Braden Eichmeier
16-665 – Robot Mobility
November 2, 2019

Air Mobility Assignment

1) System Diagram

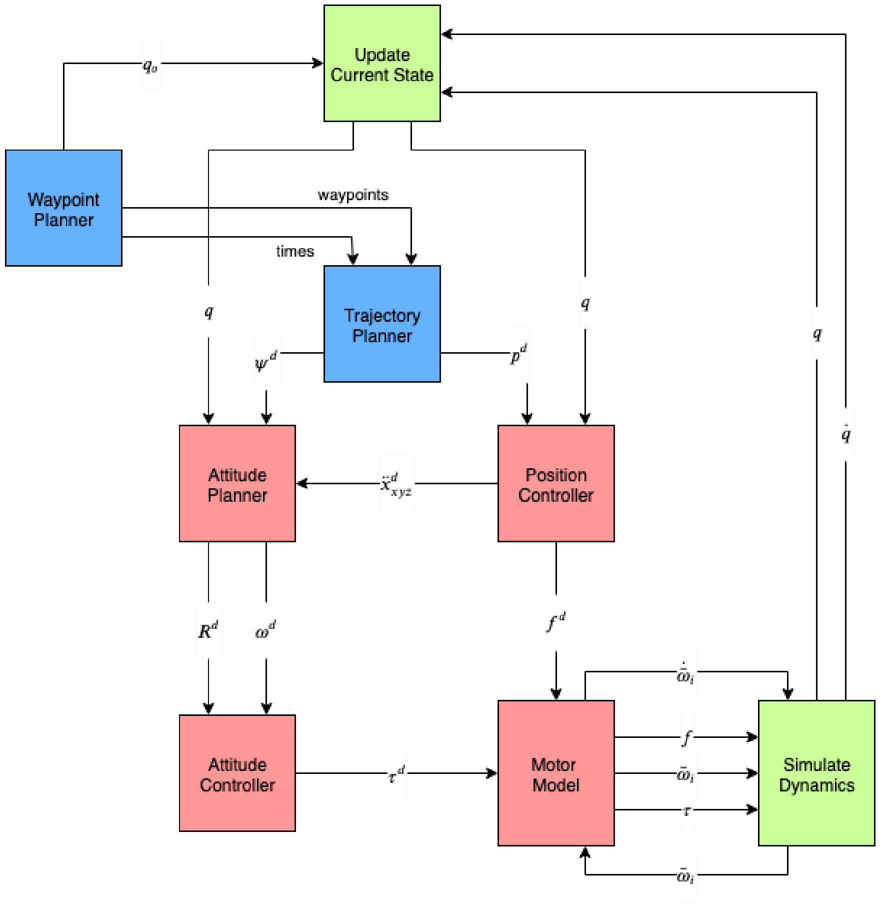


Fig. 1: Software pipeline for the quadrotor simulation.

2) Hover Performance

Figure 2 shows the error associated with simulating the hover scenario detailed in the problem statement. The waypoint changes at seconds 2, 4, 6, and 8 as indicated by the vertical blue lines in the plots. When achieving each new goal, the quadrotor shows a sinusoidal rise. This rise phenomena occurs due to the oscillations in achieving the desired attitude in the inner loop. The quadrotor achieves the new waypoint approximately one second after receiving the goal and oscillates around the point until the new goal is received.

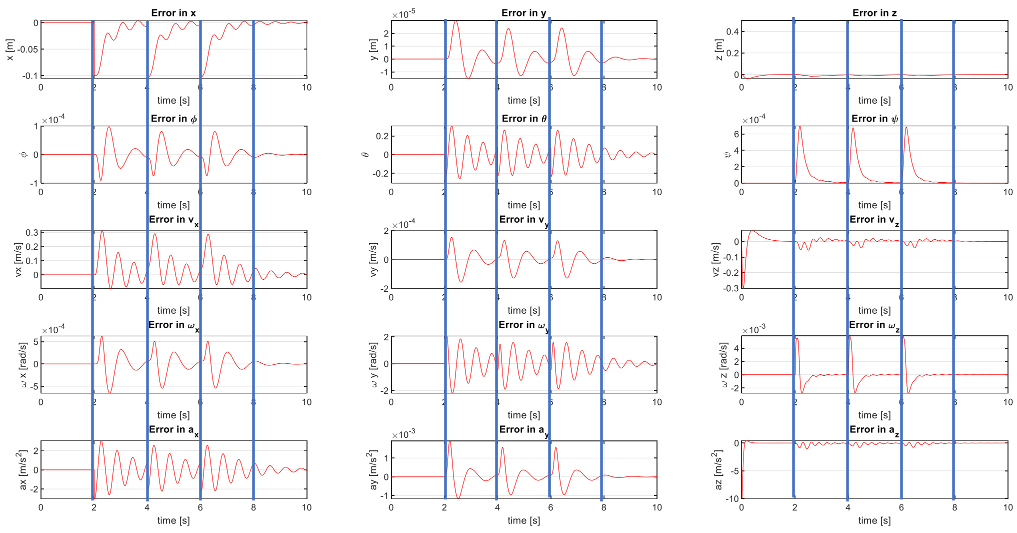


Fig. 2: Error plots showing the position and orientation errors for hover.

Adjusting the proportional and derivative gains alters the convergence and oscillation of the system. Increasing the proportional gain makes the system react more aggressively, which can lead to divergence if Kp is increased too much. Increasing the derivative gain dampens the oscillations in the system. Dividing the proportional gain by 1.3 resulted in a more smooth response as shown in Fig. 3.

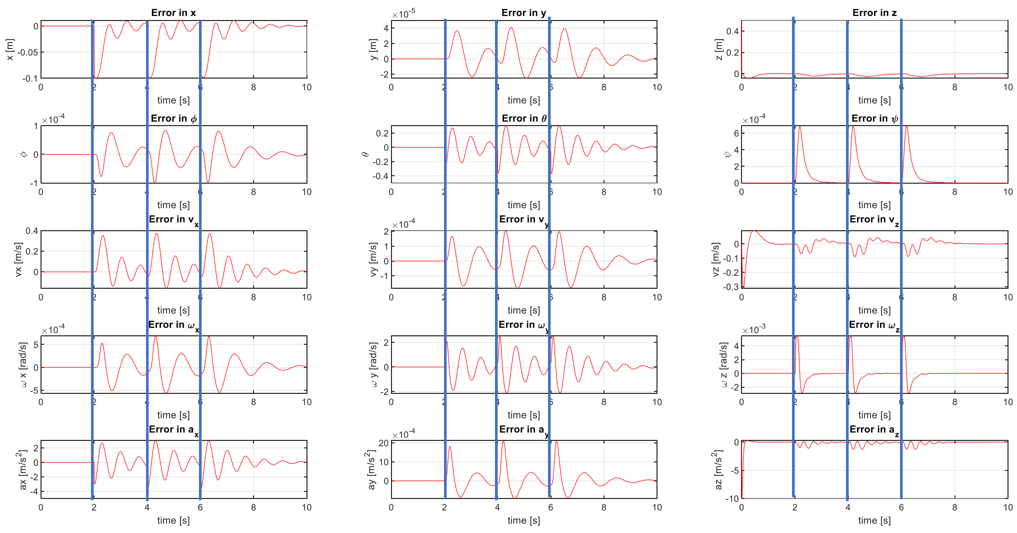


Fig. 3: Reducing the derivative gain causes the system to better converge to the waypoints.

Modifying the gains for the attitude control behaves similarly with adapting to the attitude commands. Modifying the gains showed a good convergence occurs when increasing both the proportional and derivative gains. This is logical because it makes the inner loop respond faster than the outer loop and reduce dynamic coupling. Figure 4 shows the error results from multiplying the inner loop proportional and derivative gains by 2 and 1.5 respectively.

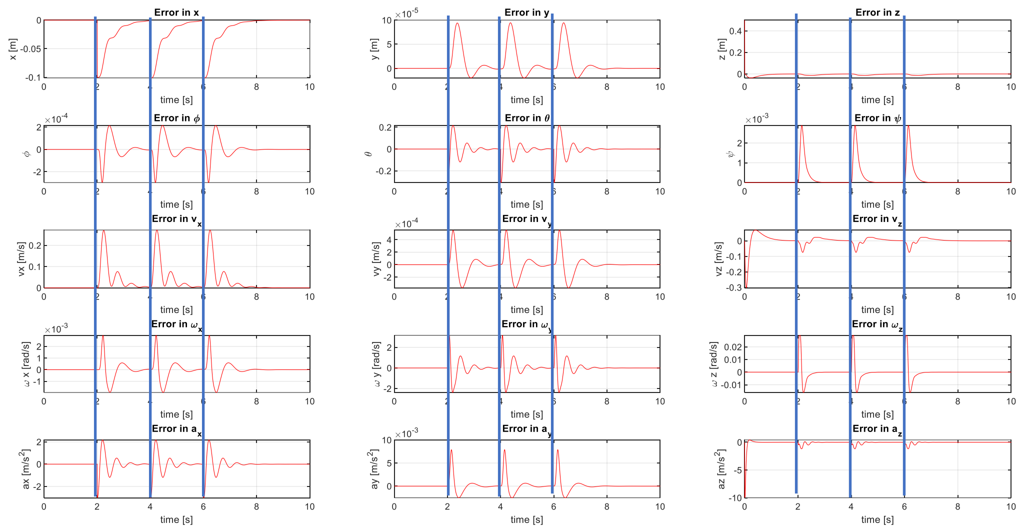


Fig. 4: Increasing the inner loop gains gives quick attitude convergence and smooth position convergence.

3) Line-tracking Performance

Figures 5 and 6 show the position and error plots for the tracking controller. The top-right figure in Fig. 5 shows the system performs well win achieving the desired states without oscillations. However, there is a small delay in achieving the desired states when beginning takeoff or descent.

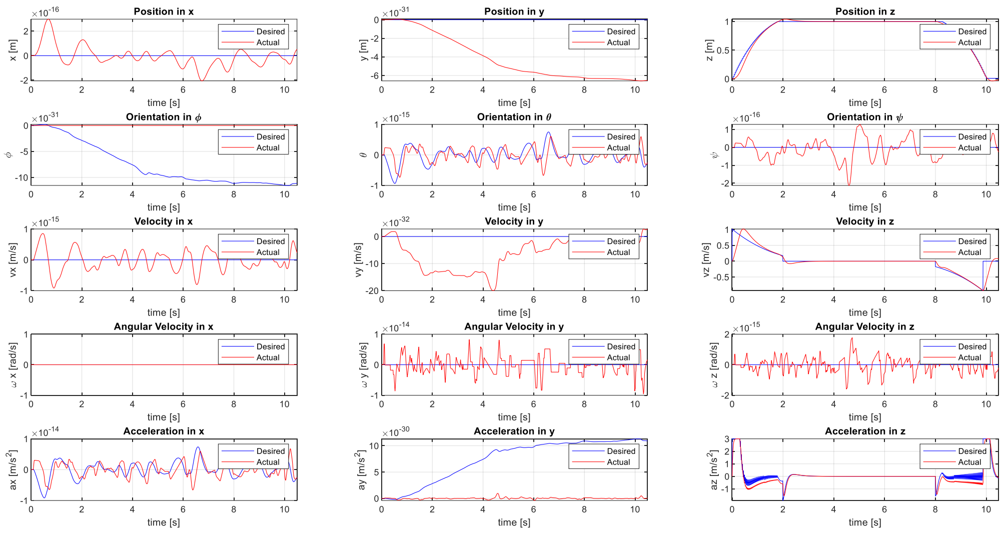


Fig. 5: Position plots for the line tracking behavior in response to a takeoff, hover, and landing sequence.

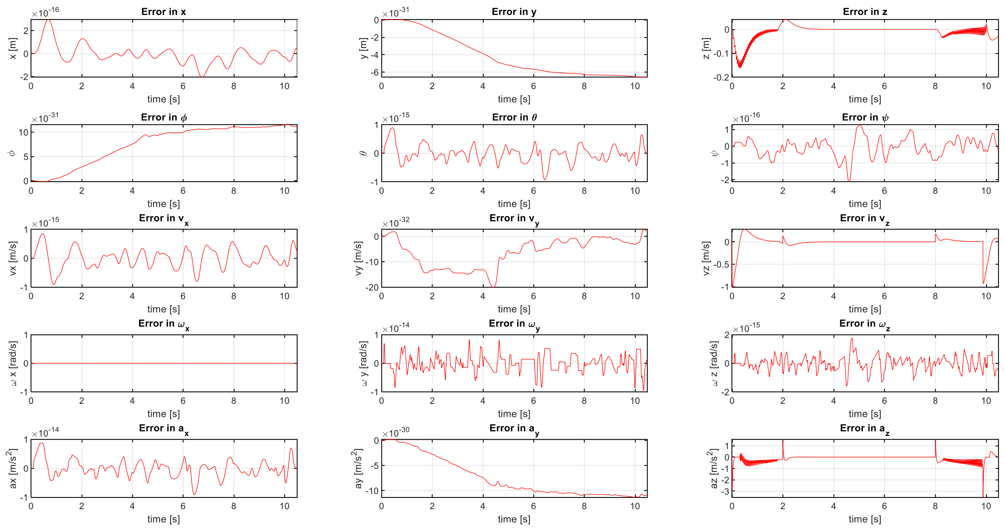


Fig. 6: Error plots for the takeoff, hover, and landing sequence shown in Fig. 5.

Modifying the gains to the system did not significantly alter the plots due to the already low error. Figures 7 and 8 show updated trajectory metrics with the proportional gains tripled compared to the previous figures. The largest difference is an increased delay in responding to the ascent phase.

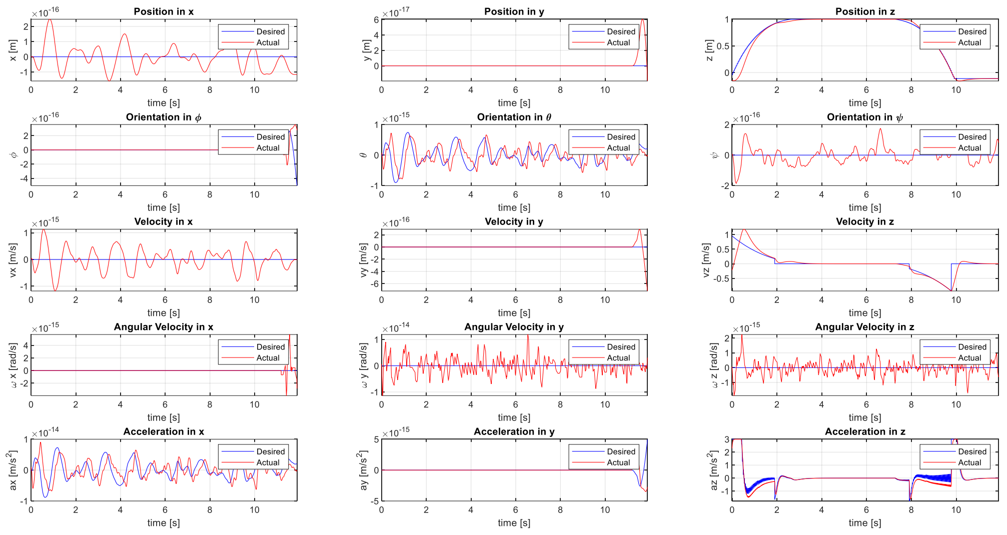


Fig. 7: The position tracking with increased outer loop gains shows little improvement in convergence.

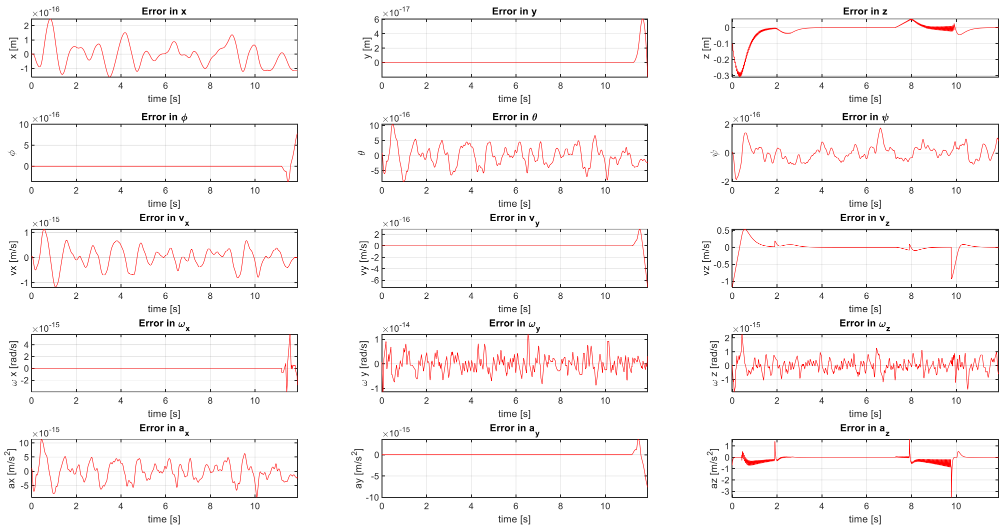


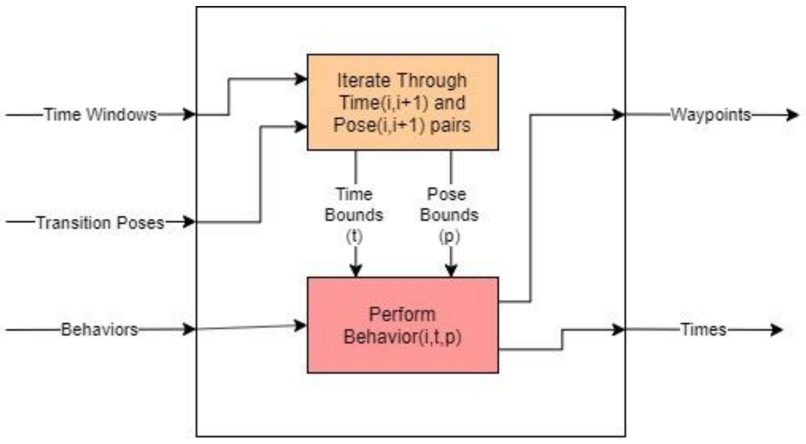
Fig. 8: The z-position error shows increased noise and oscillation with the increased position gain.

4) State Machine

Figure 9 illustrates the state machine I implemented in the code. There are three inputs to this state machine. The first is an array of times to dictate the start and end times for each behavior. The second input is an array of poses to define the start and end positions of each behavior. The final input is an array of the desired behaviors for the simulation. The time and pose arrays should have one more entry than the behavior array, which corresponds to each behavior having a start and end state. This state machine iterates through behaviors based on the time bounds. Once the end time has been reached, the machine iterates to the next behavior.

Each behavior is defined by a unique function that precomputes the desired positions for each moment in time. The behaviors are defined to operate in a single unit time (1 second), then the state machine scales the time stamps from the behavior function to match the time bounds from the time array. This method makes the behaviors function independent of time scaling. Similar scaling is performed for the magnitude of the behaviors using the input poses.

For this assignment, I created several behavior functions: hover, ascend, descend, ellipse, and pirouette. I also created a second implementation of this state machine for trajectory_planner.m to precompute the function derivatives in a similar manner. Figure 9 also shows the code to implement Question 3 using this state machine.



```
other = [1,1,1,1];
times = [0,2,4,8,10];
behavior = {@hover; @takeoff; @hover; @descend};
pose1 = [0,0,0,0];
pose2 = [0,0,0,0];
pose3 = [0 0 1 0];
pose4 = [0, 0, 1, 0];
pose5 = [0 0 0 0];
poses = [pose1; pose2; pose3; pose4; pose5];
[points, times] = st_machine(behavior, poses, times, other);
```

Fig. 9: Block diagram and example code for the state machine implemented to precompute behavior trajectories.

5) Gain Selection and Tuning

Table 1 shows the system's response to a step input in both height and heading. The height step response and error are shown in more detail in Figs. 10 and 11 respectively. The height and heading responses and errors are shown in Figs. 12 and 13.

Table 1: Quadrotor system response to a step input for the height and heading

Value	M_P (%)	t_r (s)	t_s (s)	y_{ss}
Height	7.5	0.215	0.3	0.1 m
Heading	0	0.38	0.44	15°

Modifying the position controller gains shows the system is stable for $K_P = [40{,}90]$ and $K_d = [4{,}10]$. The system allows higher K_d with higher K_P values. Increasing the proportional gain reduces the response time but increases overshoot and ringing. Increasing the derivative gain substantially reduces the overshoot and settling time while having little effect on the rise time. Proportional and derivative gains of 65 and 11.75 result in a quick system with little overshoot.

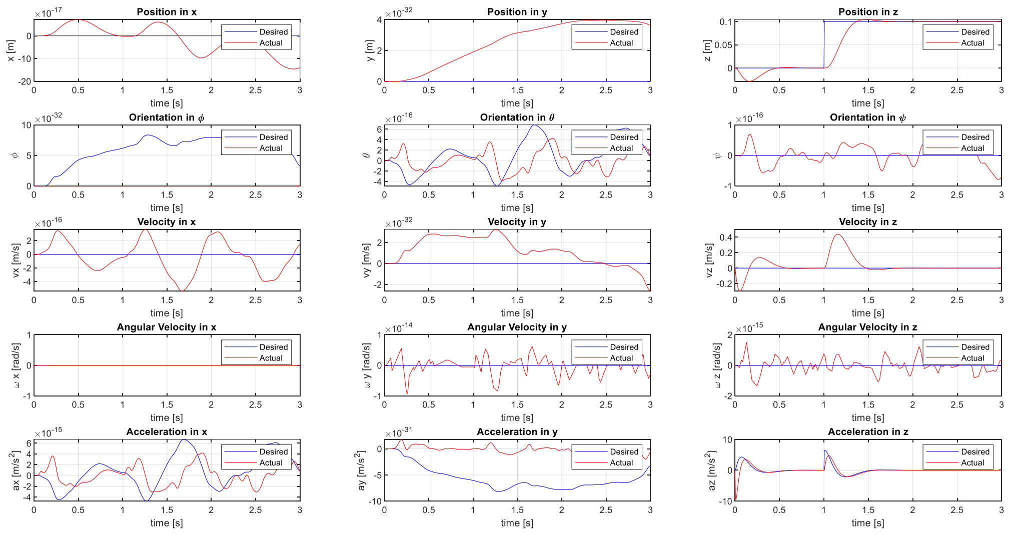


Fig. 10: State plots in response to a 0.1 m step input using tuned gain parameters.

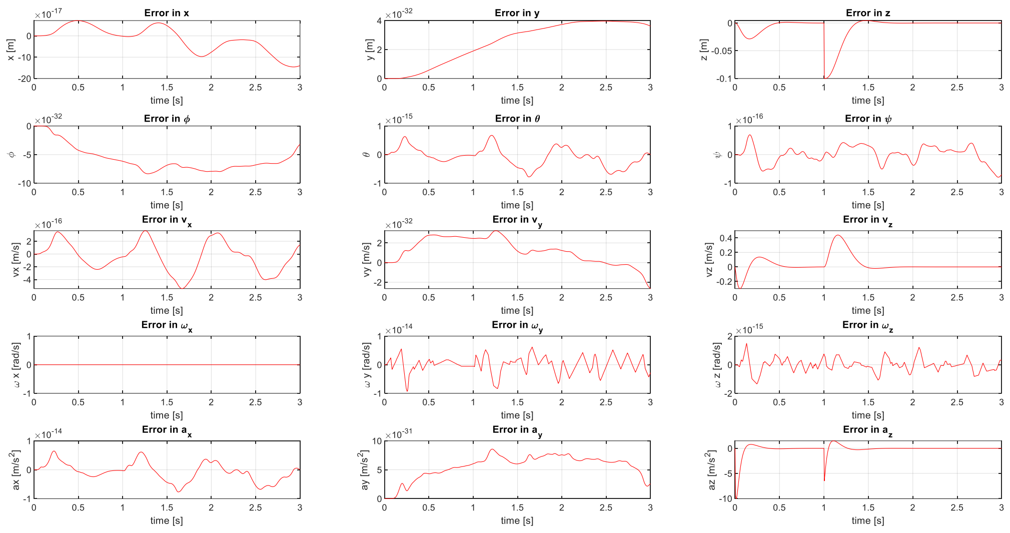


Fig. 11: Error plots in response to a 0.1 m step input using tuned gain parameters.

The attitude controller gains behave differently than the position controller. For good behavior, the system should respond faster than the position controller. The stable range for the position controller is $K_P = [40,160]$. For the derivative gain, I could only find a minimum for stability of 18. The highest K_P results in an overdamped system, thus a small K_d will produce the most responsive system. The optimal gains for K_P and K_d are 160 and 18 respectively.

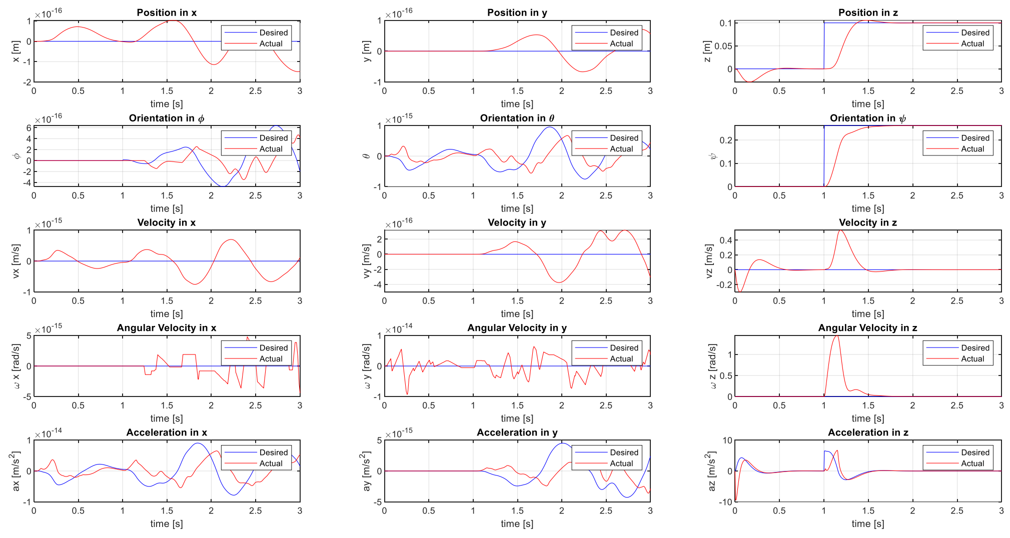


Fig. 12: State plots in response to a 15° step input using tuned gain parameters.

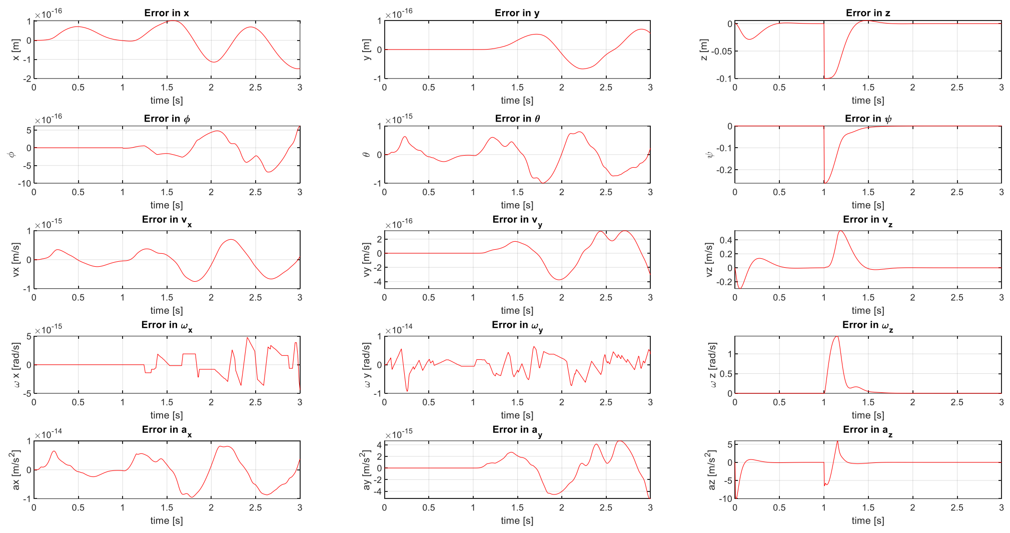


Fig. 13: Error plots in response to a 15° step input using tuned gain parameters.

6) LQR Controller design and Evaluation

a) Hover Performance

Using an LQR controller on the hover performance, I found the system converges to the desired positions with little overshoot. Once attaining the desired values, the systems oscillates a miniscule amount around the goal state. I found rapid convergence by increasing the z-value. Figures 14 and 15 show the position and error plots for the system with the Q-z value at 13 and the Q-x value of 1. All other Q values are set to 0.2 and the R values are set to 1.

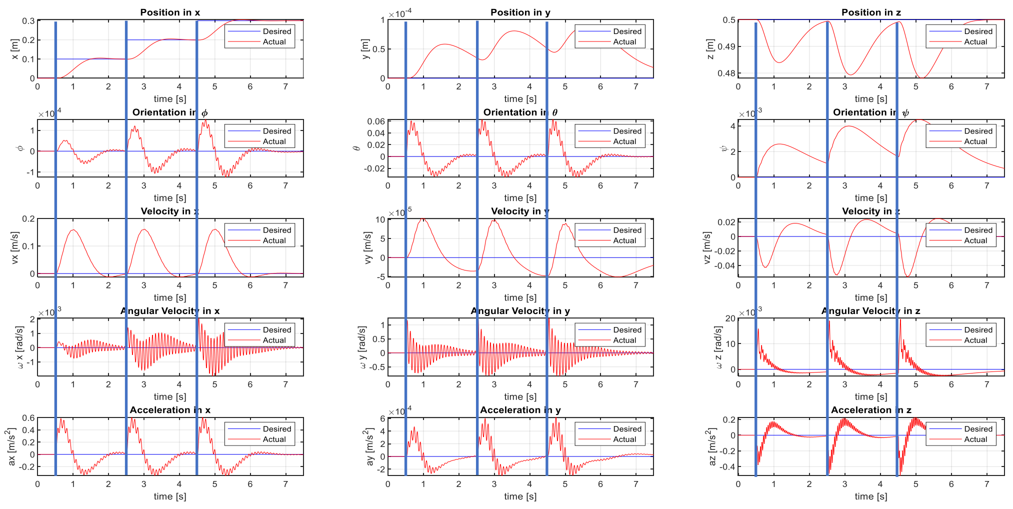


Fig. 14: The LQR controller shows good convergence in both the x and z positions with little tuning.

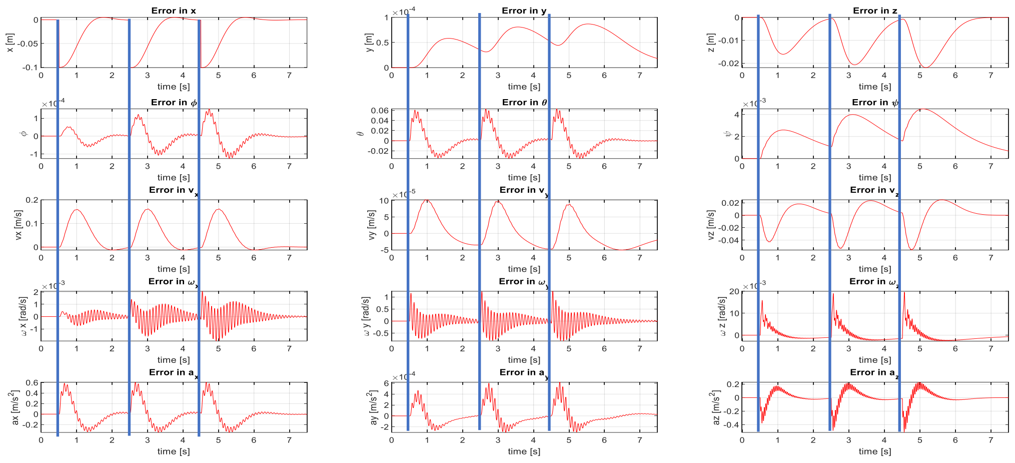


Fig. 15: The error plots show significant oscillations in the acceleration plots and smooth convergence in the positions.

The system shows little change with alterations to the Q-z value. Increasing the value leads to slightly quicker convergence to the goal state. Increasing the Q-x value drastically quickens the x-position convergence. Doing so causes the z-position to have a steady state error of about 20%. Halving the Q-x value similarly delays the system's convergence to the goal positions. The delayed convergence also reduces the maximum z-error to within 0.5 mm. Figures 16 – 19 show the position and error plots for the modified Q matrix parameters.

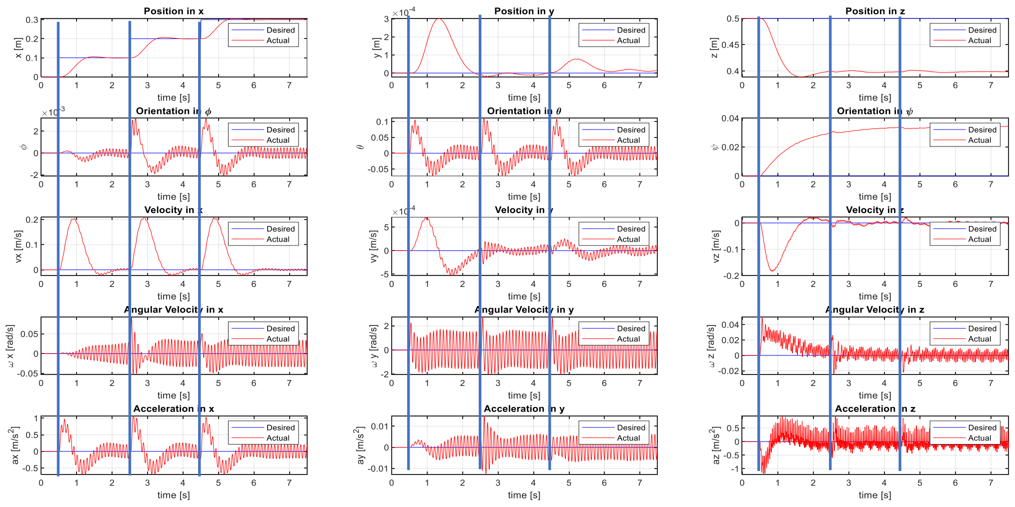


Fig. 16: Increasing the Q-x values by about 33%. The z-position error is magnified without steady state convergence.

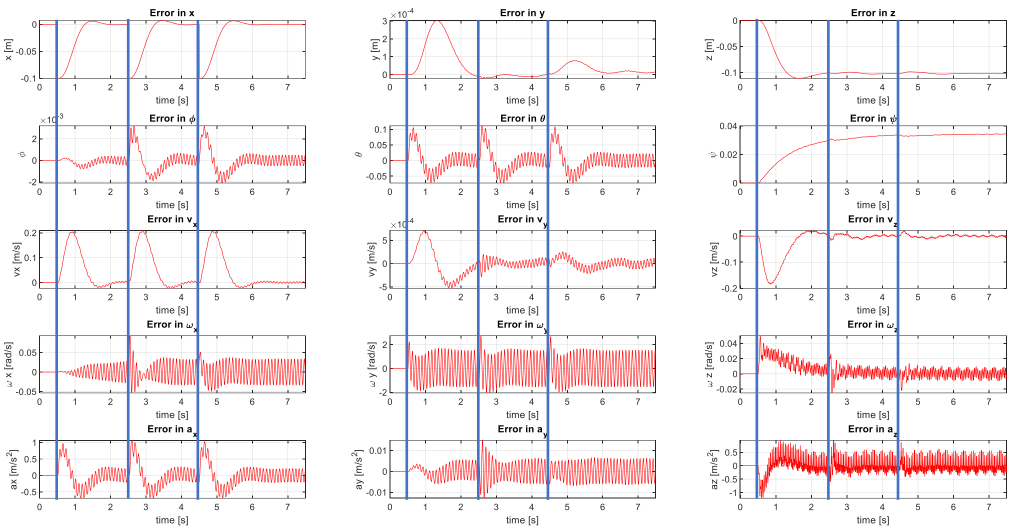


Fig. 17: The increased Q-x value amplifies the oscillations in nearly all the position plots.

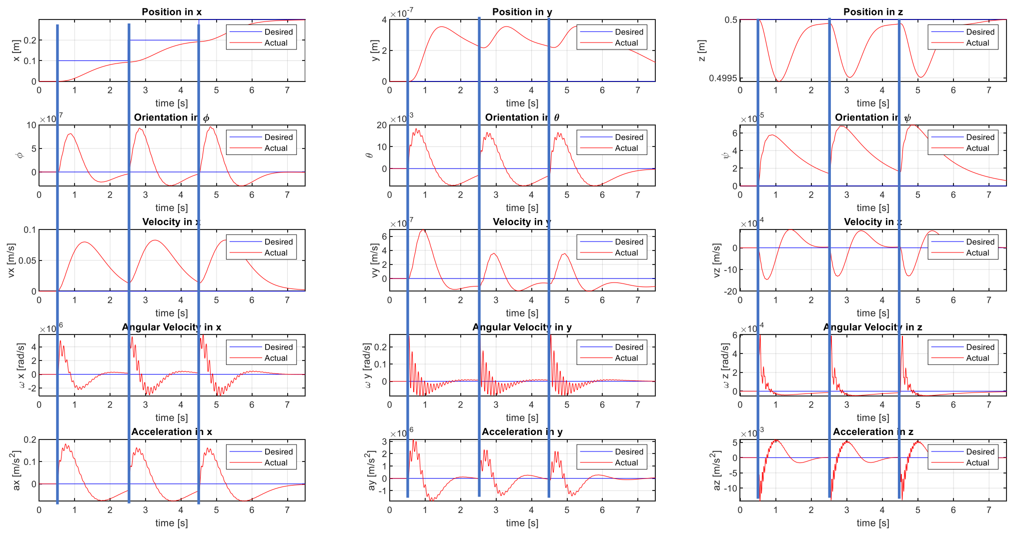


Fig. 18: Reducing the Q-x value by half causes the system to barely converge to the goal before iterating to the next goal point.

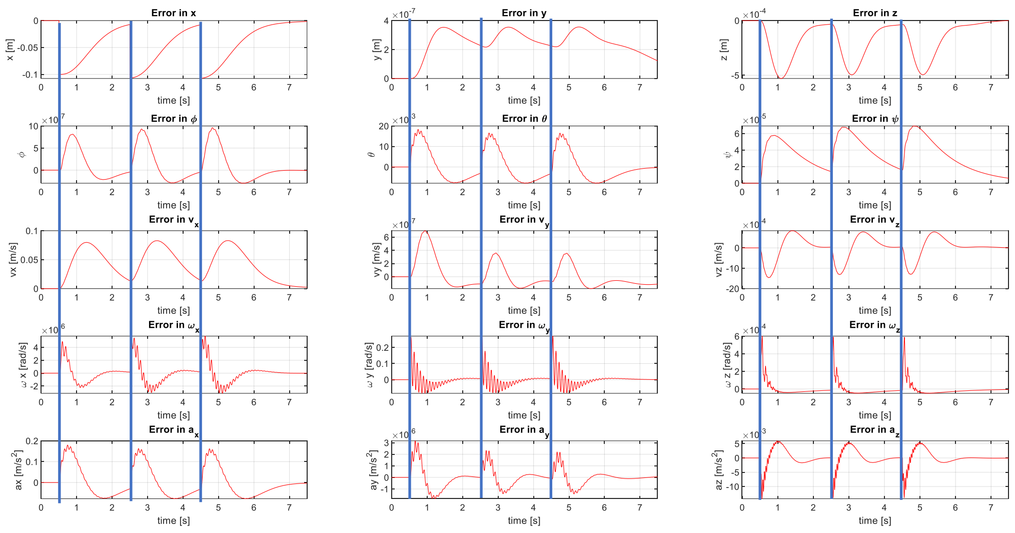


Fig. 19: The error plots show the miniscule error in each of the states.

The LQR controller behaved quite differently to the PD controller. The PD controller had a slow, oscillatory behavior to the goal states. An interesting behavior occurs in the angular and acceleration states using the LQR controller. There are significant oscillations as a unique controller is computed at each time step. The system shows better convergence with the LQR controller in response to a step input in hover behavior.

b) Line-tracking Performance

The LQR controller described in section (a) behaves similarly to the PD controller described in Question 3. The LQR controller lags the desired path for about 1 second before it overshoots the desired trajectory. At 2.5 seconds the system reconverges to the path with negligible oscillations. In the descent flight, the system experiences a slight (5 cm maximum) lag with the desired path. There is no overshoot when converging to the descent path or the hover after descent.

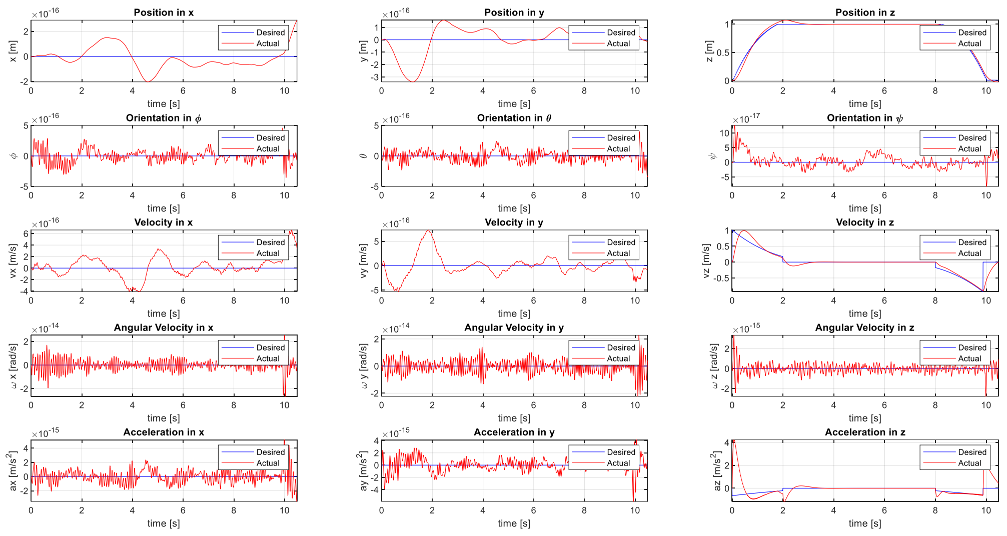


Fig. 20: The z-position tracking shows similar behavior for both the PD and LQR controllers.

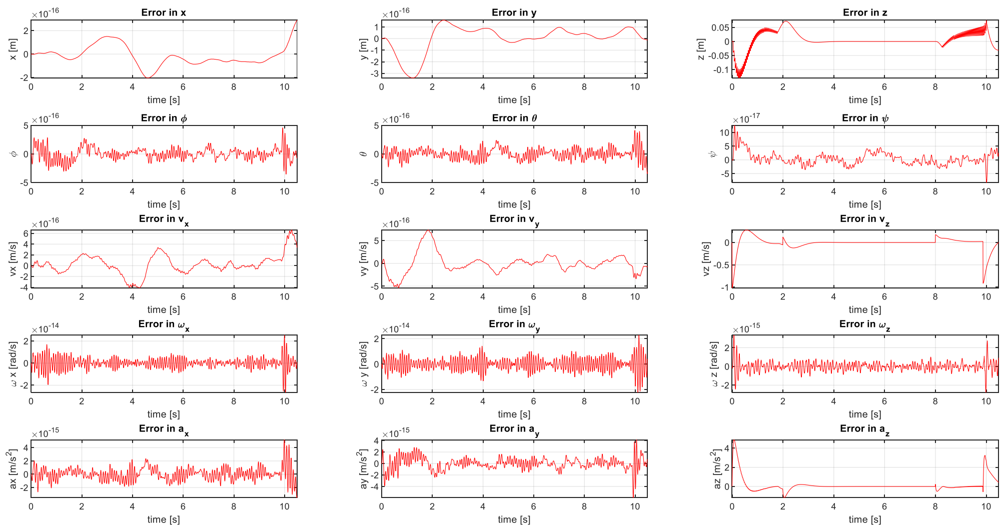


Fig. 21: Maximum error for the described LQR controller does not exceed 10 cm, and occurs directly after takeoff.

Modifying the Q-z value shows direct effects on the system's convergence. Increasing the Q-z value results in lower error. Despite this, the overshoot at the end of the ascent phase does not change much by increasing the Q-z value. This likely occurs due to a non-flat commanded trajectory. Lowering the Q-z value increases the system's lag and overshoot. Figures 22 and 23 show the system behavior with Q-z reduced to 5.

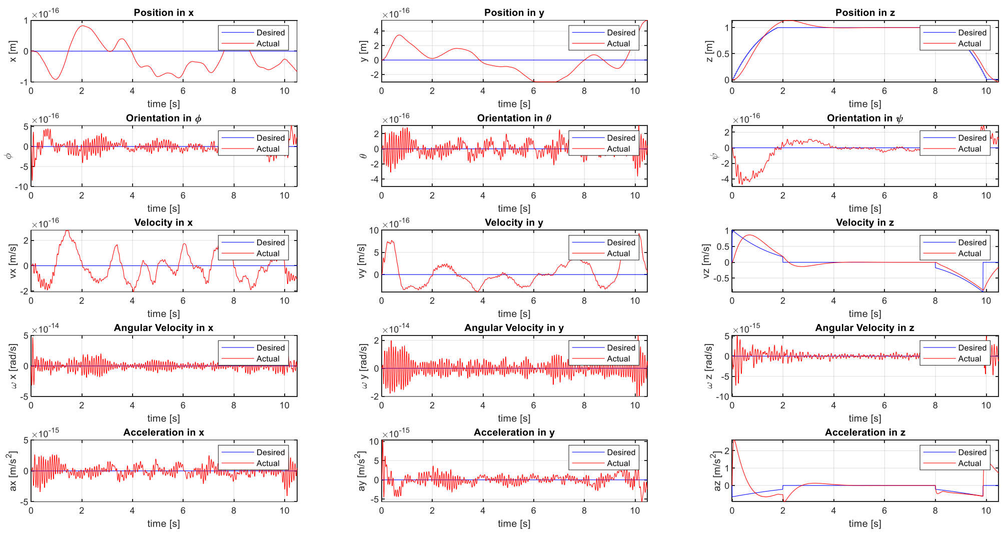


Fig. 22: Reducing the Q-z value causes the system to experience a single overshoot of 18%.

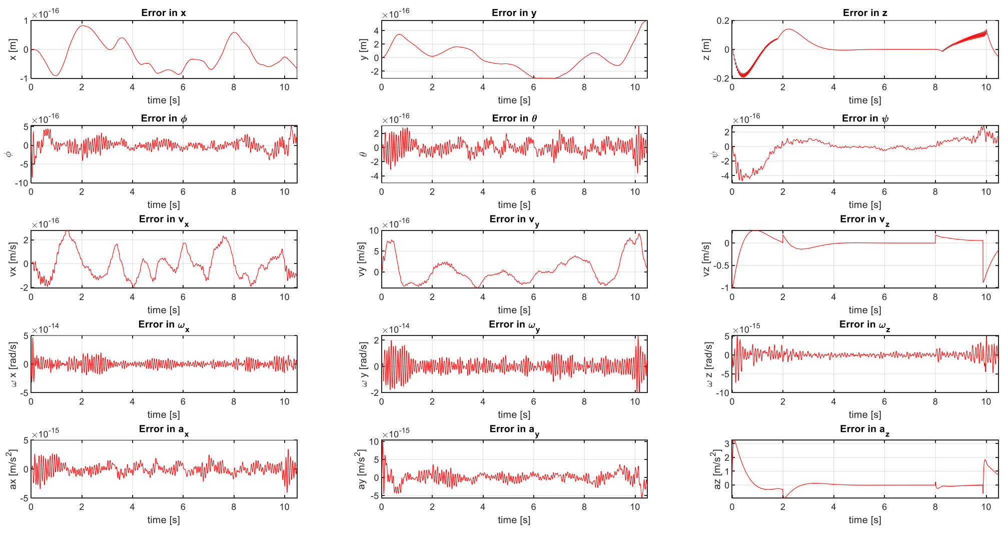


Fig. 23: Significant error in the relaxed LQR controller only occur in the z-position and derivative values.

When compared to the PD controller, the LQR controller behaves and tunes similarly. The PD controller does allow greater flexibility in tuning the overshoot behavior of the system. On the other hand, tuning the LQR controller required much less time to achieve a well performing system.

c) Gain Selection and Tuning

Table 2 shows the system response to a step input in both height and heading. The LQR controller shows about a 20% slower rise time and settling time to the step inputs compared to the PD controller. The overshoot in the height tracking is reduced by about 40%. Tuning the LQR controller from the LQR controller described in (a) showed good results with a Q-z value of 300 and a Q-ψ value of 4. Figures 24-27 show the system response to the height and heading step inputs.

Table 2: Quadrotor system response to a step input for the height and heading using LQR.

Value	M_P (%)	t_r (s)	t_s (s)	y_{ss}
Height	4.5	0.255	0.345	0.1 m
Heading	0	0.475	0.535	15°

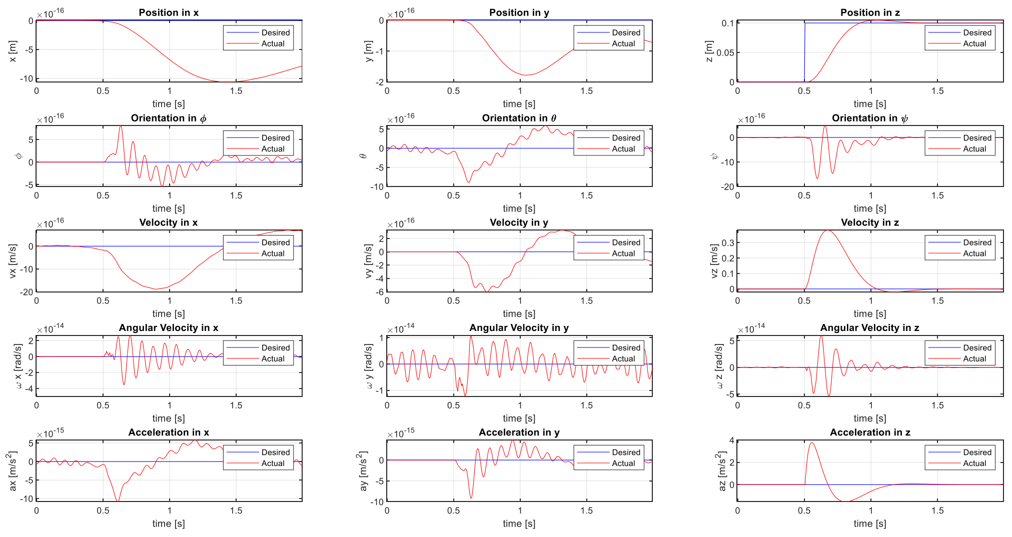


Fig. 24: System response to a step input in height results in smooth convergence and small overshoot.

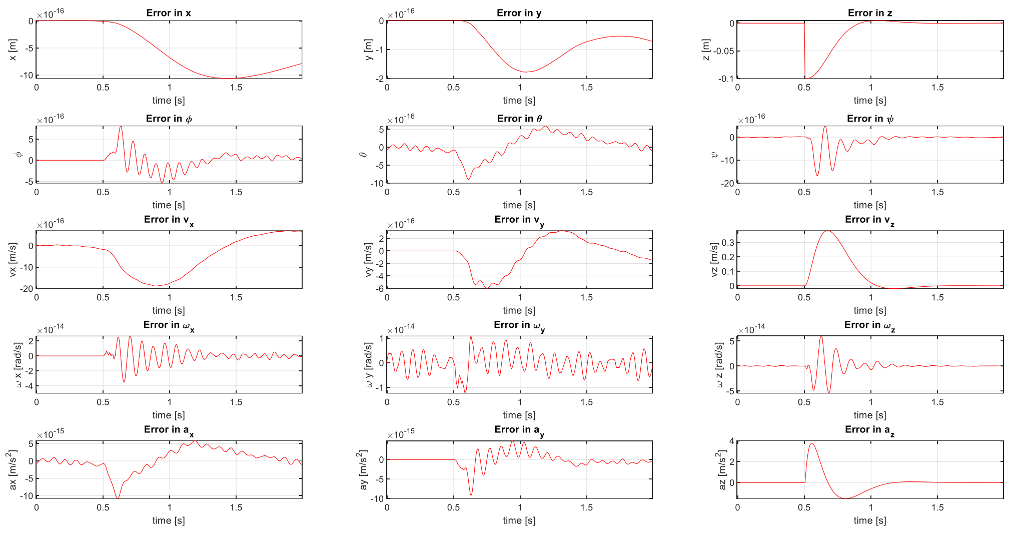


Fig. 25: Error plots showing the system response to a step input.

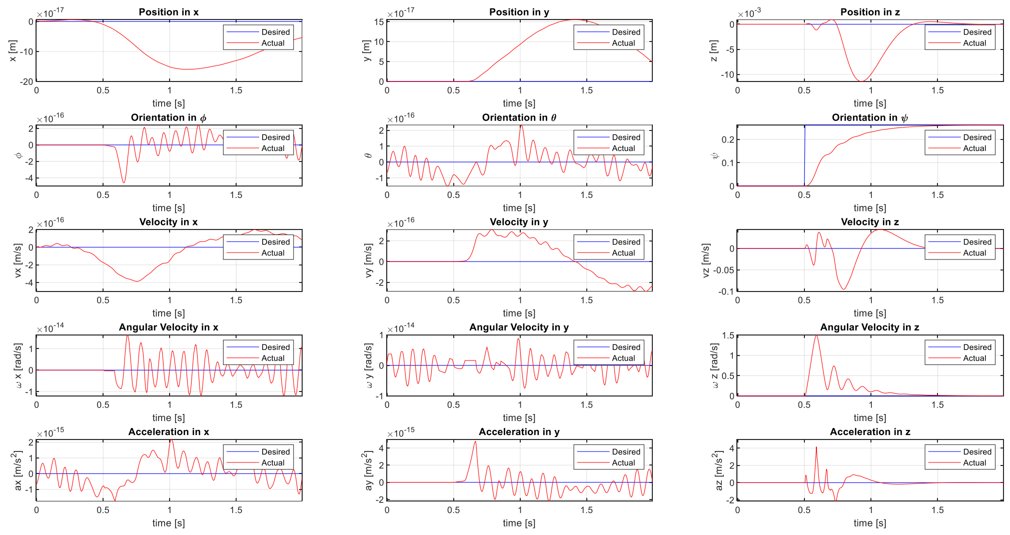


Fig. 26: Position plots for the heading step input show an oscillatory, underdamped convergence to the input.

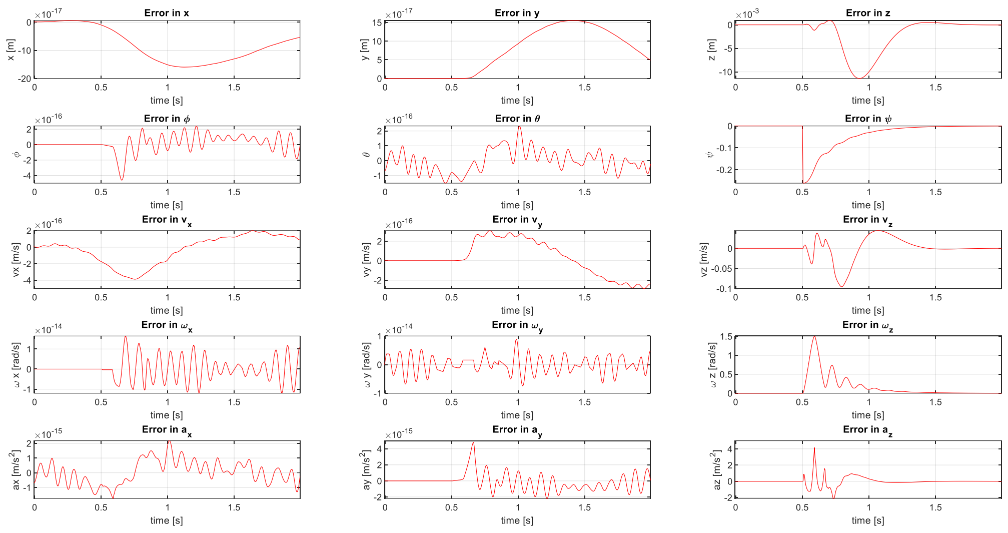


Fig. 27: Error plots for the heading step input show significant error only occurs in ψ, and a_z.

Modifying the Q-z and Q-ψ values produces interesting results. Increasing the ψ value to 8 causes the ψ value to converge much more rapidly. The increased gain also attenuates the oscillations in ψ as it approaches the goal point. After increasing ψ, the z-position tracking experiences a much larger overshoot than the previous controller. Increasing Q-z produces little change in the system. Figures 28 and 29 show the system responding simultaneously to a height and heading step input with the Q-ψ value increased to 1.6.

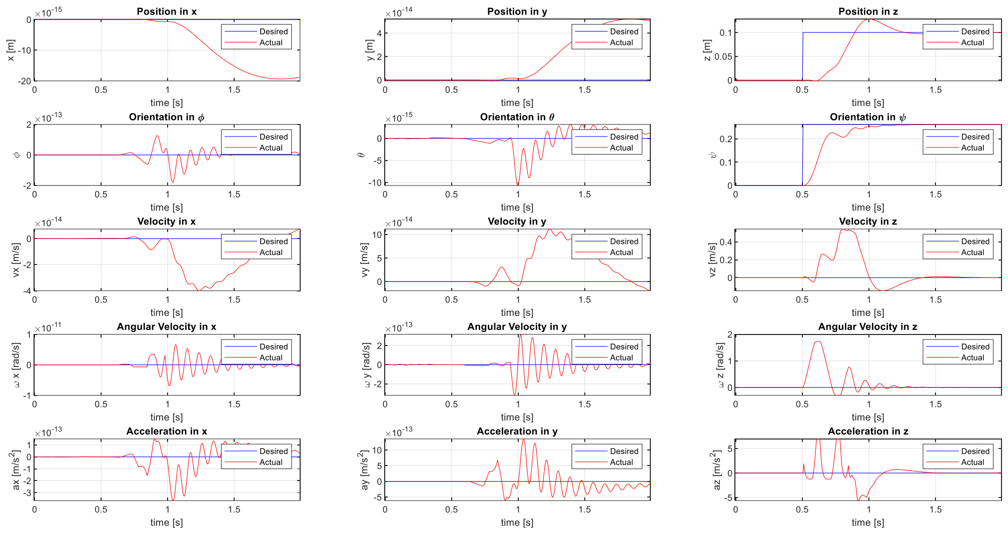


Fig. 28: State plots for the system with increased Q-ψ in response to step inputs in height and heading.

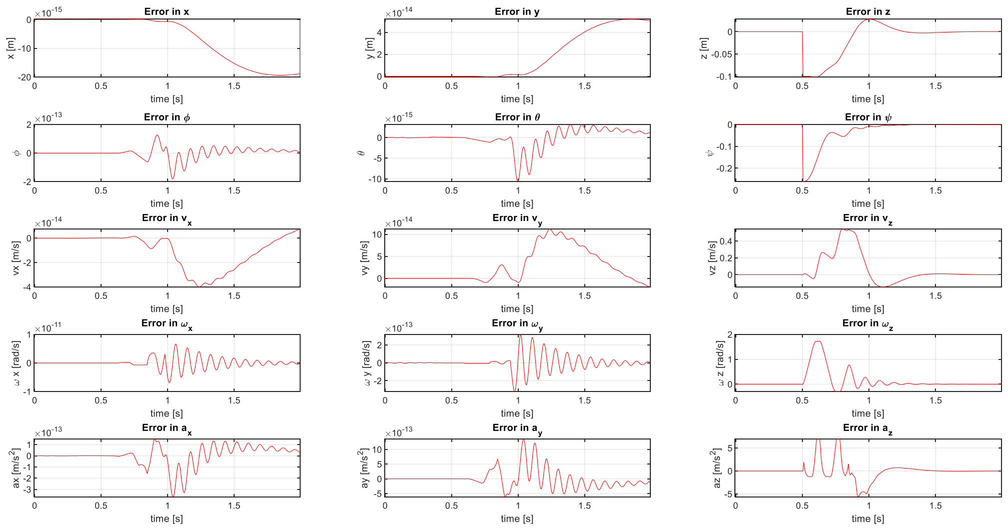


Fig. 29: Error plots for the system responding to step inputs in height and heading.

7) Bounded Acceleration Trajectory Generation

I defined the trajectory using the polynomial in Eq. 1. I then bounded the polynomial to have a maximum acceleration of 3 m/s^2. Using this strategy, the system tracks the trajectory without exceeding the maximum acceleration over a time scale of 4.5 seconds. The system tracks the trajectory well down to a time scale of 3.4 seconds. After this point, the actual trajectory oscillates around the desired trajectory. This occurs because the correction acceleration is too extreme and the bounded system over corrects in tracking the trajectory. Figures 30 and 31 show the position and error performance plots for a time scale of 4.5 seconds.

$$z(t) = 3t^2 - 2t^3 \tag{1}$$

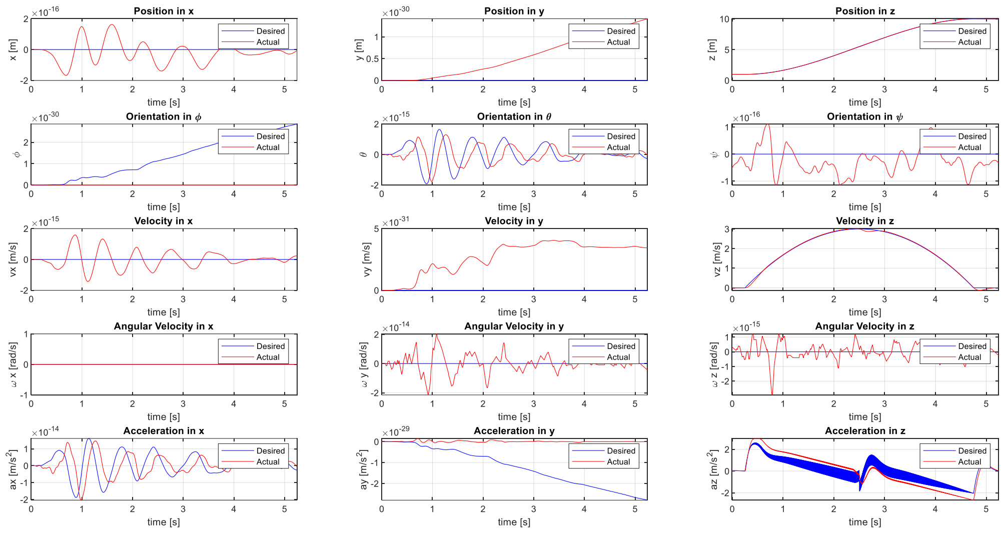


Fig. 30: The system successfully tracks a 9 m altitude increase without exceeding an acceleration of 3 m/s^2.

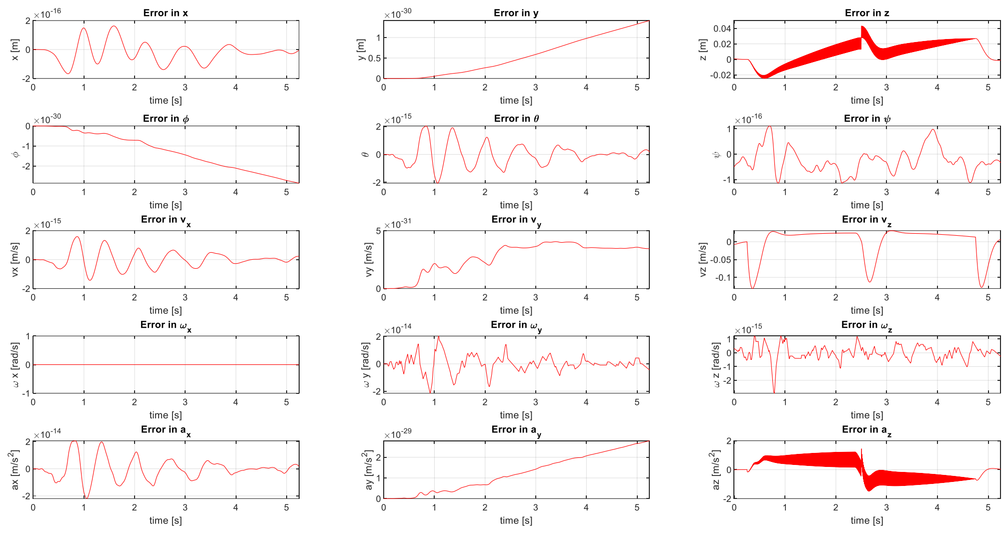


Fig. 31: The maximum z-position error is 4 cm. The wide lined error comes from the system aggressively correcting to the desired position. It is introduced in the position controller.

At a time scale of 3.2 seconds the system begins to show divergence from the desired path. Increasing the motor constant shows slight improvement to the system. The poor performance occurs due to the bounded acceleration. Adjusting the motor constant will only enable the motors to attain the desired accelerations more rapidly, not exceed the maximum accelerations. Because these desired accelerations are bounded, improved motors are not expected to significantly improve performance. Figures 32 and 33 show the position and error plots for this excessive acceleration.

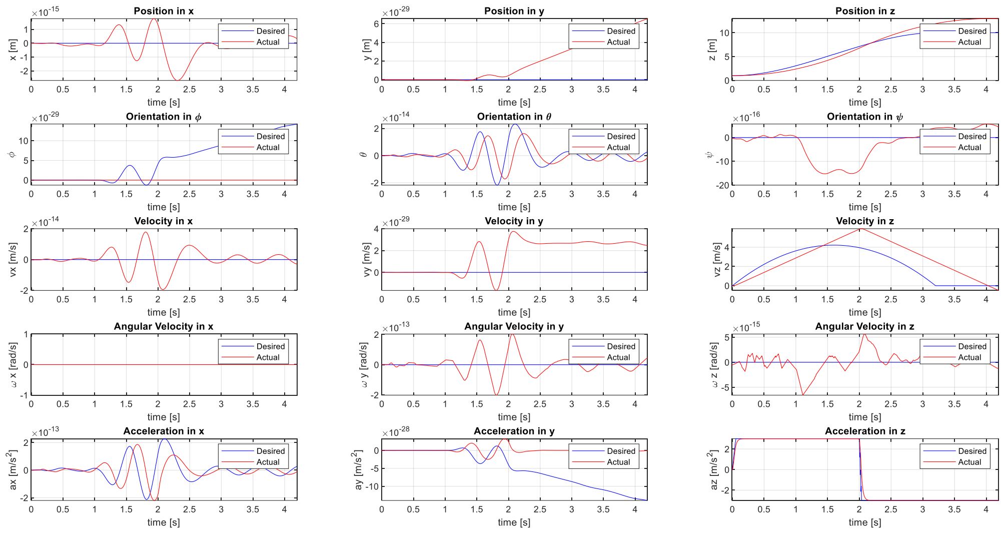


Fig. 32: The system diverges from the desired path, especially during the final hover. This occurs because the required acceleration is much larger than the bounded acceleration (see the Acceleration in Z-plot).

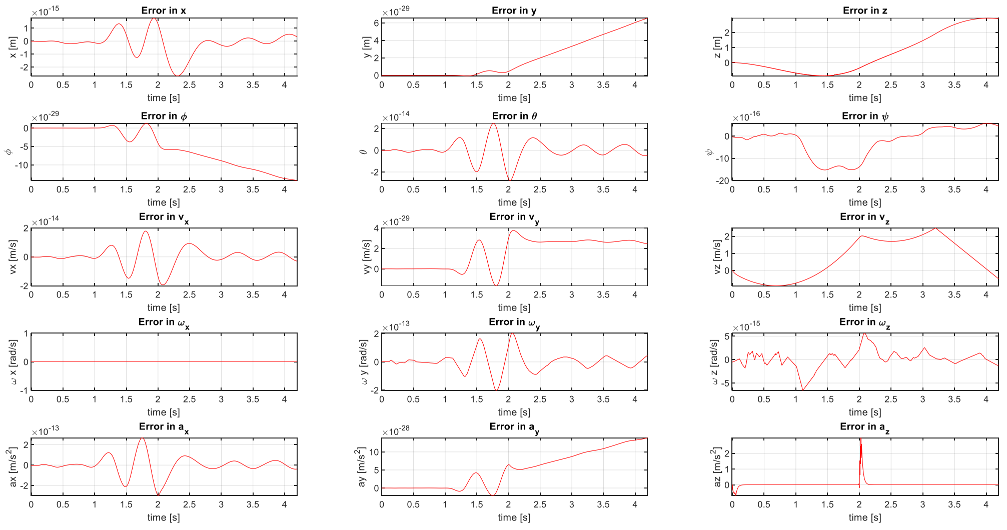


Fig. 33: The Z-error plot shows the divergence at the end of the trajectory. Note the Error in a_z plot remains relatively low, which shows increasing the motor constant will have little effect on the system performance.

8) Elliptical Trajectory

Figures 34 and 35 show the position and error plots for the elliptical tracking. The velocity plots show a commanded speed of 1 m/s. This controller performs well in the startup and continuing ellipses. Towards the end of the final ellipse the system shows overshoot in both the x and y positions. The overshoot likely occurs because the system has a high position gain relative to the derivative gain. As the drone begins to slow at the end of its trajectory, it attempts to converge to both the changing velocity profile and location, which results in the slight divergence.

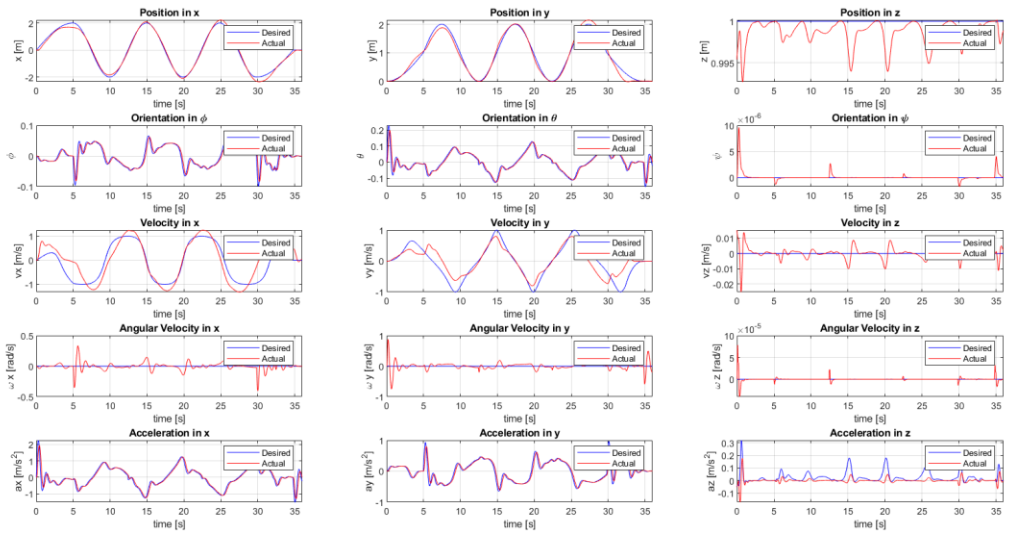


Fig. 34: Position plots for the drone tracking the ellipse at 1 m/s. The system successfully tracks position and acceleration with only slight deviations from the desired velocity.

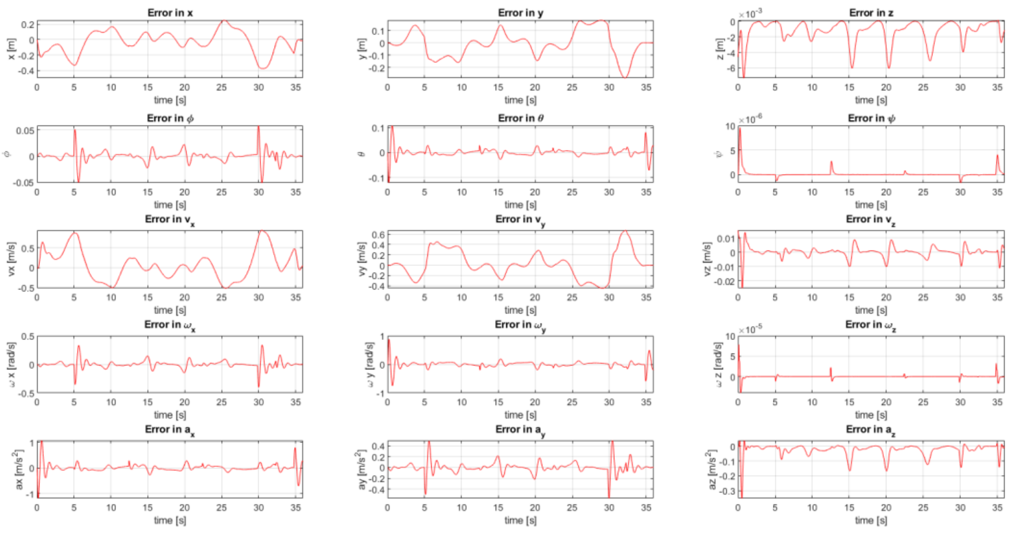


Fig. 35: Error plots for the drone tracking the ellipse at 1 m/s.

Reducing the velocity commands reduces the overshoot in the final ellipse. Increasing the velocity amplifies the overshoot (error) throughout the trajectory. The system begins to diverge when the commanded velocity approaches 2 m/s. This is because the drone must achieve accelerations larger than 3 m/s^2, which exceed the error bounds of the drone and cause it to diverge. Figures 36 and 37 show the position and error plots for a commanded velocity of 1.75 m/s.

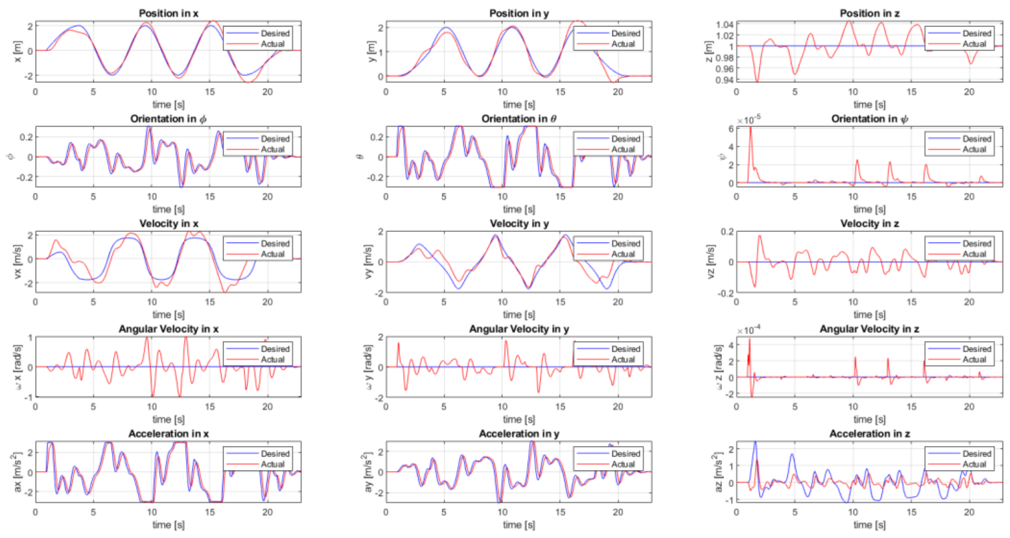


Fig. 36: Increasing the velocity during ellipse tracking shows the system diverges near the endpoints of the ellipse. This occurs due to the excessive acceleration during sharp turns.

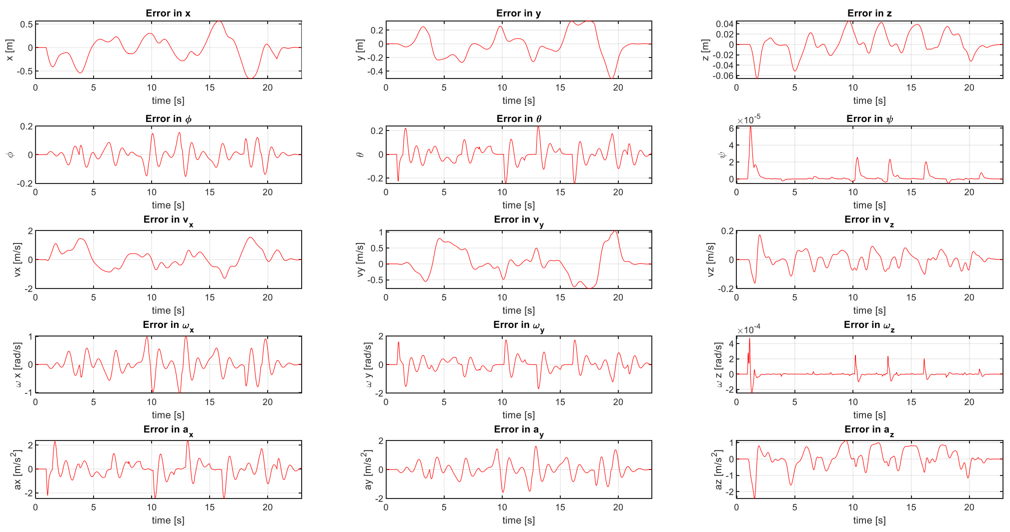


Fig. 37: The maximum position error occurs in the final turn of the third ellipse.

Figure 38 shows a cumulative error plot for the system. After creating the total cumulative error plot, I normalized the plots by the number of points per second in order to accurately compare different simulations. This scaling did not alter the cumulative error trends.

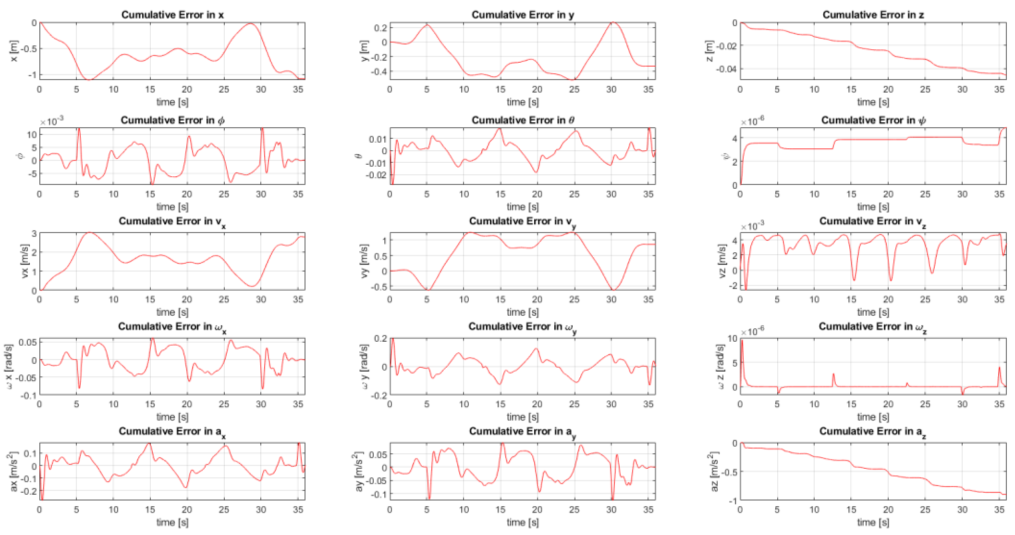


Fig. 38: Cumulative error plots for elliptical tracking at 1 m/s. Substantial error only occurs in velocity.

9) Robot Pirouette

Figures 39 and 40 respectively show the position and error plots for the pirouette tracking. At 1 m/s, the drone shows similar behavior to the ellipse tracking. The only difference is the orientation tracking. In pirouette tracking, the system tracks the heading angle with only a small delay.

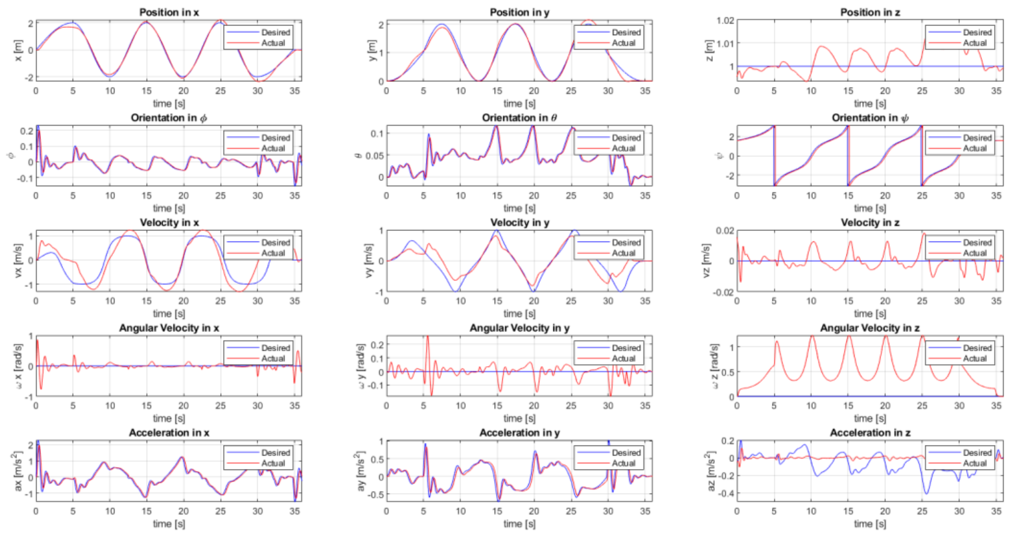


Fig. 39: Position plots for the drone tracking the pirouette at 1 m/s. The system successfully tracks position, acceleration, and heading with only slight deviations from the desired velocity.

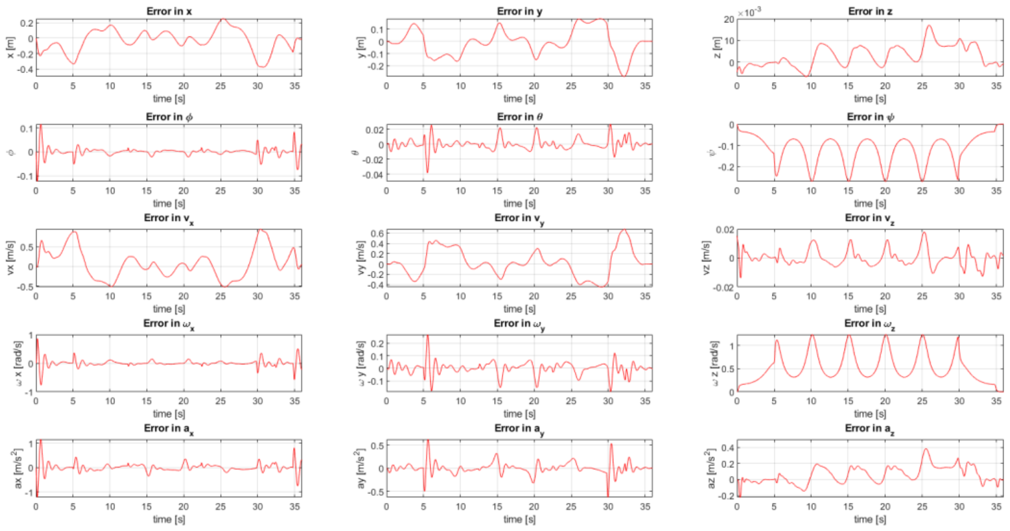


Fig. 40: Error plots for the drone tracking the pirouette at 1 m/s.

Modifying the velocities reacts similarly to the elliptical tracking. The magnitude of the tracking errors in the velocity, position, and heading all scale with the commanded velocity. However, the system can track the pirouette at 2 m/s, which was the divergence speed for the ellipse tracking. The pirouette tracking diverges just after 2 m/s. Figures 41 and 42 show the position and error plots of the pirouette at a commanded velocity of 1.75 m/s.

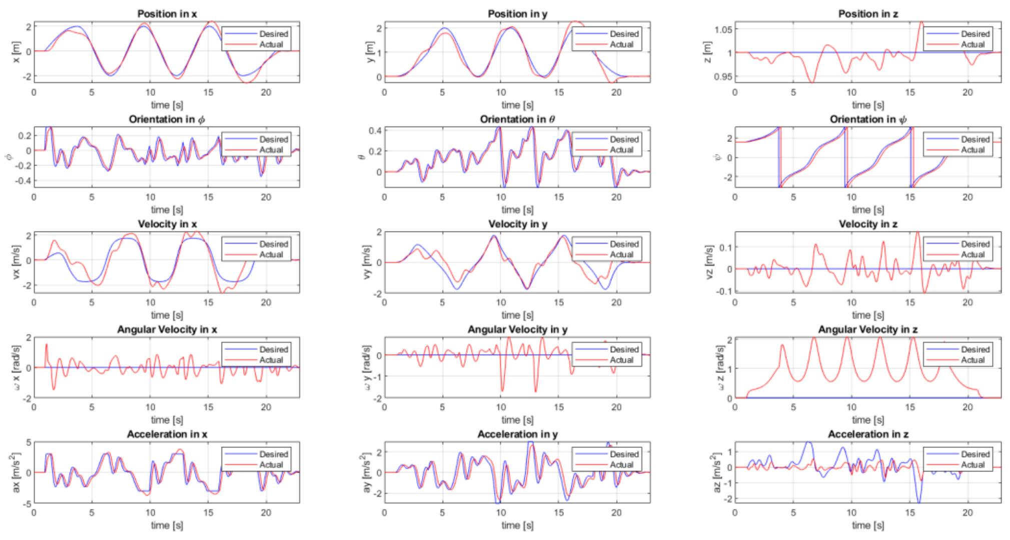


Fig. 41: Pirouette tracking behaves similarly to ellipse tracking at increased velocities.

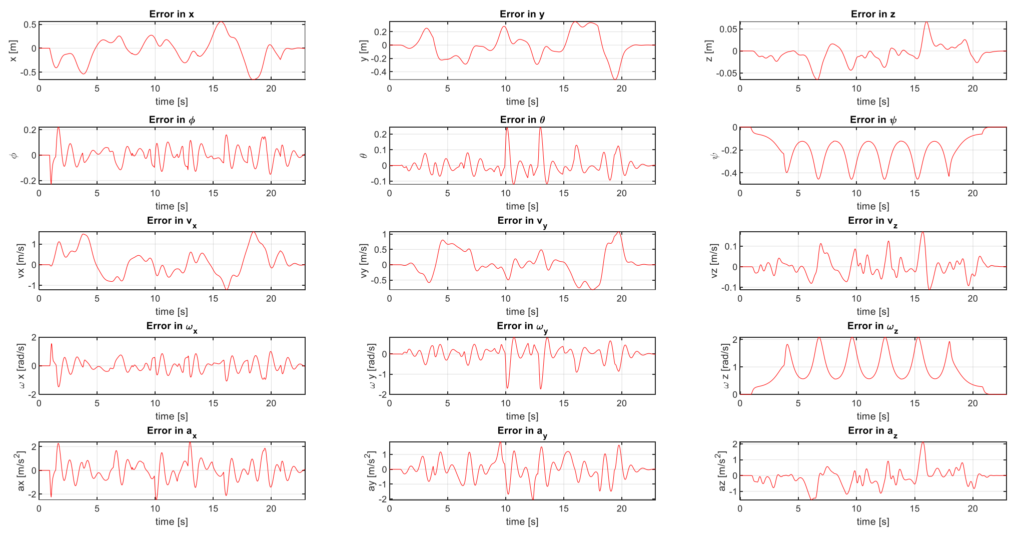


Fig. 42: Error plots in the commanded position behave similarly to the ellipse tracking. The error in commanded heading seems to scale linearly with velocity compared to Fig. 40.

I created the cumulative error plots for the pirouette tracker using the same method discussed in the ellipse tracking section. The only difference is the heading error. The heading has a slight offset throughout the whole simulation. This results in a nearly constant decreasing cumulative error. Figure 43 shows the cumulative error from this simulation.

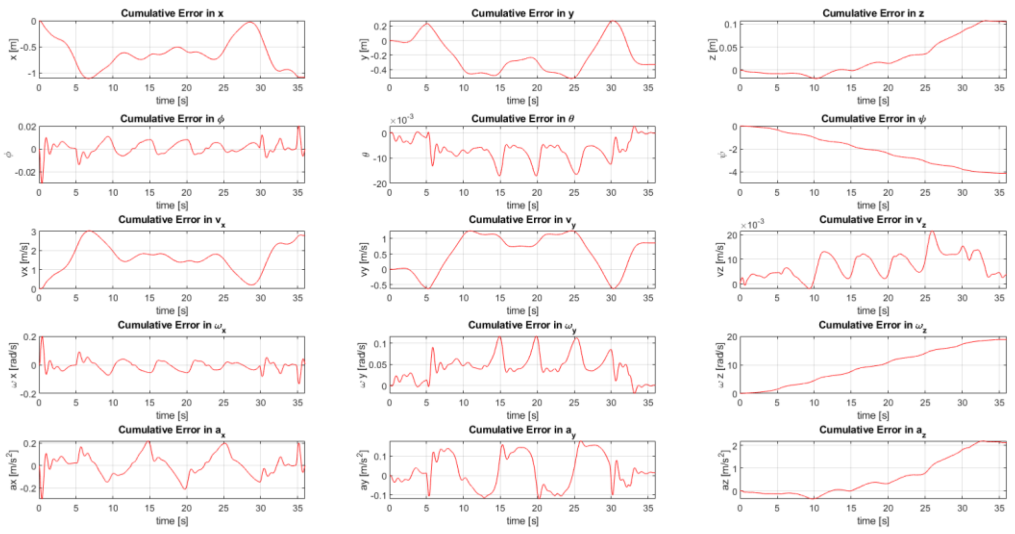


Fig. 43: Cumulative error plots for pirouette tracking at 1 m/s. Substantial error only occurs in velocity and heading.

Collaboration:

I collaborated with Shaun Ryer throughout this project. Our collaboration involved understanding the starter code, mapping the starter code modules to the lecture slides, debugging, and general advice/guidance in writing the report.