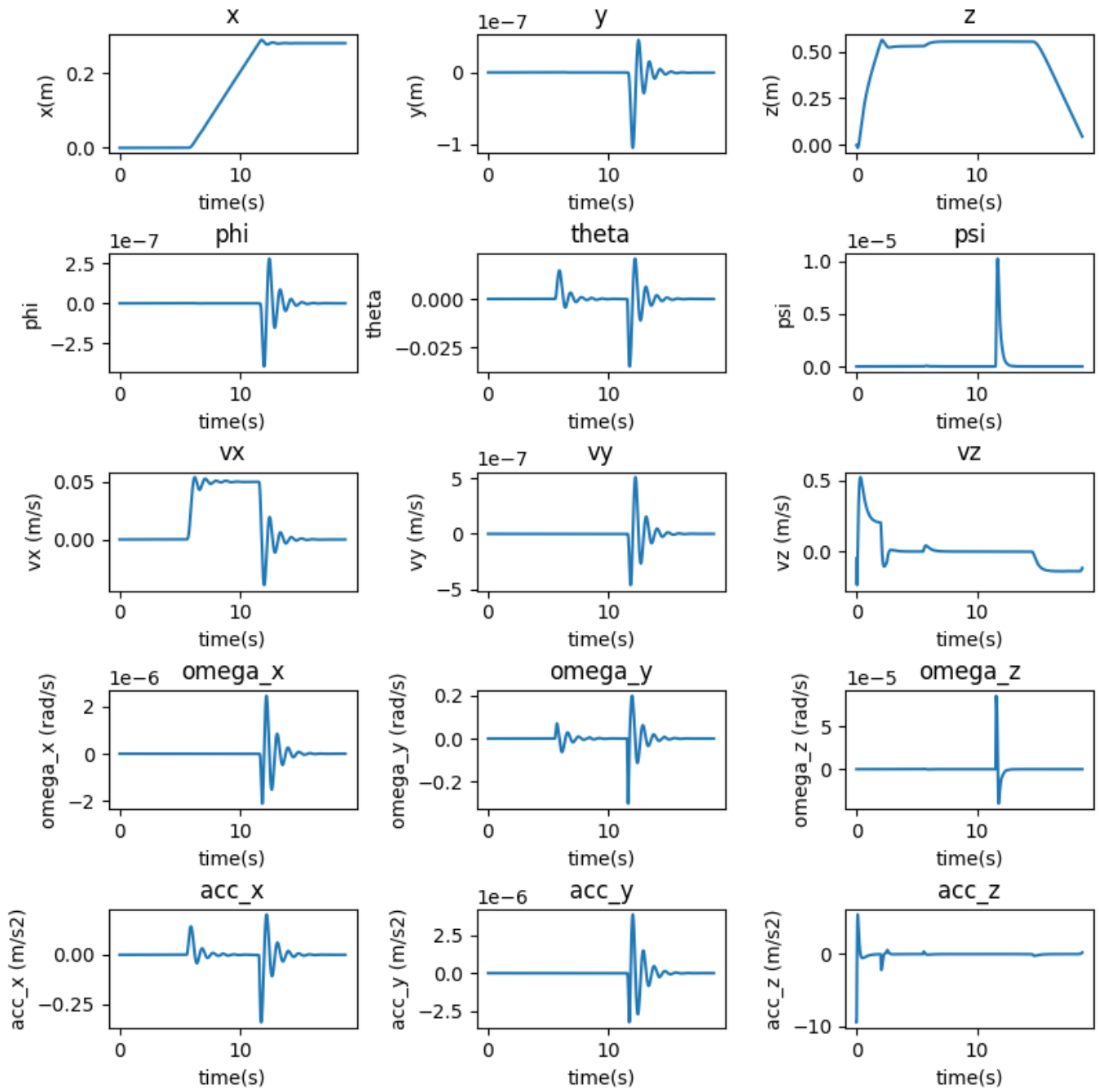
Error plots for the entire duration:

The state plots for the given duration of the tracking phase are:

Error plots for the tracking duration:

Question 5

```
Gains:
KXatt = [Kp_phi, Kp_theta, Kp_psi]
KXpos = [Kp_x, Kp_y, Kp_z]
"kpatt": [  190,    190,     70],
"kdatt": [   30,     30,     18],
"kppos": [   20,     20,     18],
"kdpos": [    8,      8,      9],
```
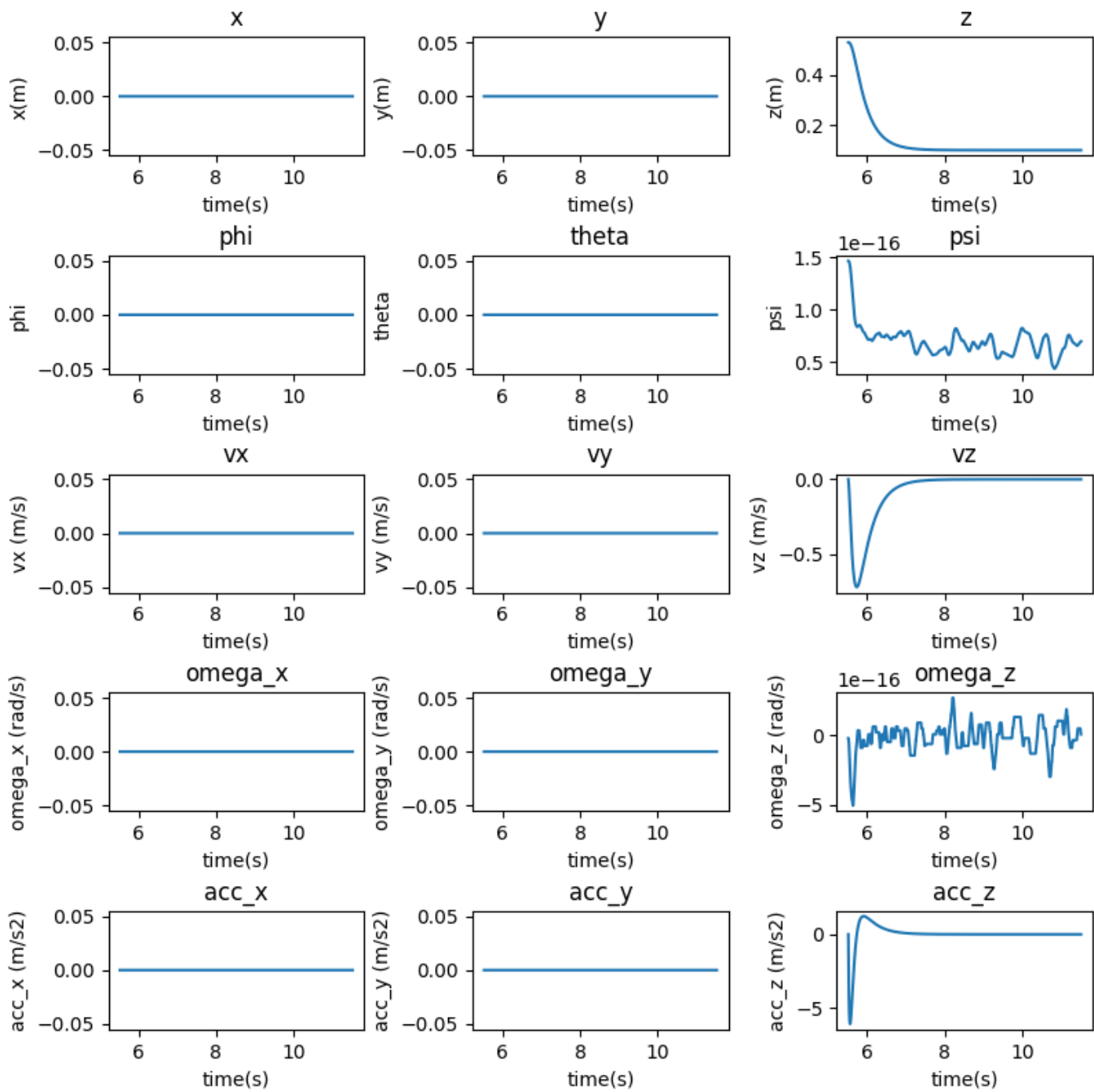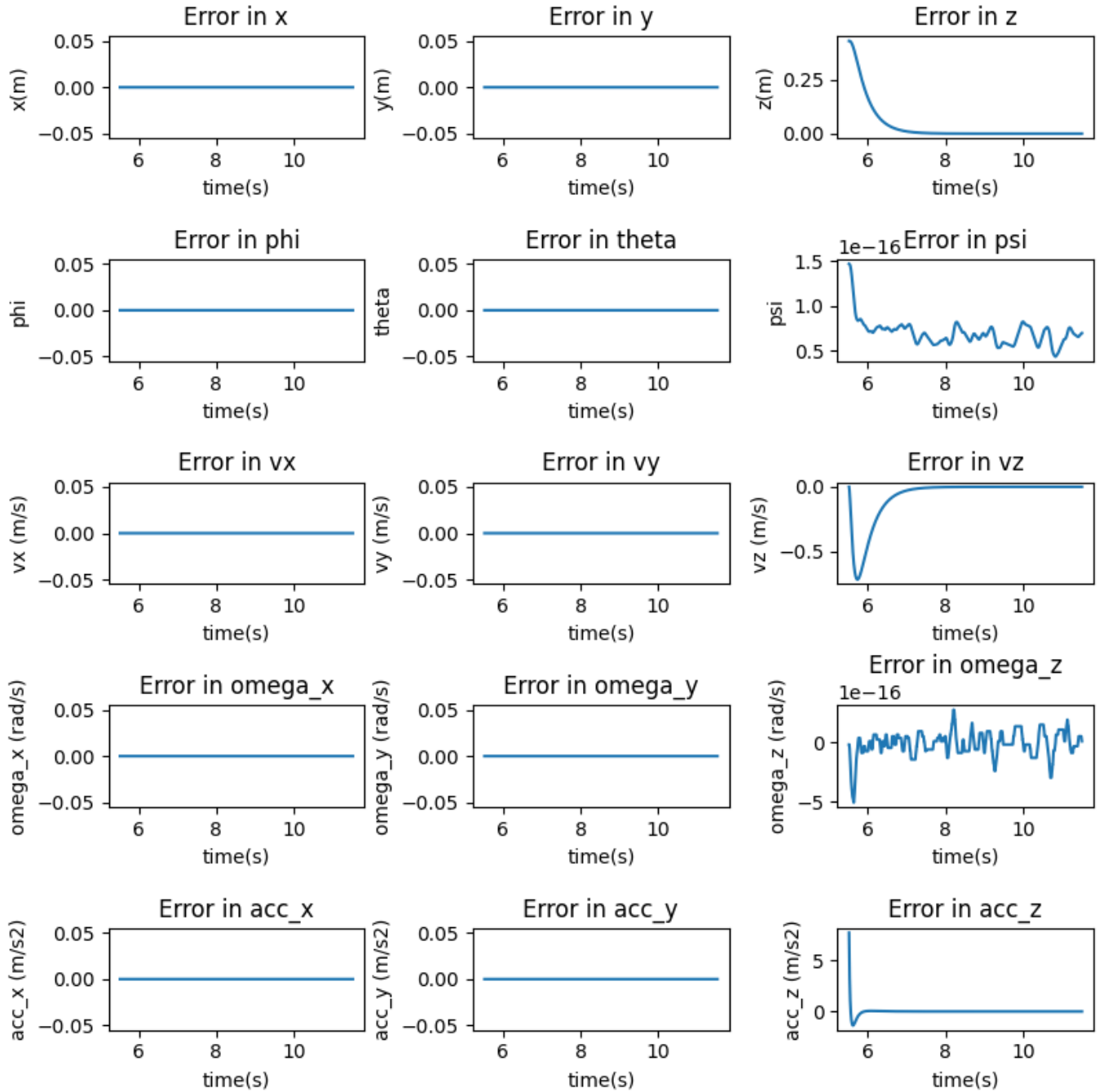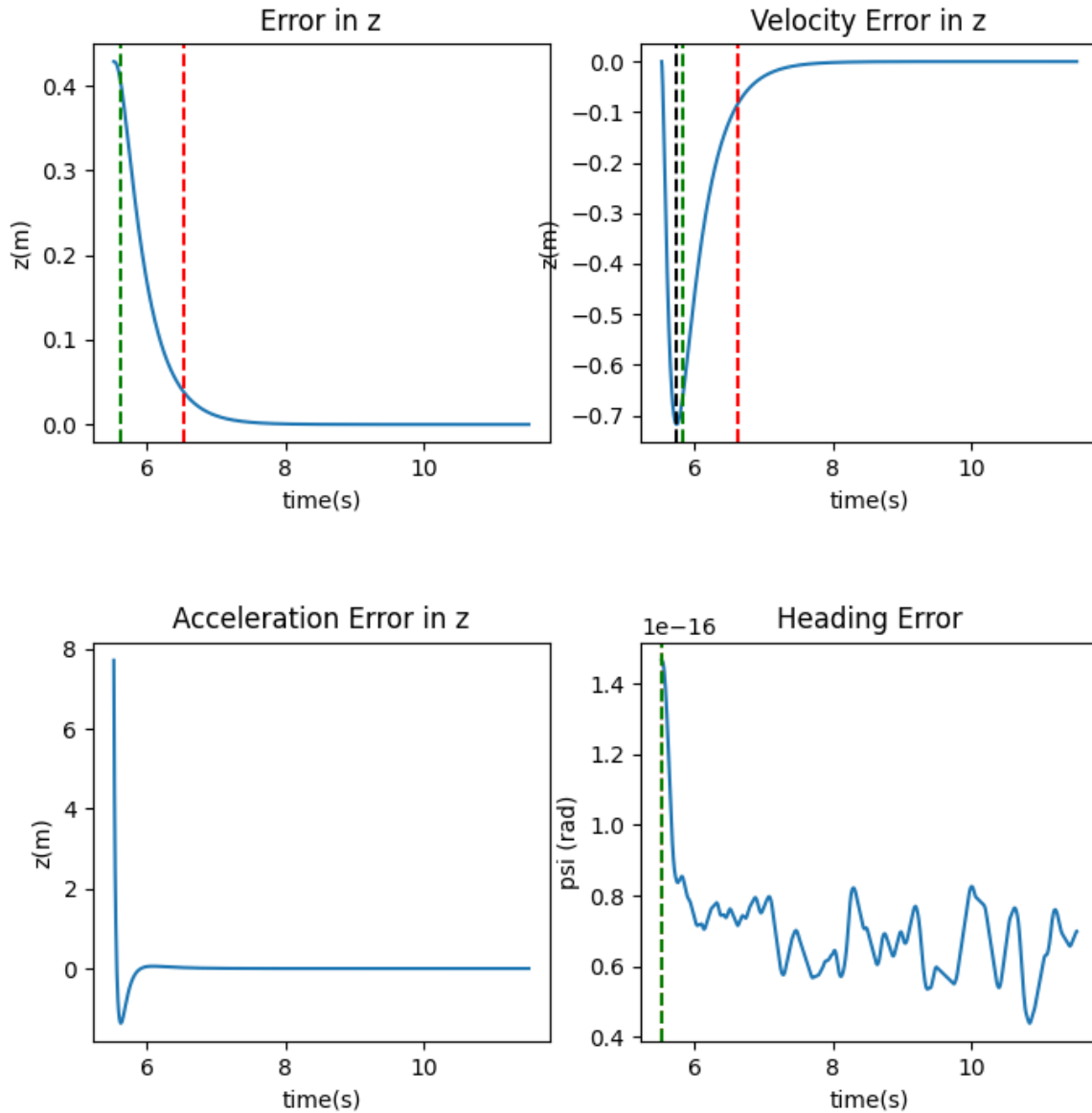
The actual poses of the drone are as follows:

The error plots for the requested scenario are as follows:

The rise and settling times are shown:

Error in z

Velocity Error in z

Acceleration Error in z

Heading Error

```
-------System Statistics--------
[Z] Settling time =  0.9999999999999787
[Z] Rise time =  0.9099999999999806
[Z] Steady State values =  0.10000003323214719
[Vz] Settling time =  0.894999999999809
[Vz] Rise time =  0.794999999999983
[Vz] Steady State values =  0.10000003323214719
[psi] Settling time =  0.0
[psi] Rise time =  0.0
[psi] Steady State values =  4.002280801119769e-15
```
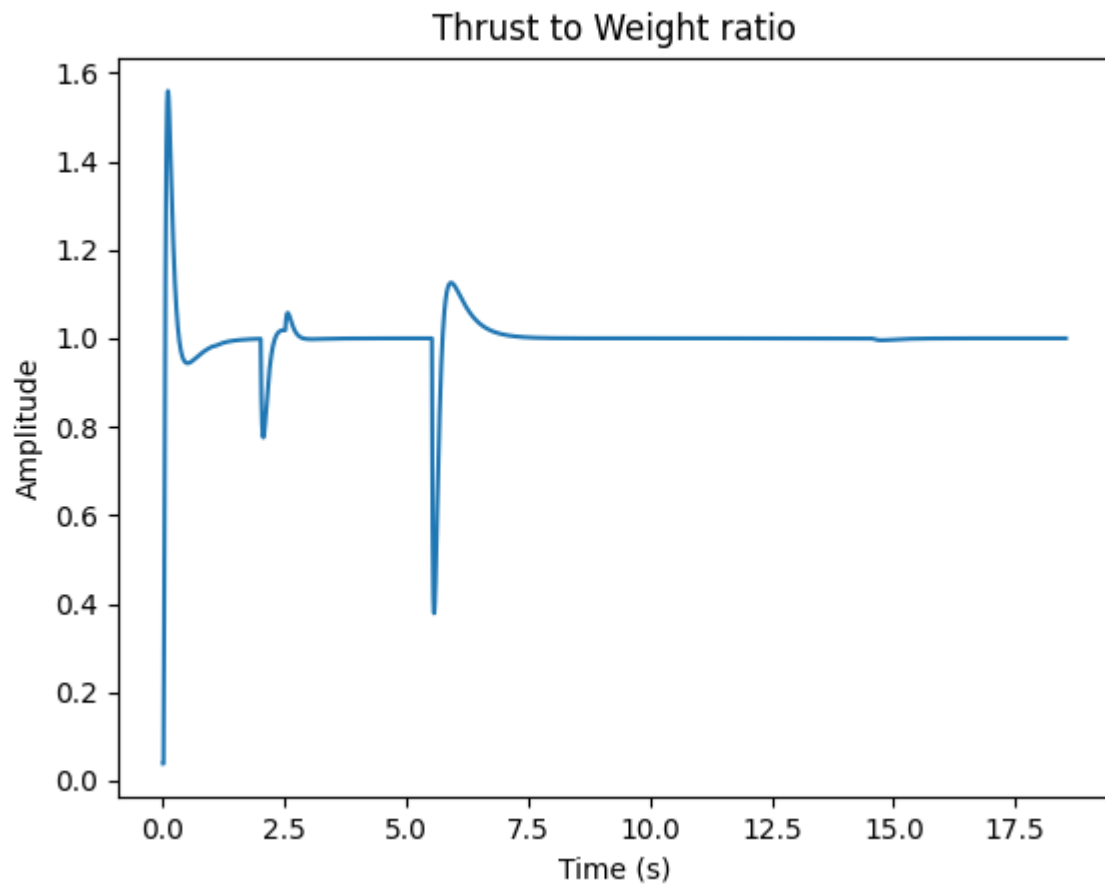
- From the plots it is evident that all the overshoots are 0% as the system is overdamped. The code automatically saves these values to the output folder.

Note: Since the starting and ending velocities of the drone is set as the same value of 0, the notions of settling and rise time do not have much meaning. Here, the start duration is computed from the peak value of the velocity as it is a measure of the performance of the drone.

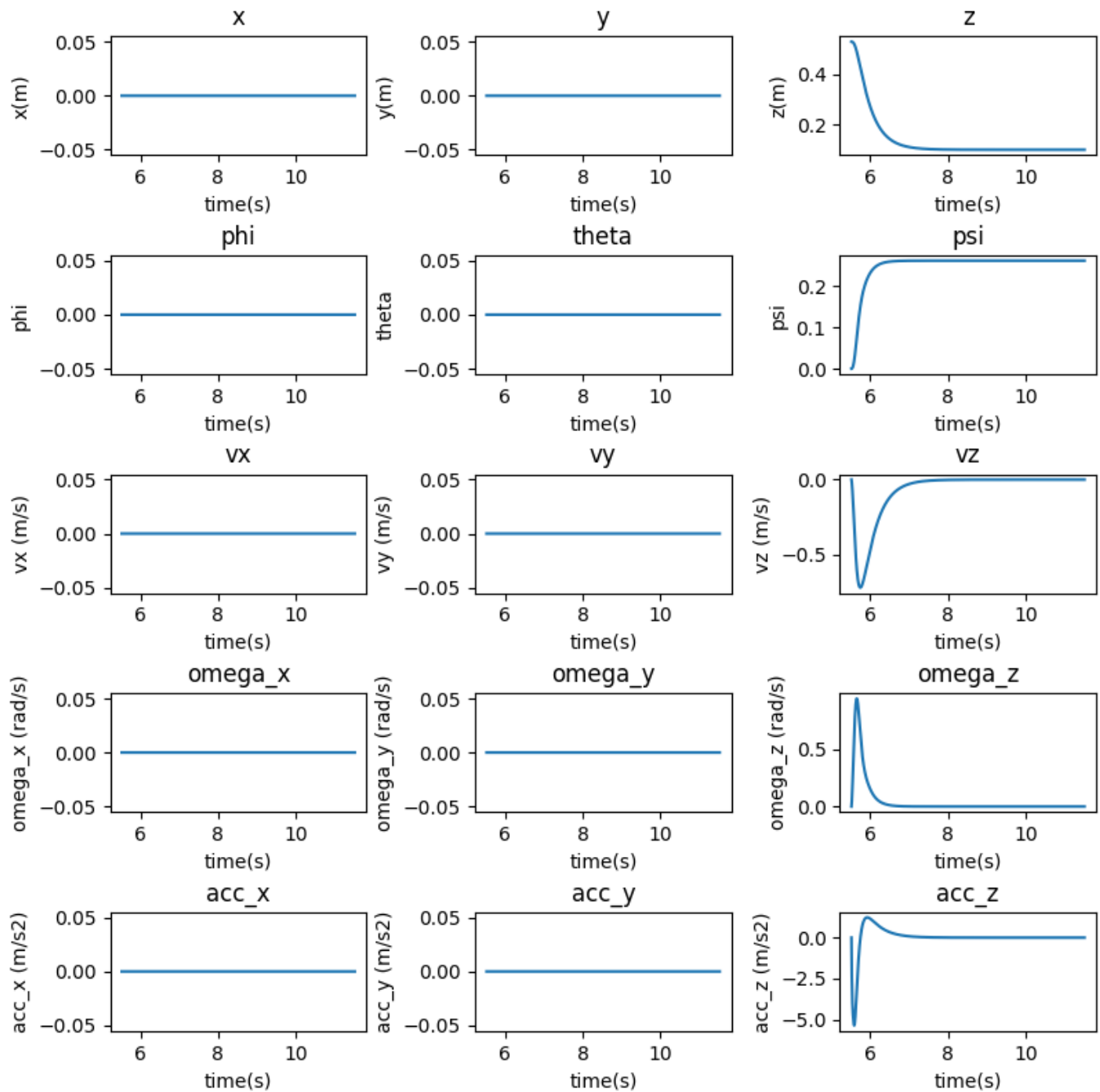Ensuring that the performance is physically plausible using the thrust to weight plot for this scenario:

Thrust to Weight ratio

We can see that the performance is well within physical limits.

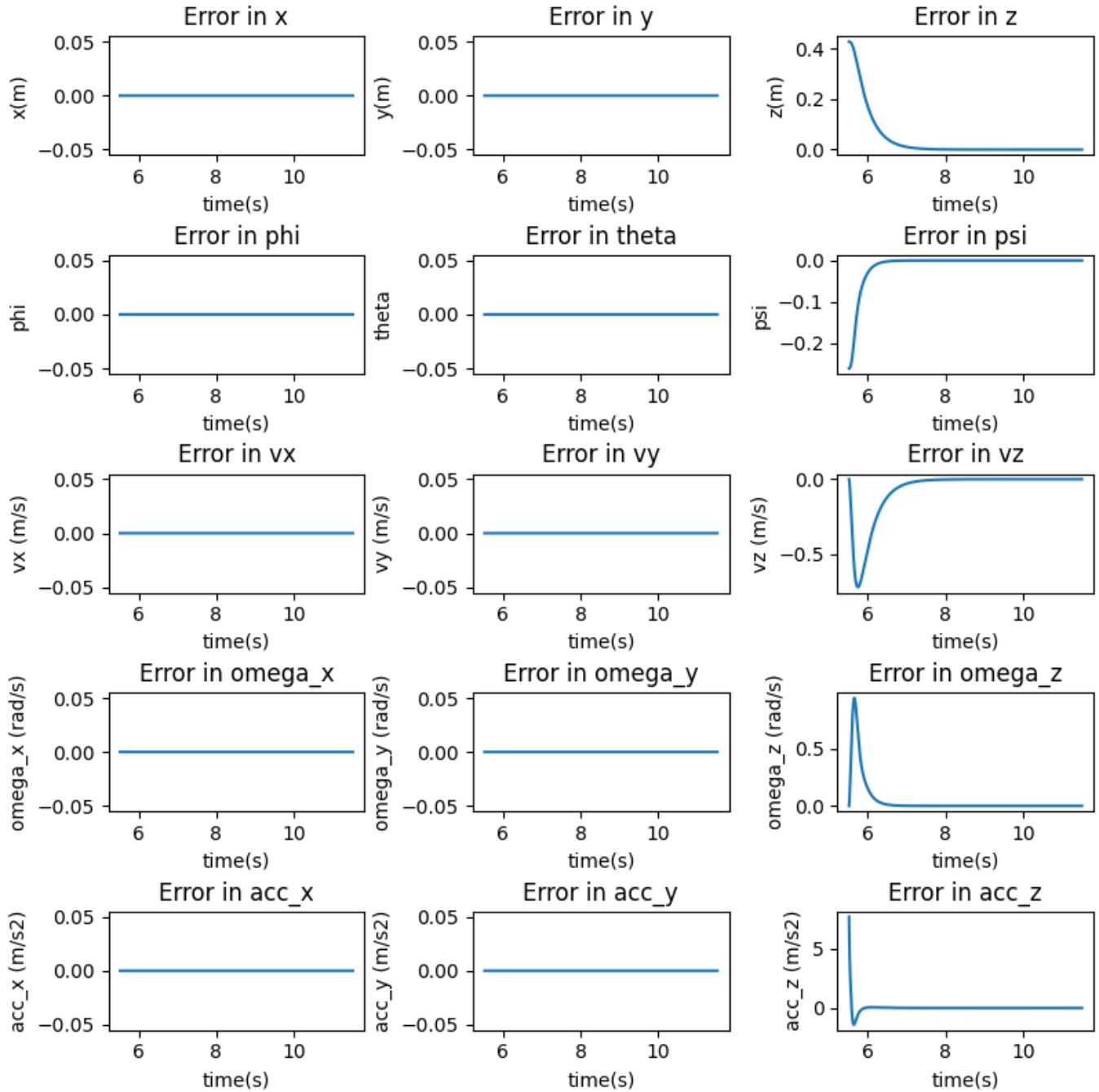The trajectory in 3D can be seen like this:

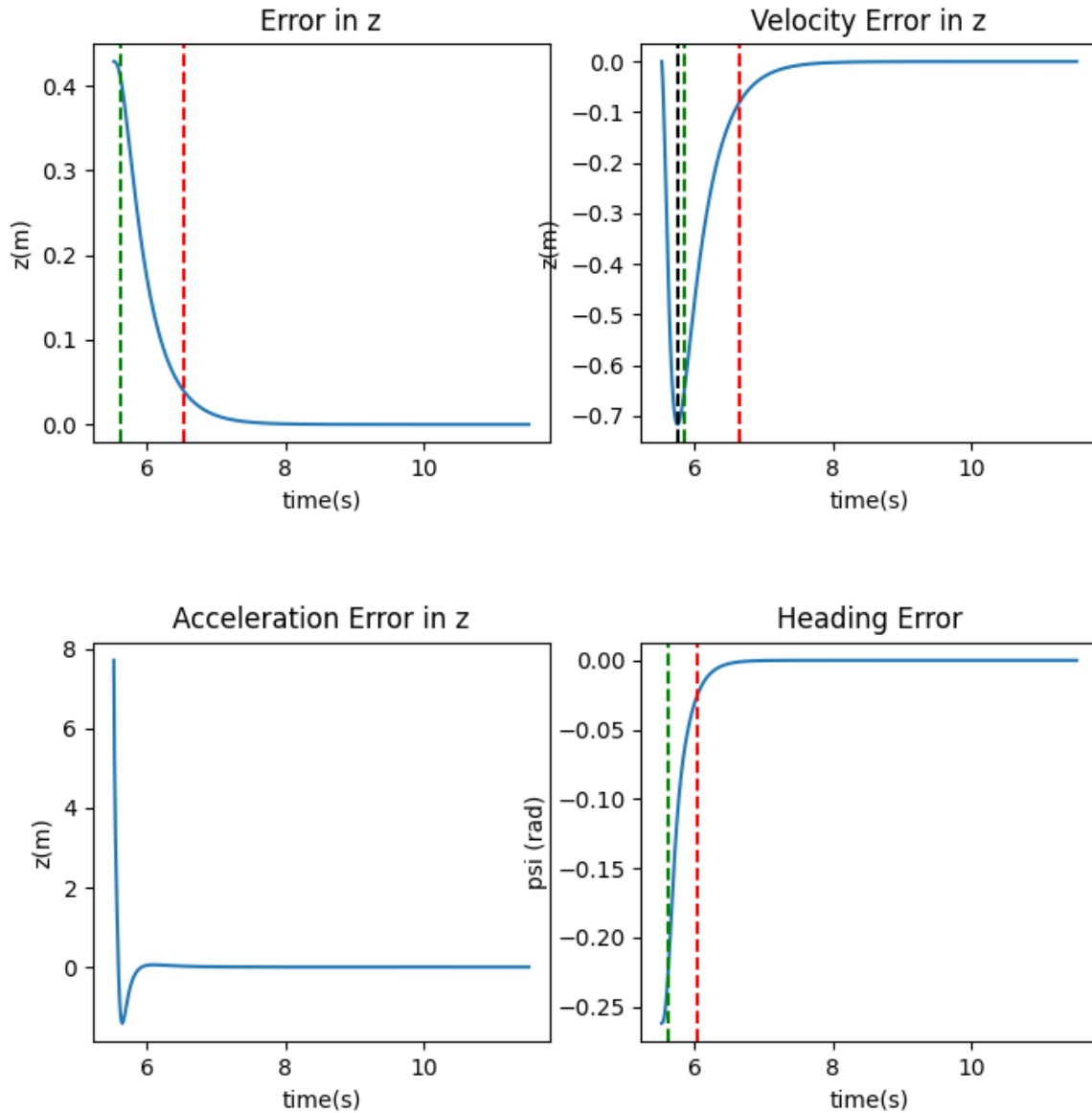Position
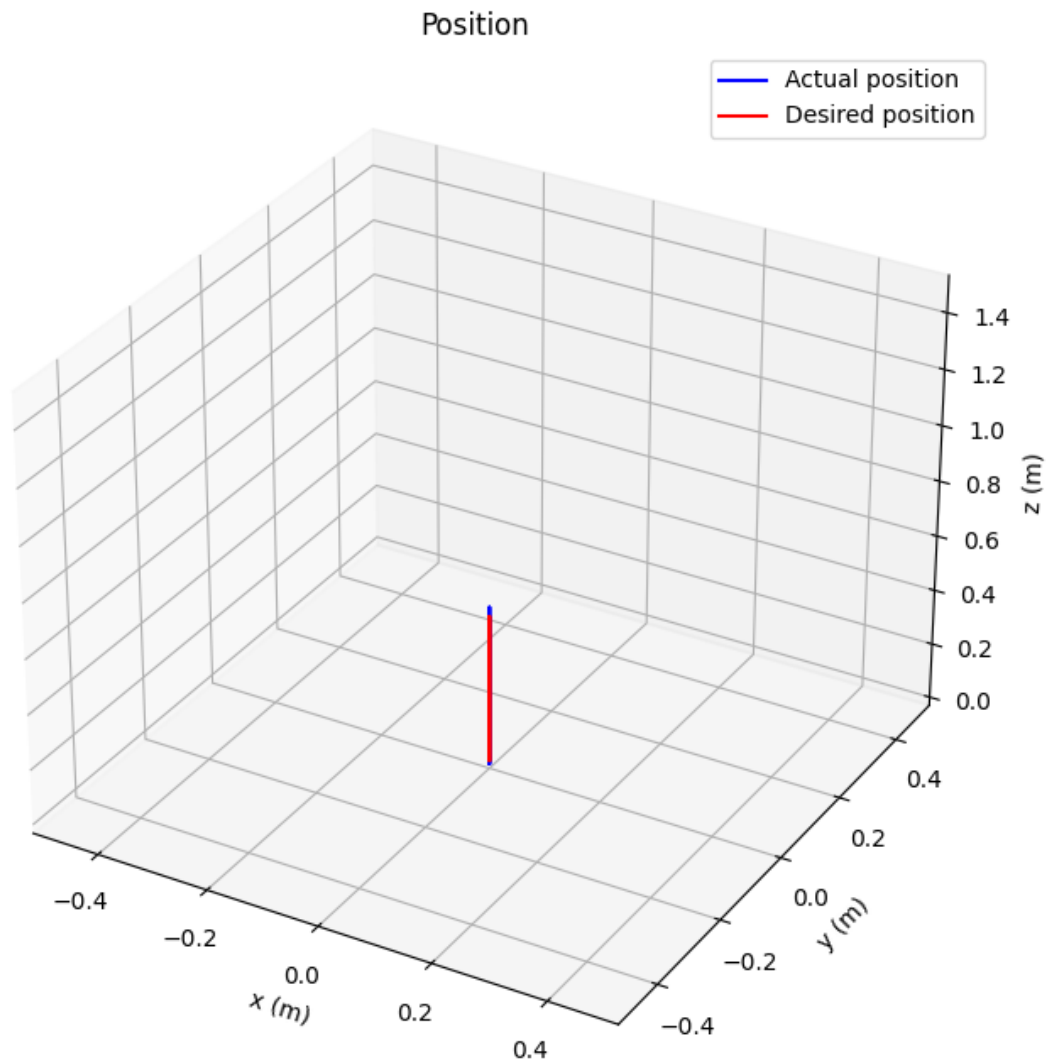
Including Variation in yaw.

The actual poses of the drone are as follows:

The error plots for the requested scenario are as follows:

The rise and settling times are shown:

```
-------System Statistics--------
[Z] Settling time =  1.0099999999999785
[Z] Rise time =  0.9099999999999806
[Z] Steady State values =  0.1000000342445711
[Vz] Settling time =  0.88999999999981
[Vz] Rise time =  0.794999999999983
[Vz] Steady State values =  0.1000000342445711
[psi] Settling time =  0.5099999999999891
[psi] Rise time =  0.4249999999999094
[psi] Steady State values =  14.999999999999153
```
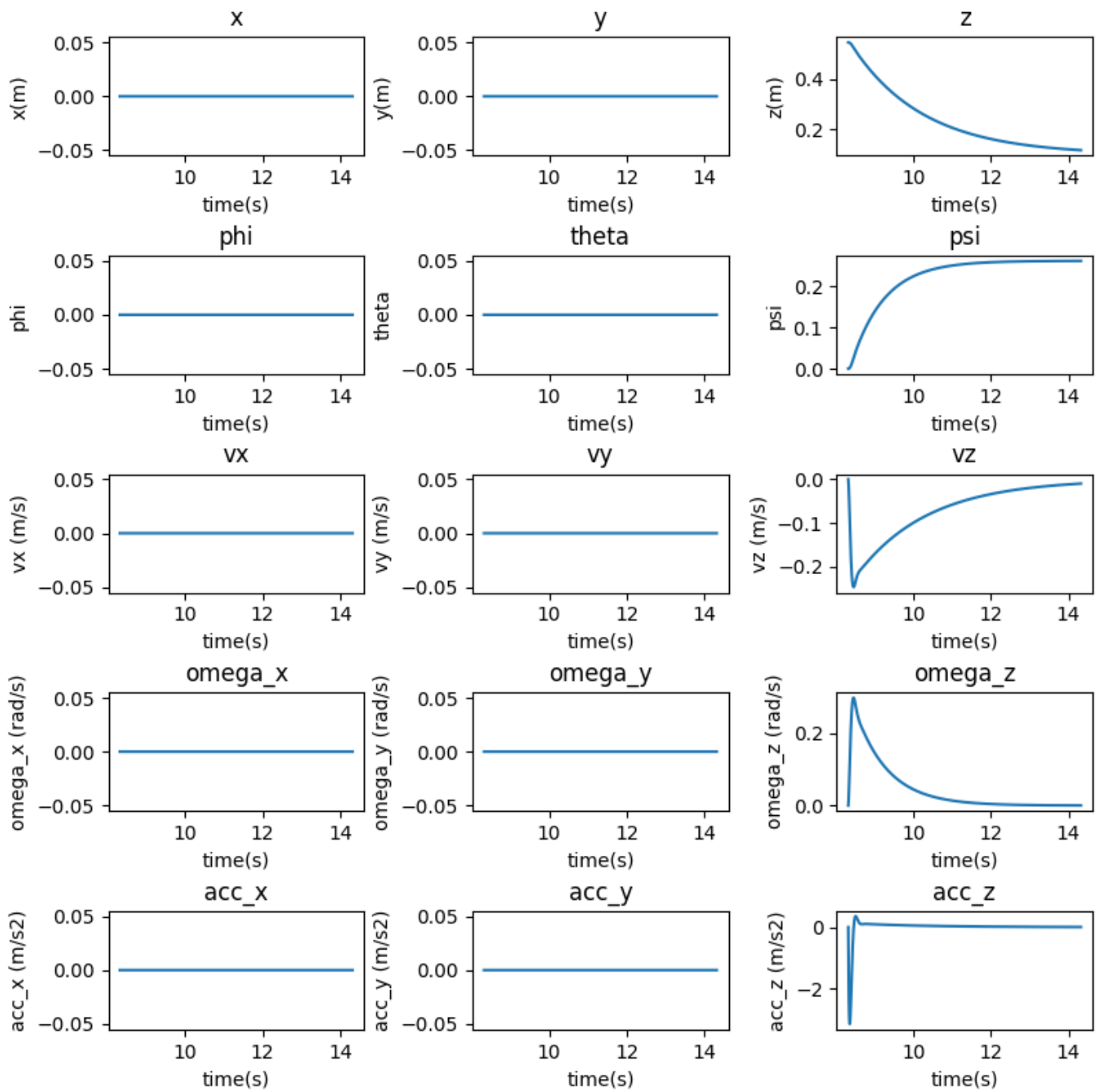
- From the plots it is evident that all the overshoots are 0% as the system is overdamped.

- When including yaw, the settling time increases by an extremely small margin (this can be attributed to numerical precision). This does make sense since the drone is simply altering the moments in the z direction which does not affect the upward thrust. This makes the two motions independent meaning that the settling and rise times will be the same.
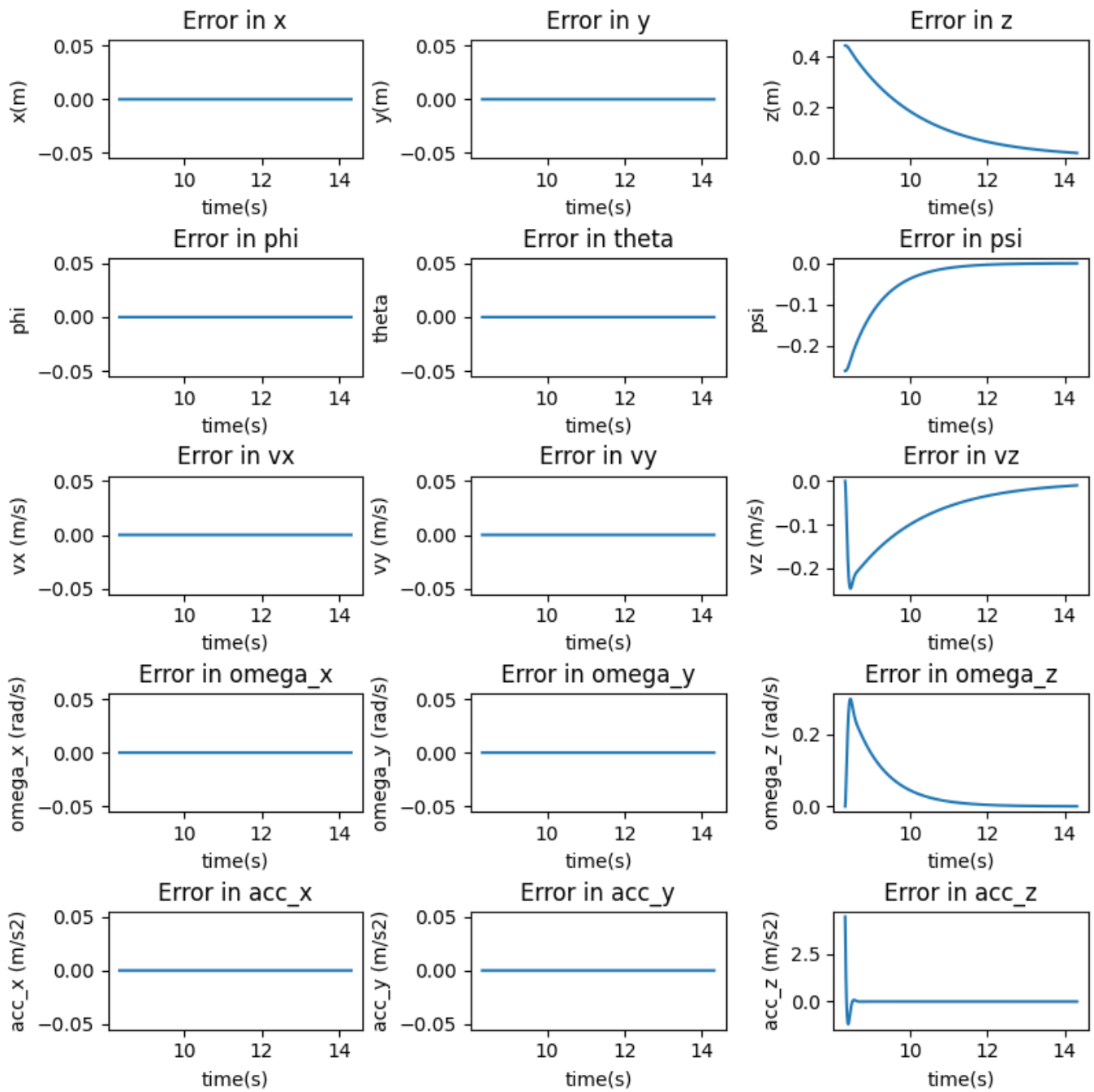
Modifying Gains

```
Gains:
KXatt = [Kp_phi, Kp_theta, Kp_psi]
KXpos = [Kp_x, Kp_y, Kp_z]
params['kpatt'] = [190, 190, 20]
params['kdatt'] = [30, 30, 18]
params['kppos'] = [20, 20, 10]
params['kdpos'] = [8, 8, 19]
```
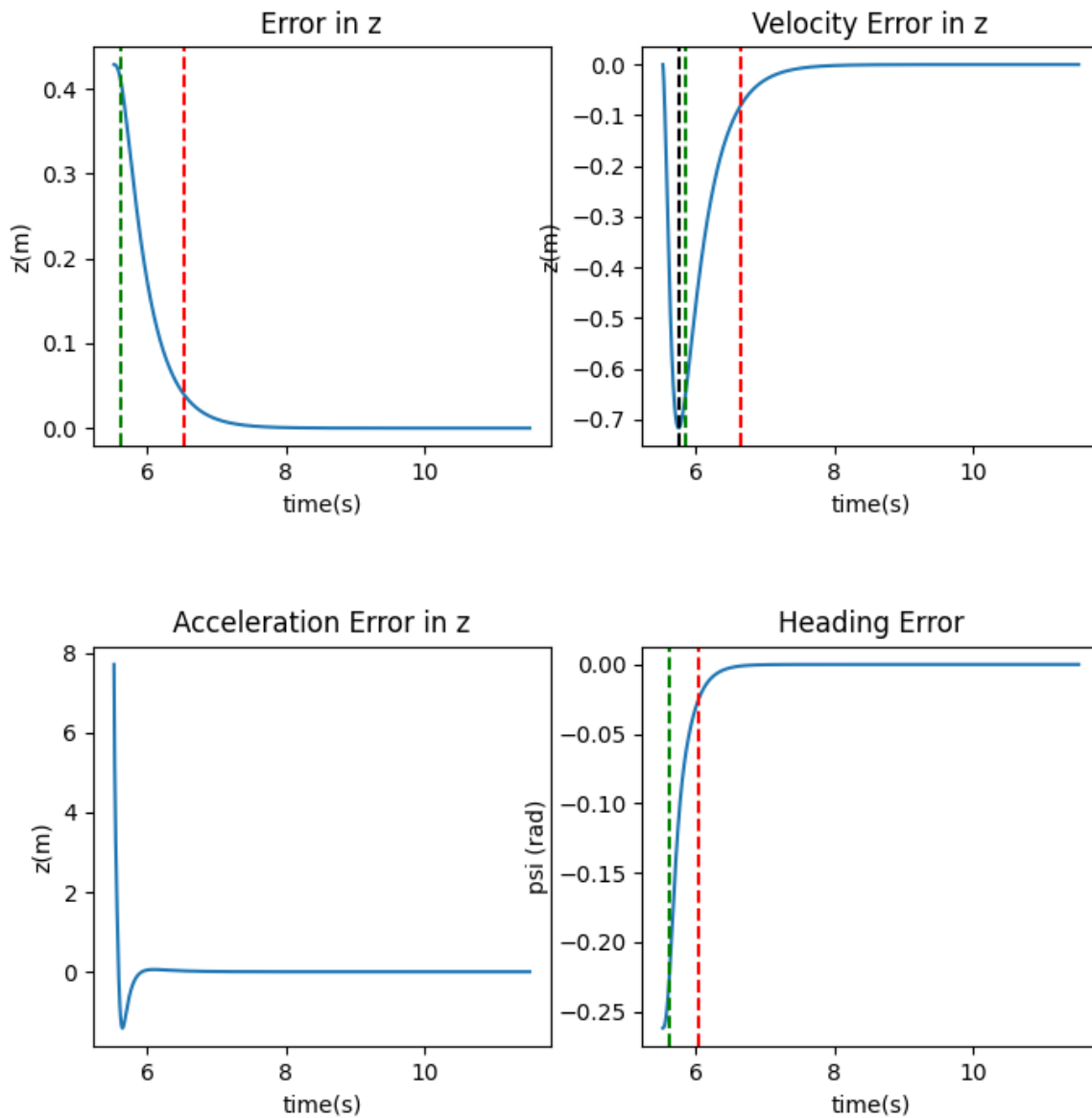
The actual poses of the drone are as follows:

The error plots for the requested scenario are as follows:

Error in x	Error in y	Error in z
Error in phi	Error in theta	Error in psi
Error in vx	Error in vy	Error in vz
Error in omega_x	Error in omega_y	Error in omega_z
Error in acc_x	Error in acc_y	Error in acc_z

The rise and settling times are shown:


```
-------System Statistics--------
[Z] Settling time =  4.495000000000703
[Z] Rise time =  4.300000000000672
[Z] Steady State values =  0.1176625984130827
[Vz] Settling time =  3.4900000000005456
[Vz] Rise time =  3.430000000000536
[Vz] Steady State values =  0.1176625984130827
[psi] Settling time =  1.990000000000311
```

```
[psi] Rise time =   1.8500000000002892
[psi] Steady State values =   14.987203888210878
```

- From the plots it is evident that all the overshoots are 0% as the system is overdamped.

- With the new set of gains, we note that the settling time and rise time has increased drastically for yaw, the z-position and the velocity in the z-direction. This is to be expected as :

  - The Kp for yaw has been reduced significantly causing the drone to respond slower in yaw.
  - The Kp_z has also been decreased significantly causing the drone to respond slowly in velocity and position in the z-direction.

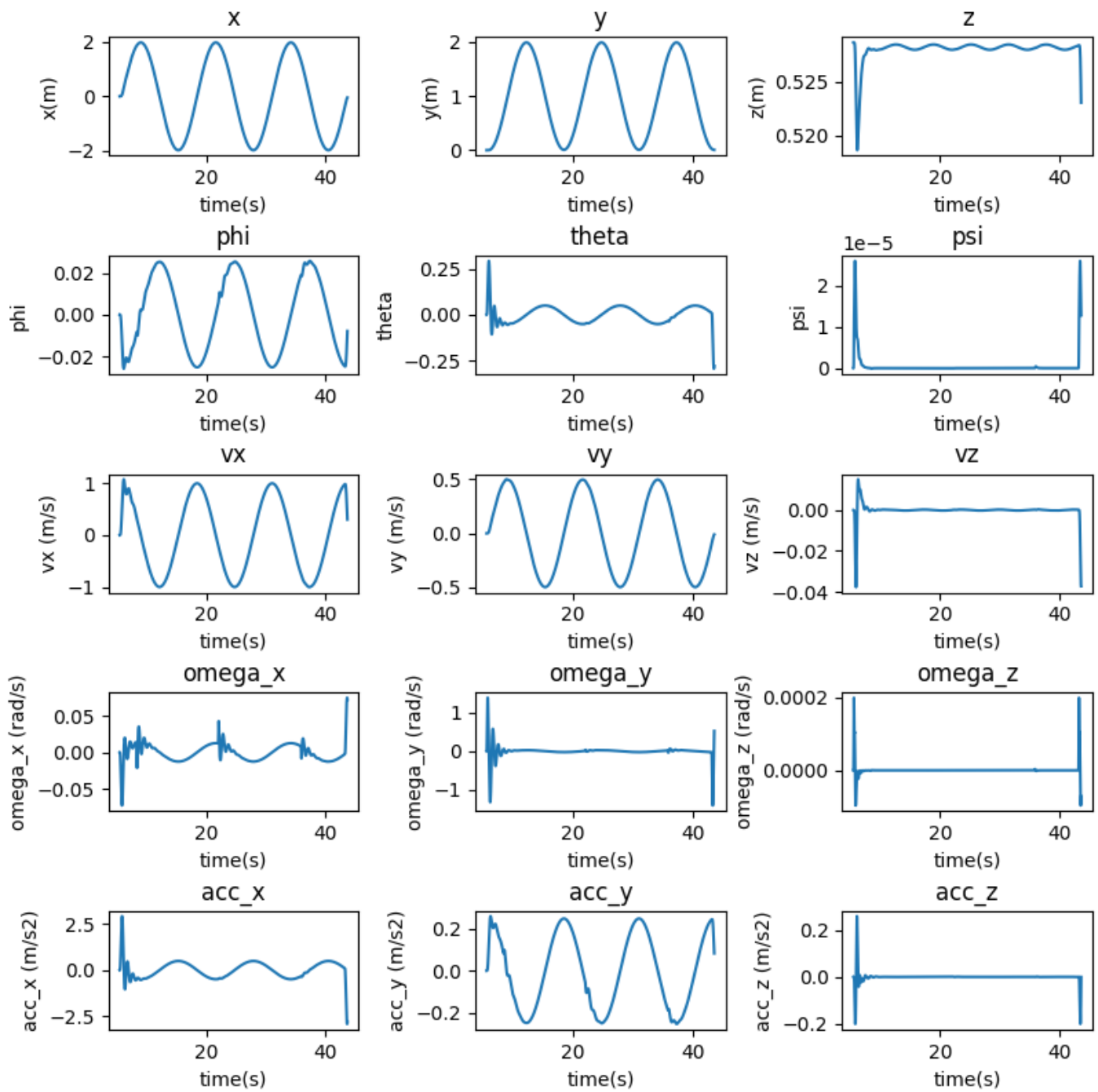Ensuring that the performance is physically plausible using the thrust to weight plot for this scenario:
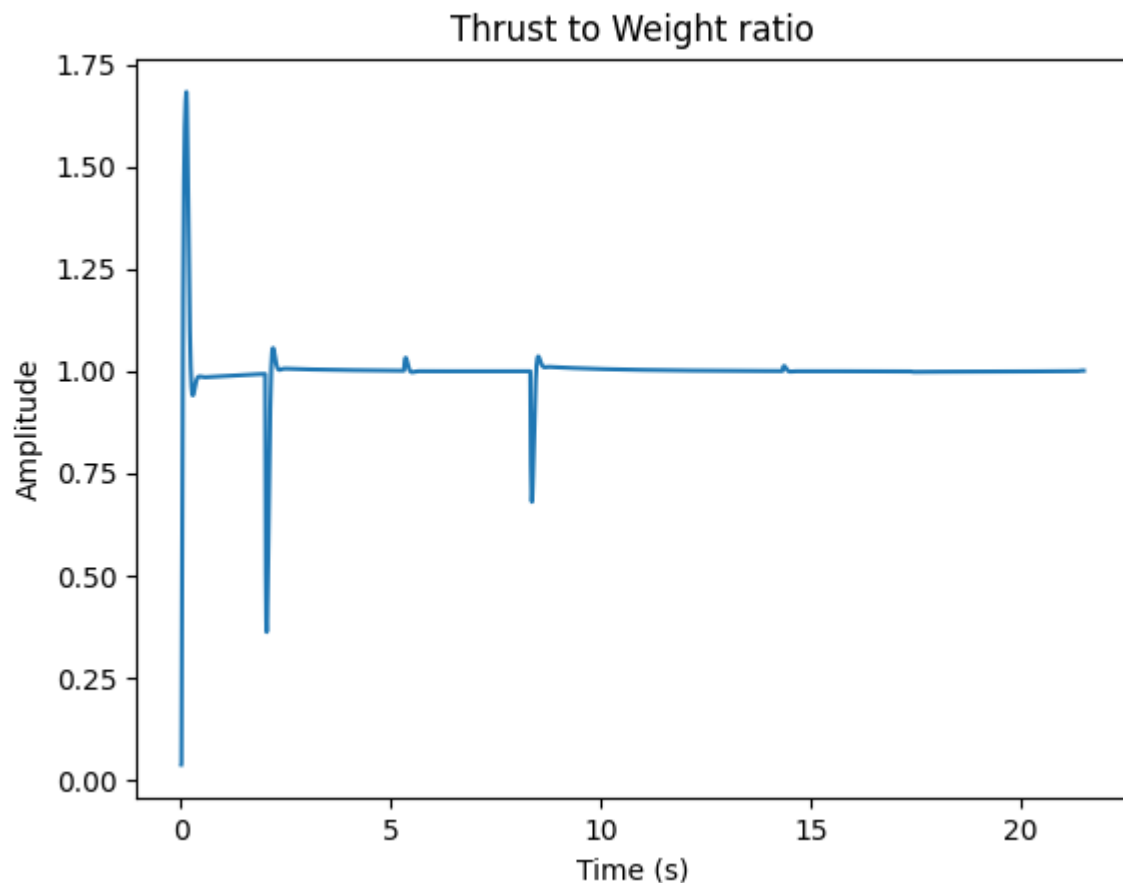
We can see that the performance is well within physical limits.
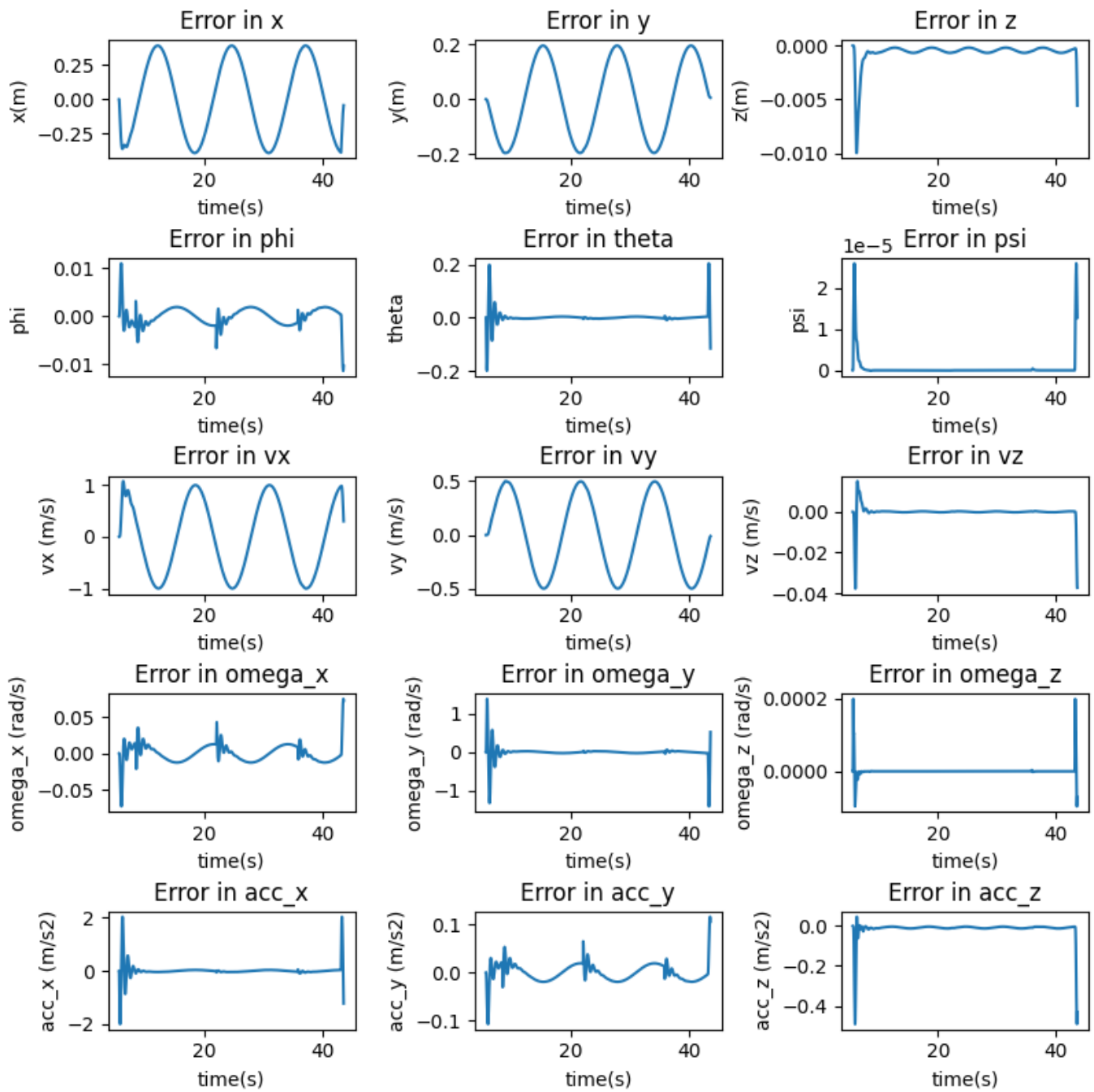
Question 8
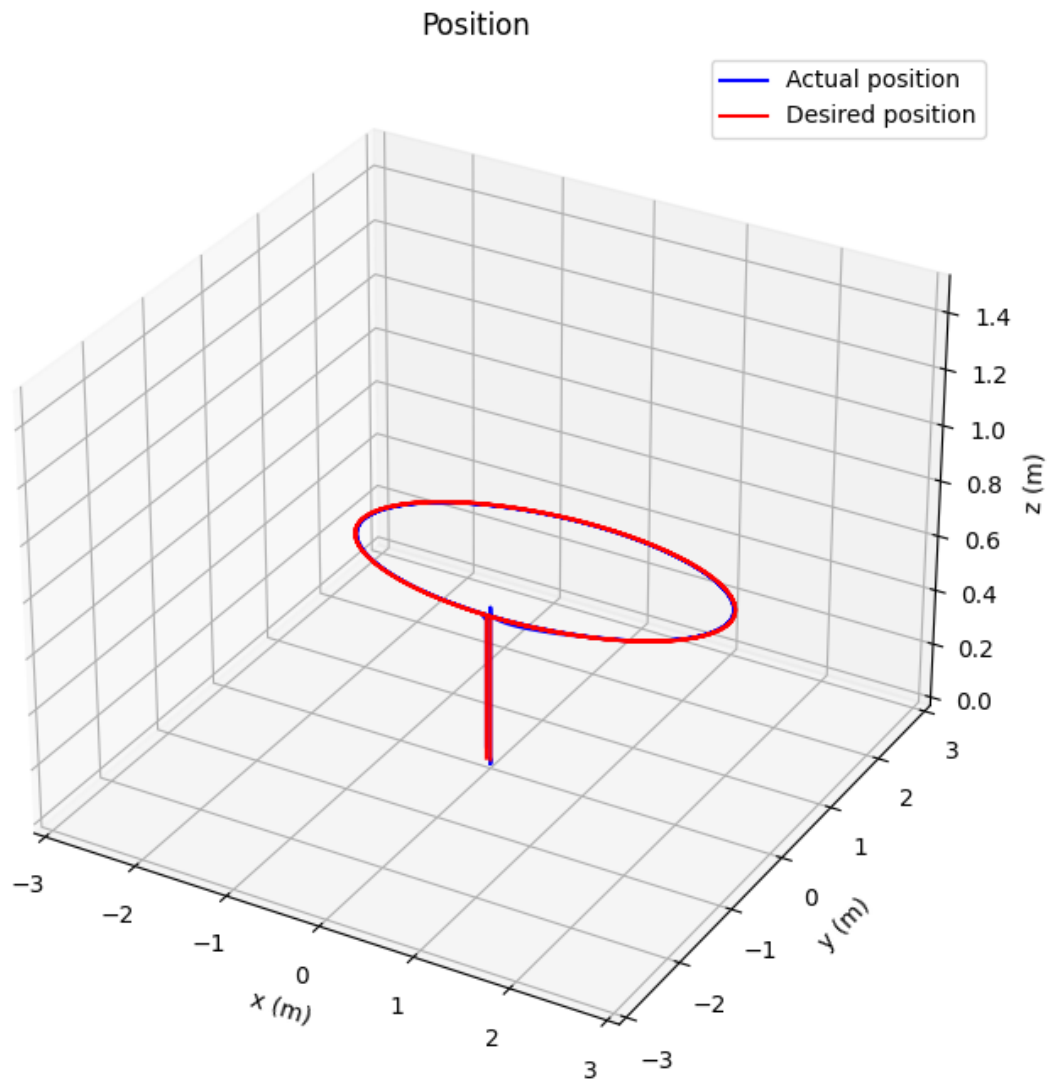
Position

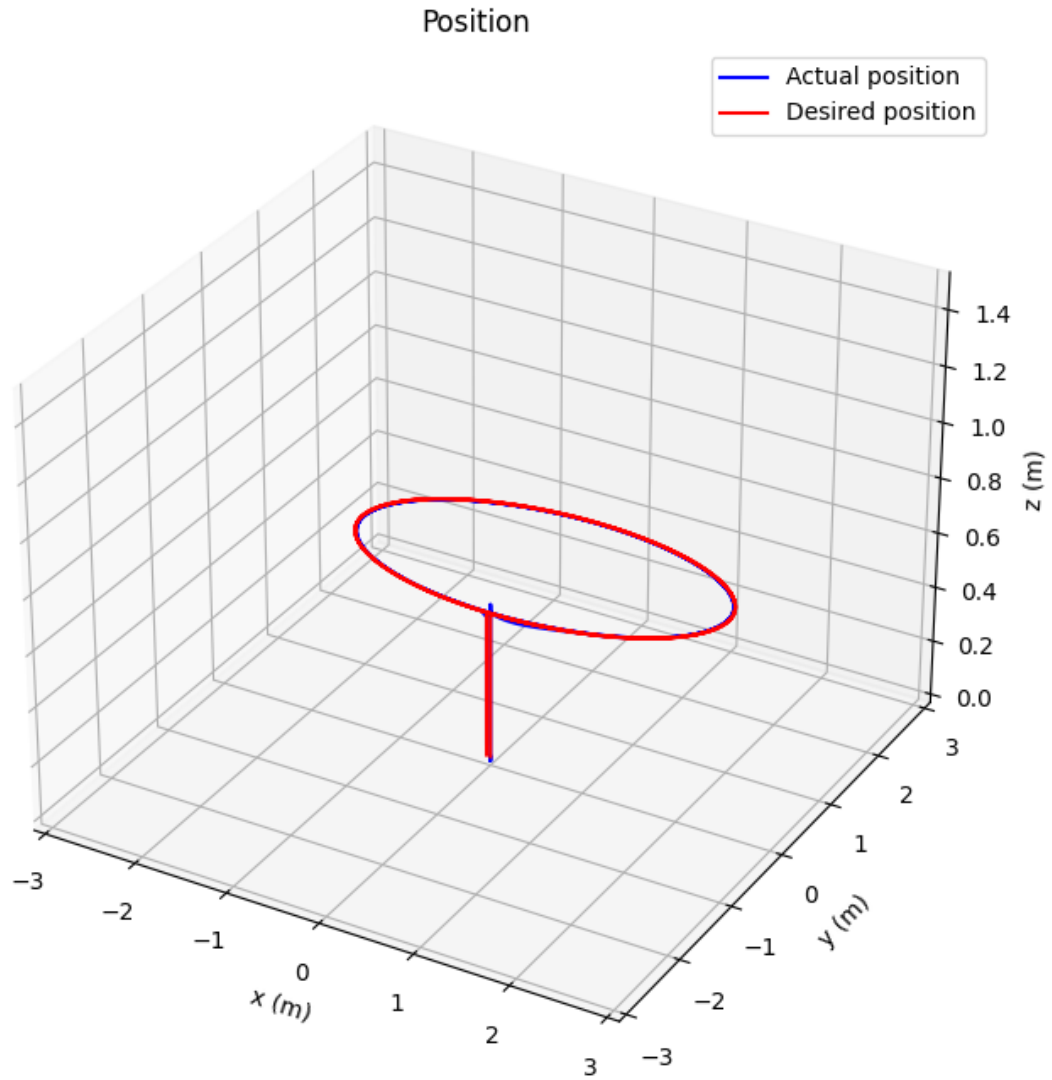As seen in the above graph, the drone is able to track the trajectory appropriately.

The states are plotted here:

The error response is plotted here:

Question 9

Position

As seen in the above graph, the drone is able to track the trajectory appropriately.

The states are plotted here:

The error response is plotted here:
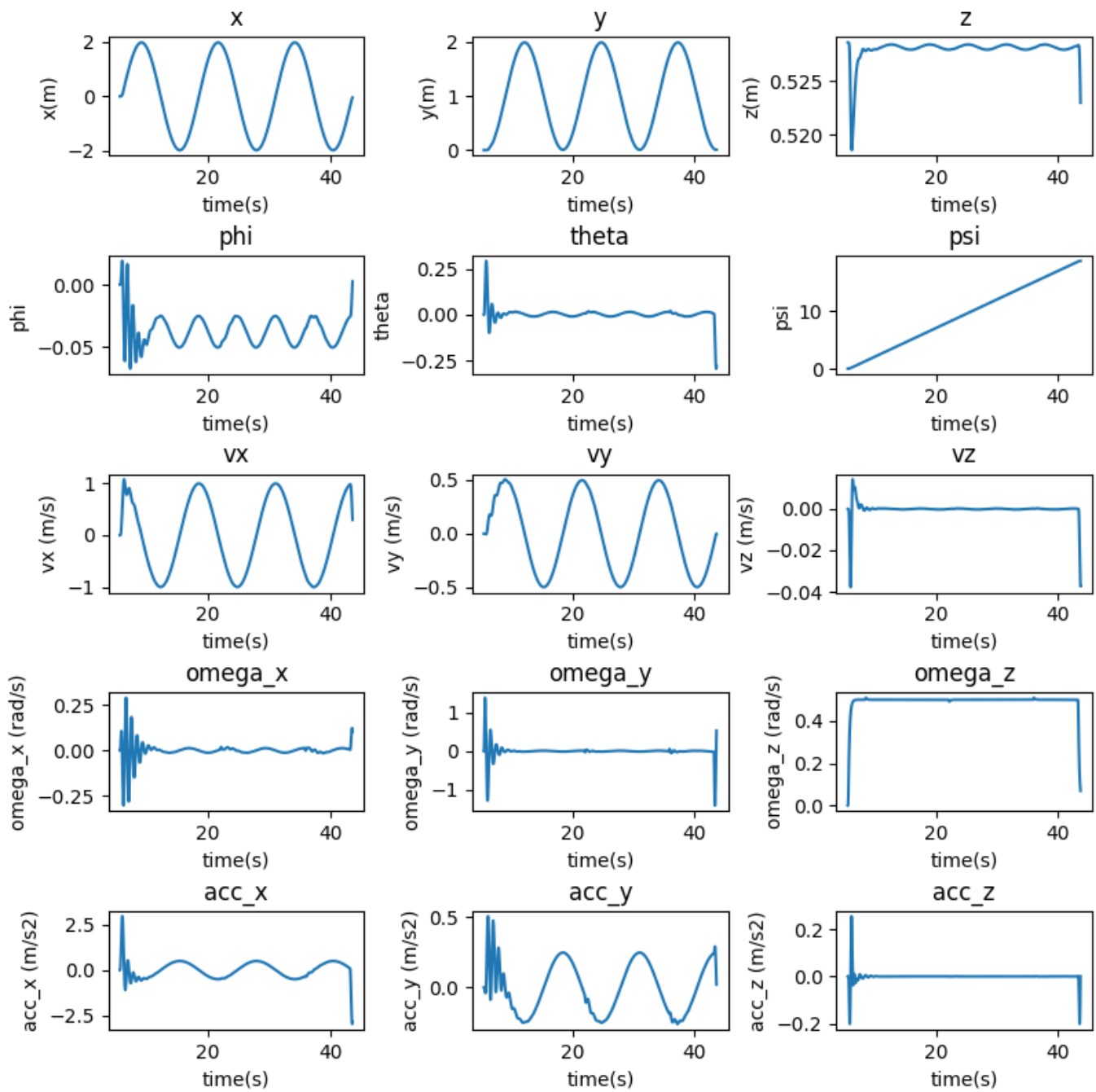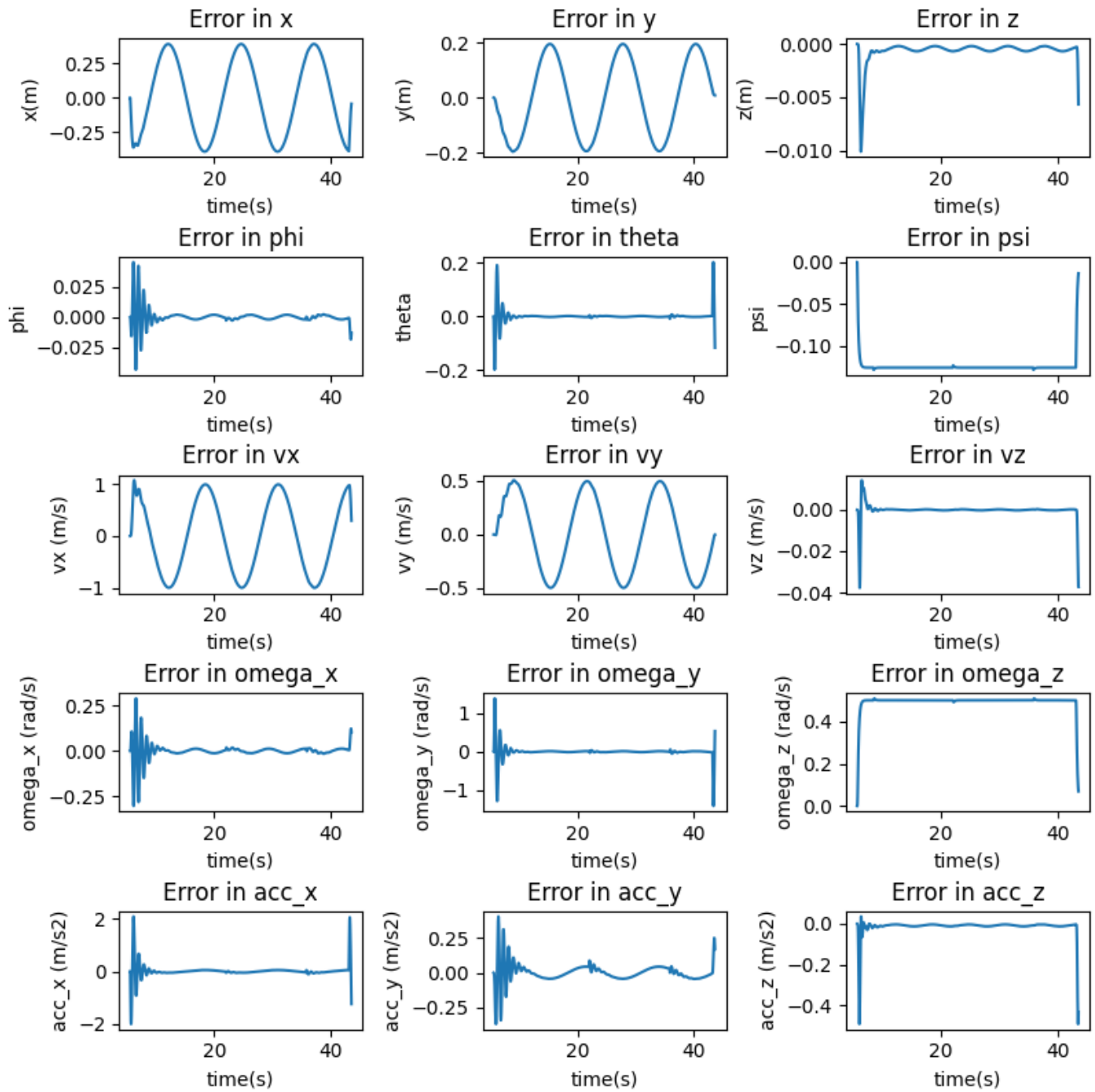
Here, we can see that the performance becomes slightly worse with higher errors in all plots. Although the tracking is still completely successful with the heading changing with time such that the drone always points at the center of the ellipse.