ASE 479W Laboratory 2 Report
Simulator Refinements and Quadcopter Control

Harrison Qiu Jin

February 24, 2022

1 Introduction

Unmanned aerial vehicles, or UAVs, have many different applications in a variety of industries. UAVs pose many challenges that are still being widely researched. As researchers develop different decision-making policies and algorithms, it is important for them to be able to easily test and iterate those models. While testing can certainly be done on physical hardware UAVs, any mistakes in the software or faulty decision-making can lead to costly crashes. Thus, it is important for researchers to be able to test their work on a simulated UAV with a robust dynamic model. This paper explores the development of a rudimentary simulator for a quadrotor drone.

Simulating the motion of a quadrotor drone requires a robust kinematic and dynamic model of the drone. This paper describes refinements that can be made to a rudimentary quadrotor UAV simulator so that it behaves more realistically. The first refinement is the effect of aerodynamic drag on the drone's motion. Drag is an important aspect of UAV motion because it can significantly change the net force that acts on the drone, especially at high speeds. Secondly, the dynamics of the motors that power the quadrotor's rotors are explored. In a rudimentary simulator, the angular rates of the rotors are set directly by the simulator such that they can change instantaneously. On a physical drone, the rotor rates cannot change instantaneously, as they are a function of the voltage that is applied to the motors. This adds some latency to the acceleration of the drone that is important to model in a simulator. Finally, PD control for both the drone's trajectory and attitude control are developed. PD control allows the motion of the drone to be more robust to external disturbances or noise. It also adds a layer of abstraction such that the user would only need to provide a trajectory rather than calculate the actual rotor rates needed for a particular trajectory. These refinements together result in what will hereafter be referred to as the "high-fidelity" simulator.

2 Theoretical Analysis

2.1 Aerodynamic Drag Forces

As the quadrotor drone travels, it encounters a non-negligible amount of aerodynamic drag. This drag force is a function of many different factors. The attitude, and specifically the direction of the body z axis, is important in calculating the drag force because it affects the amount of surface area on the quad that is normal to the drag force. Another element of the quadrotor's state that affects drag force is the velocity of the quadrotor. The drag force will always be in the opposite direction

of the quadrotor's velocity. The magnitude of the quad's velocity also affects the magnitude of the drag force that acts on the quad, as drag is caused by the air's resistance to the motion of the quad. Of course, this implies that the properties of the air are also important to consider when calculating the amount of aerodynamic drag. The property that contributes the most to aerodynamic drag is the mass density of the air.

It is difficult to fully determine the relationships between all of these variables analytically. The best way to develop a model for aerodynamic drag is to run an experiment where each of these variables are tested. One way to run this experiment would be in a closed wind tunnel with a model quadrotor fixed to a force sensor. Each of the factors described above could then be altered individually. Instead of moving the quadcopter, the motion of the quadrotor could be simulated by changing the velocity of the wind. By observing how the drag force changes with each factor, a reasonable model for the aerodynamic drag could be developed.

In the absence of being able to run the aforementioned experiment, the aerodynamic drag d_a must be approximated with a simplified model. For the purposes of this paper, the quadrotor is modeled as an infinitely thin circular disk in the x_B-y_B plane. With this model, d_a can be approximated with the following expression.

$$d_a = \frac{1}{2} C_d A_d \rho f_d(z_I, v_I) \tag{1}$$

Here, C_d is a unitless drag coefficient, A_d is the area of the circular disk, and ρ is the mass density of the air. $f_d(z_I, v_I)$ is a function of the body z axis expressed in the inertial frame (z_I) and the velocity of the quad (v_I). It is known that aerodynamic drag is proportional to both the magnitude of v_I squared as well as the cross-sectional area of the quad perpendicular to the direction of travel [1]. The expression $cos(\theta)$ can be used as the fraction of the cross-sectional area perpendicular to v_I, where θ is the angle between v_I and z_I. Thus, the following expression can be found for f_d

$$f_d(z_I, v_I) = \|v_I\|^2 cos(\theta) = \|v_I\|^2 \frac{v_I \cdot z_I}{\|v_I\|\|z_I\|} = \|v_I\|(v_I \cdot z_I) \tag{2}$$

Substituting (2) into (1) yields the following expression for the magnitude of aerodynamic drag d_a.

$$d_a = \frac{1}{2} C_d A_d \rho \|v_I\|(v_I \cdot z_I) \tag{3}$$

With this expression for aerodynamic drag as well as the fact that aerodynamic drag acts in the opposite direction of motion, the drag force acting on the quadrotor can be added into the force balance equation.

$$m\ddot{r_I} = -d_a v_I^u - mzZ_I + \sum_{i=1}^{4} F_{Ii} + d_I \tag{4}$$

Here, v_I^u represents the unit vector pointing in the direction of v_I.

2.2 Motor Dynamics

The motors of a quadrotor determine the angular rates of each rotor. However, motors are not able to instantaneously change the angular rate of a rotor in the real world. A voltage is applied to each

motor, which causes the attached rotor to reach the desired angular rate after a certain amount of time has passed. Using the coupled differential equations derived from mechanical and electrical relationships, the following transfer function can be used to model the relationship between the applied voltage and angular rate of the rotor.

$$\frac{\Omega(s)}{E_a(s)} = \frac{c_m}{\tau_m s + 1} \tag{5}$$

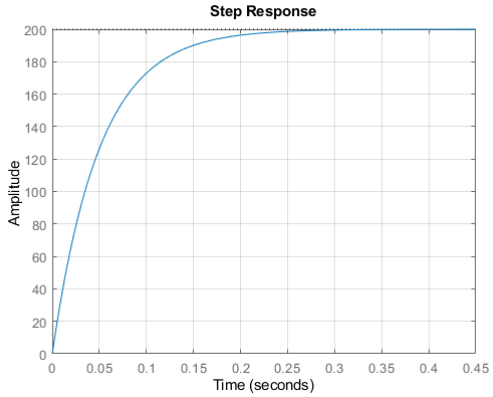
The transfer function (5) can be converted into a first order differential equation in the time domain.

$$(\tau_m s + 1)\Omega(s) = c_M E_a(s)$$

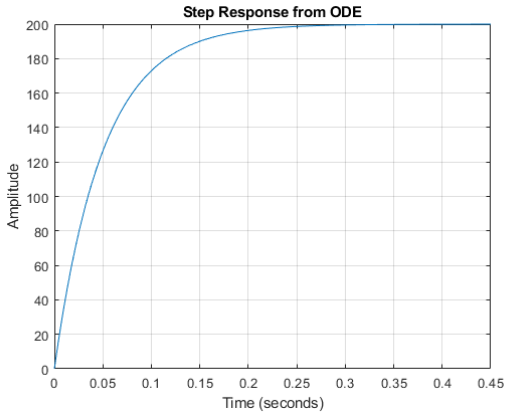
$$\tau_m s\Omega(s) + \Omega(s) = c_M E_a(s)$$

$$\tau_m \dot{\omega} + \omega = c_M e_a \tag{6}$$

The step response for both (5) and (6) is shown below in Figure 1.



(a) Step response for the transfer function from $E_a(s)$ to $\Omega(s)$



(b) Step response for the first-order differential equation relating ω and e_a

Figure 1: Step responses for both the transfer function and first order differential equation relating angular rate of the rotor and applied voltage.

The step response for (5) is identical to the step response for (6), which is the expected result. The 90% rise time for this transfer function is approximately 0.11 seconds. In theory, the rise time should not be affected by the magnitude of the step input. However, the physical meaning of this transfer function must be considered. In practice, there is a limit to how quickly the rotors can accelerate from 0 angular rate, meaning that there is a threshold for the input E_a at which the rise time will start to increase past 0.11 seconds.

2.3 Basic PD Control

Given that attitude errors are small and that the inertia matrix J of the quad is diagonal, all six degrees of freedom can be described by a scalar double-integrator system of the following form:

$$\ddot{y}(t) = k_y u(t)$$

$$Y(s) = \frac{k_y}{s^2}$$

This plant can be controlled by a PD controller of the form $C(s) = k + k_d s$. Figure 2 shows the unit step response for $k_y = 1$, $k = 10$, and $k_d = 10$.

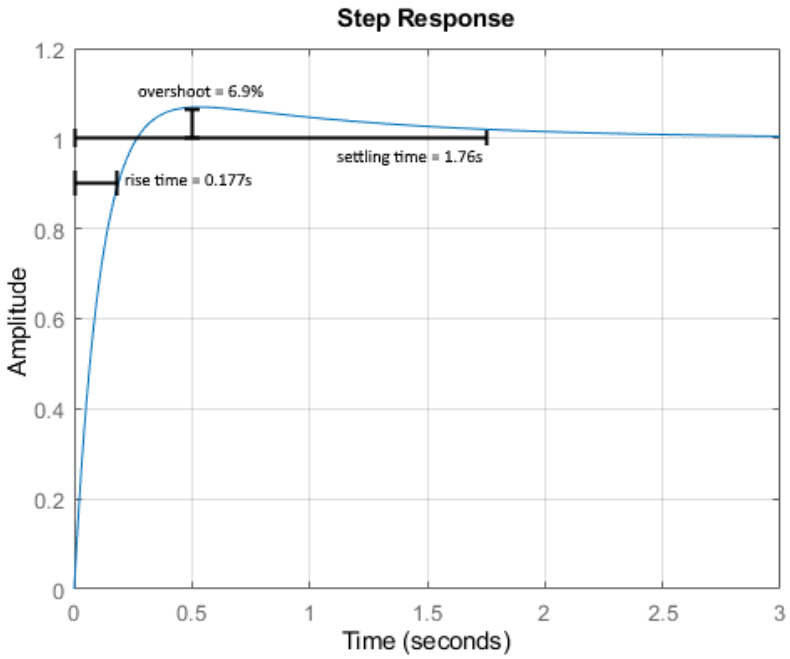


Figure 2: The step response for a double-integrator system controlled by a PD controller.

Using MATLAB's `stepinfo` function, the 90% rise time T_r is 0.177 seconds, the percent overshoot P_o is 6.9%, and the 2% settling time is 1.76 seconds.

2.4 Attitude Control

Euler's formula for a rotation about an axis of rotation $\hat{\boldsymbol{a}}$ and a rotation angle ϕ is the following:

$$R(\hat{\boldsymbol{a}}, \phi) = \cos\phi I_{3\times 3} + (1 - \cos\phi)\hat{\boldsymbol{a}}\hat{\boldsymbol{a}}^\intercal - \sin\phi\,[\hat{\boldsymbol{a}}\times] \tag{7}$$

$\hat{\boldsymbol{a}}$ is called the eigenvector of rotation. Let $\hat{\boldsymbol{a}} = [a_1, a_2, a_3]^\intercal$. Then, the fully evaluated expression for $R(\hat{\boldsymbol{a}}, \phi)$ is the following, where $\cos\phi$ is abbreviated as $c\phi$ and $\sin\phi$ is abbreviated as $s\phi$.

$$R(\hat{\boldsymbol{a}}, \phi) = \begin{bmatrix} c\phi & 0 & 0 \\ 0 & c\phi & 0 \\ 0 & 0 & c\phi \end{bmatrix} + (1 - c\phi) \begin{bmatrix} a_1^2 & a_1a_2 & a_1a_3 \\ a_1a_2 & a_2^2 & a_2a_3 \\ a_1a_3 & a_2a_3 & a_3^2 \end{bmatrix} - s\phi \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

$$R(\hat{\boldsymbol{a}},\phi)=\begin{bmatrix} c\phi+a_1^2(1-c\phi) & a_1a_2(1-c\phi)+a_3s\phi & a_1a_3(1-c\phi)-a_2s\phi \\ a_1a_2(1-c\phi)-a_3s\phi & c\phi+a_2^2(1-c\phi) & a_2a_3(1-c\phi)+a_1s\phi \\ a_1a_3(1-c\phi)+a_2s\phi & a_2a_3(1-c\phi)-a_1s\phi & c\phi+a_3^2(1-c\phi) \end{bmatrix}=\begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{8}$$

With (8), the components of $\hat{\boldsymbol{a}}$ can be found in terms of the values in the rotation matrix R.

$$\frac{R_{23}-R_{32}}{2\sin\phi}=\frac{(a_2a_3(1-\cos\phi)+a_1\sin\phi)-(a_2a_3(1-\cos\phi)-a_1\sin\phi)}{2\sin\phi}=\frac{2a_1\sin\phi}{2\sin\phi}=a_1 \tag{9}$$

$$\frac{R_{31}-R_{13}}{2\sin\phi}=\frac{(a_1a_3(1-\cos\phi)+a_2\sin\phi)-(a_1a_3(1-\cos\phi)-a_2\sin\phi)}{2\sin\phi}=\frac{2a_2\sin\phi}{2\sin\phi}=a_2 \tag{10}$$

$$\frac{R_{12}-R_{21}}{2\sin\phi}=\frac{(a_1a_2(1-\cos\phi)+a_3\sin\phi)-(a_1a_2(1-\cos\phi)-a_3\sin\phi)}{2\sin\phi}=\frac{2a_3\sin\phi}{2\sin\phi}=a_3 \tag{11}$$

The trace of R is

$$tr(R)=R_{11}+R_{22}+R_{33}=c\phi+a_1^2(1-c\phi)+c\phi+a_2^2(1-c\phi)+c\phi+a_3^2(1-c\phi)=3c\phi+\|\hat{\boldsymbol{a}}\|^2(1-c\phi)$$

Given that $\hat{\boldsymbol{a}}$ is a unit vector,

$$tr(R)=3\cos\phi+1-\cos\phi=2\cos\phi+1$$

Let R_E be the error direction-cosine matrix between the current attitude of the quad and the desired attitude.

$$R_E=\begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}$$

For the purposes of this simulator, the following attitude control law is used.

$$N_B=Ke_E+K_d\dot{e}_E+[\omega_B\times]\,J\omega_B \tag{12}$$

where

$$e_E=\begin{bmatrix} e_{23}-e_{32} \\ e_{31}-e_{13} \\ e_{12}-e_{21} \end{bmatrix}$$

Given (9), (10), and (11) it can be seen that e_E is an eigenvector of rotation for R_E. For the purposes of this paper, $\dot{e}_E$ is approximated as $-\omega_B$, which is appropriate for desired angular rates that are small.

3 Implementation

3.1 Aerodynamic Drag Force

The expression for aerodynamic drag force found in section 2.1 is easily implemented into the simulator by adding the drag force into the net force equation. However, it is important that the velocity vector is only normalized if it is nonzero. Otherwise, the normalization will result in a divide-by-zero.

3.2 Motor Dynamics

In the previous simulator, the rotor angular rates were an input into the `quadODEFunction`. However, as noted in section 2.2, the rotor rates cannot be set instantaneously in the real world. Therefore, the angular rate of each rotor is actually a state variable instead of an input. Using Equation 6, the time derivative of the angular rotor rates can be implemented with the applied voltage as an input.

3.3 Trajectory Control

To control the trajectory of the quadrotor, a basic PD controller is used. This controller takes in the desired position, velocity, and acceleration as input, and outputs a desired total thrust force F and attitude. The desired total thrust force is calculated by accounting for the error in the desired position and velocity relative to the actual position and velocity of the quadrotor. In addition, a feed-forward term that accounts for the desired acceleration is added. Finally, the effects of gravity also need to be counteracted.

$$F_I^* = ke_r + k_d\dot{e}_r + \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + m\ddot{r}_I^* \tag{13}$$

Once the total desired thrust force is found, the desired attitude can also be calculated. The desired attitude is described by a unit vector that represents the desired direction of the body z axis z_I^*. Since quadrotors can only thrust in the direction of the body z axis, the desired attitude can be found by simply normalizing the desired thrust force.

$$z_I^* = \frac{F_I^*}{\|F_I^*\|} \tag{14}$$

Finally, the total thrust that should be applied is projected onto the quadrotor's current body z axis, since that is the only direction that the quadrotor can move at that particular moment in time.

$$F = F_I^* \cdot z_I = (F_I^*)^\intercal R_{BI}^\intercal e_3 \tag{15}$$

See section 3.6 for more information on how the PD gains were tuned.

3.4 Attitude Control

Using the z_I^* vector generated by the trajectory controller along with the desired body x axis x_I^*, the attitude controller can calculate a desired attitude for the quad R_{BI}^*. However, x_I^* is not necessarily orthogonal to z_I^*. Therefore, the desired body y axis y_I^* is found first by taking the cross product of z_I^* and x_I^*. Finally, the body x axis for the desired attitude can be found by taking the cross product of y_I^* and z_I^*. This process guarantees that z_I^* remains unchanged in the final desired attitude, as this is the axis that affects the trajectory of the quadrotor. Once R_{BI}^* has been found, the torque necessary to correct the quadrotor's attitude can be found using the PD control law in (12). See section 3.6 for more information on how the PD gains were tuned.

3.5 Conversion

The total force and torque applied to the torque is a function of the thrust force provided by each rotor.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ y_1 & y_2 & y_3 & y_4 \\ -x_1 & -x_2 & -x_3 & -x_4 \\ -k_T & k_T & -k_T & k_T \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} F \\ N_{B_x} \\ N_{B_y} \\ N_{B_z} \end{bmatrix}$$

To find the force needed by each rotor, the 4x4 matrix can be inverted. This allows the dynamics simulator to determine how fast to spin each rotor given the desired force and torques provided by the trajectory and attitude controllers. However, this may result in negative forces or forces that are greater than the maximum possible output of the rotors $F_{\max}$. The desired force must also be constrained to the combined maximum output of the 4 rotors. To prevent rotors from needing to produce a force greater than $F_{\max}$, F is scaled down to 0.9, and N_B is scaled down from 1 until no rotors need to produce force greater than $F_{\max}$. Any negative rotor forces are set to 0. Rotor forces can be easily be converted into angular rates. From their, the voltage to apply to each rotor can be found as described in section 2.2.

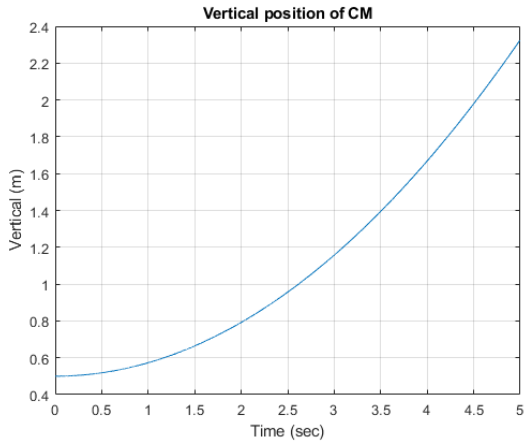
3.6 Tuning Controller Gains

The controller gains for both the trajectory controller and the attitude controller were tuned using iterative guess and check. First, the drone was set to a zero-velocity initial state. To tune the trajectory controller, a "step input" was provided to the quadrotor to move straight up without any attitude control. The proportional gain k was increased until oscillations appeared. The derivative gain k_d was then increased until the oscillations disappeared. This process was then repeated until the desired overshoot, rise time, and settling time were achieved. A similar process was used for tuning the proportional and derivative gains of the attitude controller, K and K_D respectively. For pitch and roll control, a "step input" point was provided either directly forward or to the side of the quadrotor's initial position. For yaw control, the quadrotor was simply instructed to yaw a certain amount in place.

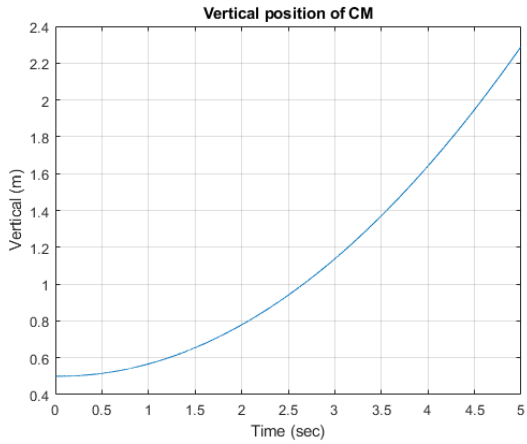
4 Results and Analysis

4.1 Effect of Motor Dynamics

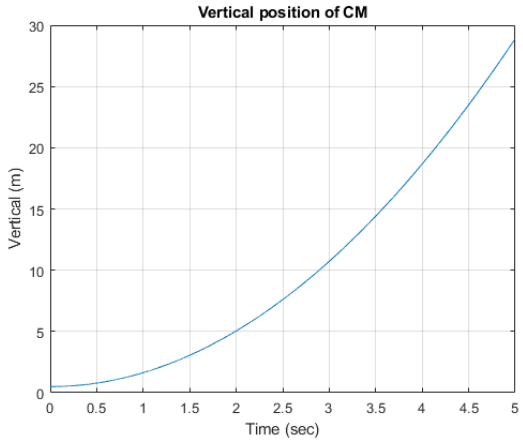
The addition of motor dynamics to the simulator means that there will be a time delay when changing the angular rate of the rotors. To observe this time delay, an experiment was run with both the original simulator and the refined simulator. In both cases, the quad began at an initial zero-velocity hover state. Then the quadrotor's rotors are set to a certain angular rate greater than that needed for them to hover. In the case of the original simulator, these rates are reached instantaneously. For the high-fidelity simulator, the appropriate voltage for that angular rate is applied. In each case, the vertical position of the quad over time was measured. This experiment was repeated for 3 different rotor rates, $\omega = 590, 650, 800$ The results of this experiment are shown below in Figure 3.



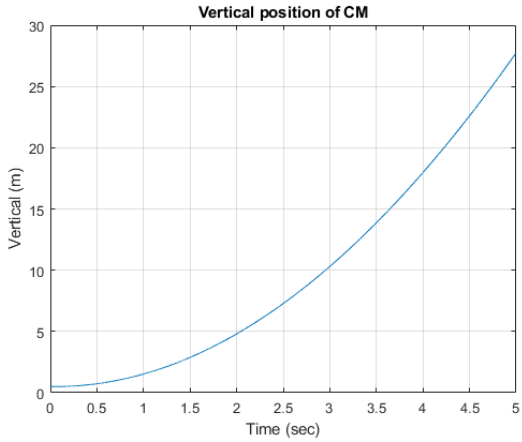
(a) Vertical position over time for the original simulator with $\omega = 590$



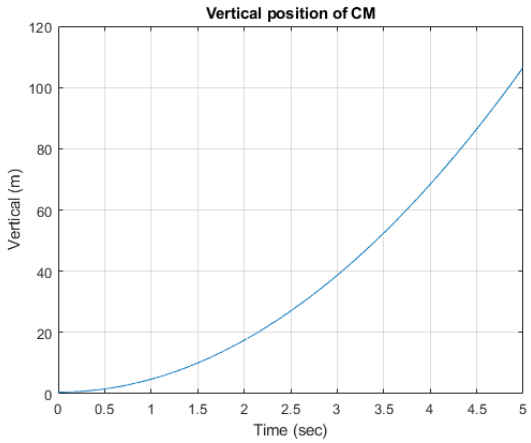
(b) Vertical position over time for the high-fidelity simulator with $\omega = 590$



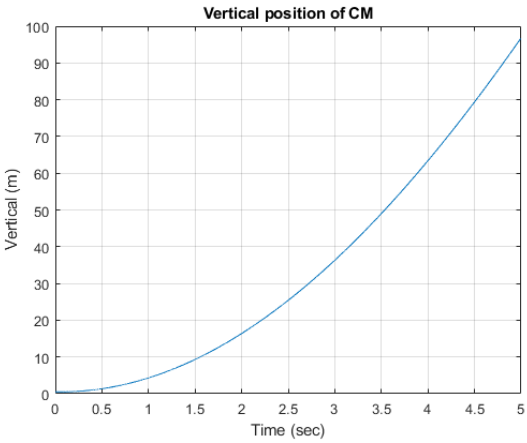
(c) Vertical position over time for the original simulator with $\omega = 650$



(d) Vertical position over time for the high-fidelity simulator with $\omega = 650$



(e) Vertical position over time for the original simulator with $\omega = 800$



(f) Vertical position over time for the high-fidelity simulator with $\omega = 800$

Figure 3: Comparison between original simulator and high-fidelity simulator.

The difference between the simulators is subtle yet noticeable. In all three experiments, the drone in the high-fidelity simulator ends slightly lower than the drone in the original simulator. This is the expected result, as there is an added latency for the rotors to change speed in the high-fidelity simulator. Thus, the drone also takes longer to accelerate.

4.2 Circular Trajectory

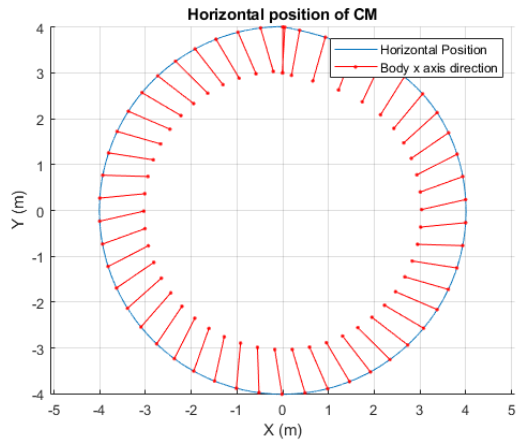
A circular flight trajectory with radius R, period T, and altitude h can be described by the following equations.

$$r_I^*(t) = \begin{bmatrix} R\sin(\frac{2\pi t}{T}) \\ R\cos(\frac{2\pi t}{T}) \\ h \end{bmatrix}, \dot{r}_I^* = v_I^* = \begin{bmatrix} \frac{2\pi R}{T}\cos(\frac{2\pi t}{T}) \\ -\frac{2\pi R}{T}\sin(\frac{2\pi t}{T}) \\ 0 \end{bmatrix}, \ddot{r}_I^* = a_I^* = \begin{bmatrix} \frac{-4\pi^2 R}{T^2}\sin(\frac{2\pi t}{T}) \\ \frac{-4\pi^2 R}{T^2}\cos(\frac{2\pi t}{T}) \\ 0 \end{bmatrix} \tag{16}$$

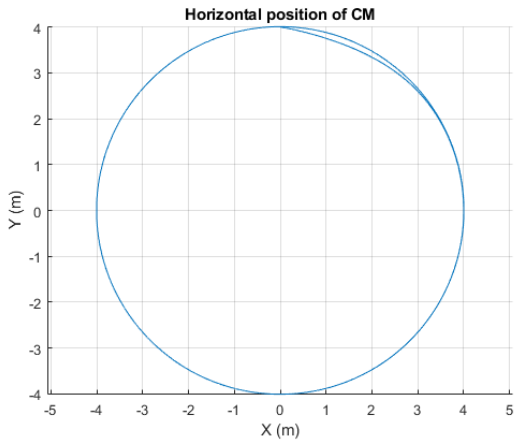
To keep the body's x axis towards the center of the circle, a desired x_I^* must also be generated. This can be done by recognizing that a_I^* is always pointed towards the center of the circle.

$$x_I^* = \begin{bmatrix} -\sin(\frac{2\pi t}{T}) \\ -\cos(\frac{2\pi t}{T}) \\ 0 \end{bmatrix} \tag{17}$$

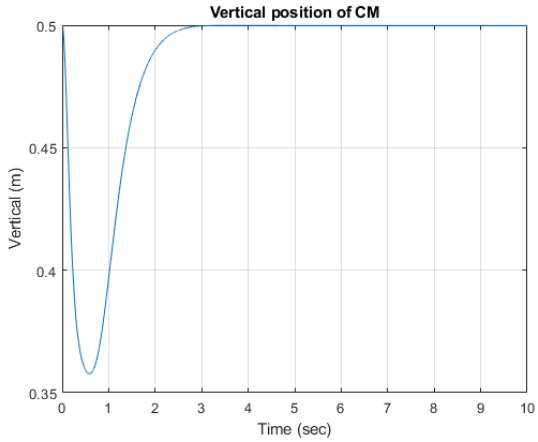
With these inputs, the simulated drone flies in a circular trajectory. This was tested using the following parameters: $R = 4m$, $T = 10s$, and $h = 0.5m$. The quadrotor was set to an initial state of zero velocity (linear and angular), zero angular rotor rate, and an initial position of $\begin{bmatrix} 0 & 4 & 0.5 \end{bmatrix}^\intercal$. The body x axis intially points towards the center of the circle. The results of the experiment are shown in Figure 4.



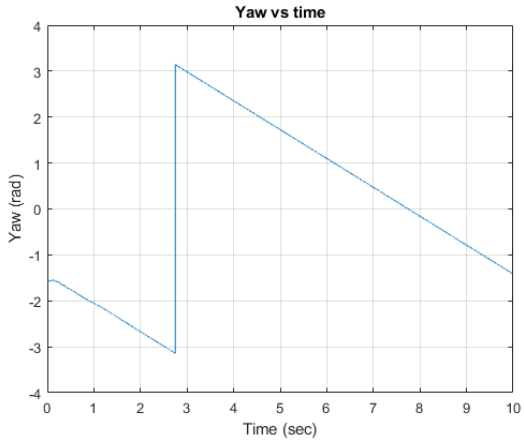
(a) The horizontal position of the center of mass over one period, with the direction of the body x indicated.



(b) The horizontal position of the center of mass over two periods.



(c) The vertical position of the center of mass over one period.



(d) The yaw of the quadrotor over one period.

Figure 4: Results of circular flight trajectory experiment.

As shown in Figure 4, the controllers seem to perform well. Figure 4a shows the direction of the body x axis over 1 period. There is some amount of error in that the body x axis doesn't point directly towards the center of the circle. This indicates that perhaps the PD gains for yaw control could be tuned better or it could be caused by some other source of error such as drag. Figure 4b shows that although there is some deviation in the first period due to the quad catching up from a zero-velocity initial state, the second period forms a relatively perfect circle. Figures 4c shows the drone "catching" itself from falling and maintaining the expected altitude of $0.5m$. Figure 4d shows that the quadrotor maintains a constant yaw rate without significant oscillation throughout the flight.

5 Conclusion

Several refinements to a rudimentary UAV simulator were explored in this paper to create a high-fidelity UAV simulator. These refinements included modeling aerodynamic drag, motor dynamics, and PD control of both the drone's attitude and trajectory. Each of these refinements added a layer of complexity to the simulator that brings it closer to behaving like a physical drone in the real world. The high-fidelity simulator was able to take in an input trajectory and successfully simulate the drone's motion following that trajectory using PD control.

References

[1] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *Proc. of the AIAA Guidance, Navigation, and Control Conference*, vol. 2, p. 4, 2007.