COMP0128: Robotic Control Theory and Systems CW2

Jeffrey Li

December 18, 2023

Notations:

The state p and dynamics $\dot{p}$ of the quadcopter are defined as follows:

$$p = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \\ \phi \\ \theta \\ \psi \end{bmatrix}, \dot{p} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \dot{\omega_x} \\ \dot{\omega_y} \\ \dot{\omega_z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{1}$$

$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ and $\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$ represents the position and velocity of the quadcopter in the inertial frame. $\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$ represents the components of the angular velocities pointing along the axis of rotation, and $\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}$ are the roll, pitch and yaw angles in the body frame of the quadcopter.

The input to the quadcopter is defined as u which is a vector of squares of angular velocities of each propeller γ_i:

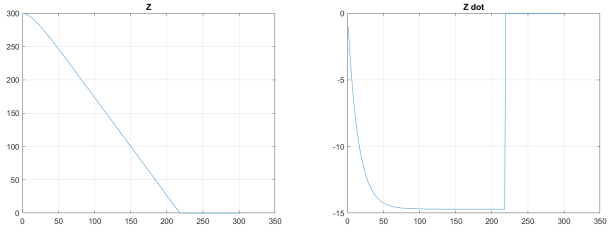
$$u = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{bmatrix} \tag{2}$$

Through out the report, we will represent $cos(x)$, $sin(x)$, $tan(x)$ functions as cx, sx, tx respectively.

Question 1:

How to run: Please run Sim_Quadcopter.m. To see simulations of different scenarios, please see the commented lines from lines 32-38. To view all plots, please see plots folder.

Answer: In the scenario of free fall, we initialize all inputs to zero, resulting in Figure 1, which illustrates the time-dependent changes in z and $\dot{z}$. Here, $\dot{z}$ gradually decreases as the quadcopter descends under the influence of its weight, eventually plummeting to zero upon impact with the ground. The deceleration of $\dot{z}$ occurs due to the escalating drag force countering the quadcopter's fall, leading to the eventual attainment of terminal velocity. The altitude, z, exhibits a curvilinear descent from the initial height of the quadcopter, driven by a more negative velocity component. All other components remain constant at zero.



(a) Variation of z over time (b) Variation of $\dot{z}$ over time

Figure 1: Key Free falling scenario plots

In [1] The acceleration of the quadcopter is modelled as follows:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \frac{1}{m} R T_B + \frac{1}{m} F_D \tag{3}$$

Where:

$$\begin{aligned} F_d &= -k_d[\dot{x}, \dot{y}, \dot{z}]^T \\ T_B &= [0, 0, \sum \gamma_i]^T \\ R &= \begin{bmatrix} c\psi c\theta & c\psi s\phi s\theta - c\phi s\psi & s\phi s\psi + c\phi c\psi s\theta \\ c\theta s\psi & c\phi c\psi + s\phi s\psi s\theta & c\phi s\psi s\theta - c\psi s\phi \\ -s\theta & c\theta s\phi & c\phi c\theta \end{bmatrix} \end{aligned} \tag{4}$$

At equilibrium, we need the following conditions:

$$\begin{aligned} [\dot{x}, \dot{y}, \dot{z}]^T &= 0 \\ [\ddot{x}, \ddot{y}, \ddot{z}]^T &= 0 \\ \phi = \theta = \psi = \omega_x = \omega_y = \omega_z &= 0 \\ \gamma_1 = \gamma_2 = \gamma_3 = \gamma_4 & \end{aligned} \tag{5}$$

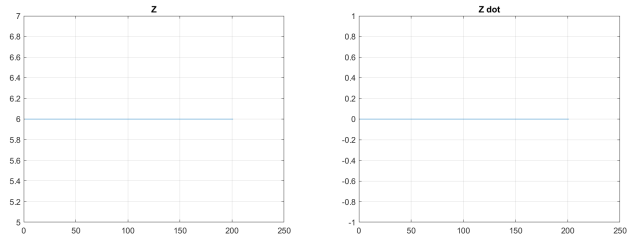
With some rearranging, we would get $RT_B = [0, 0, mg]^T$, which implies that $c\phi c\theta \sum \gamma_i = mg = 2.94$. Since $\phi = \theta = 0$, then we will have the condition:

$$\sum \gamma_i = 2.94 \tag{6}$$

In the equilibrium case, we need:

$$\gamma_1 = \gamma_2 = \gamma_3 = \gamma_4 = 0.735 \tag{7}$$

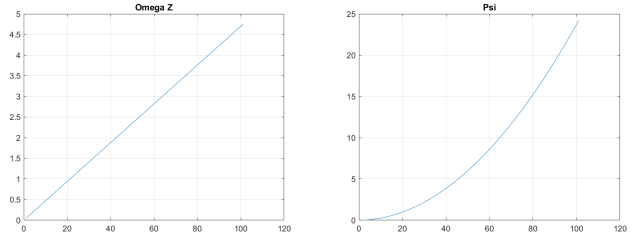
As the quadcopter is in equilibrium, all of its states will remain constant.



(a) Variation of z over time (b) Variation of $\dot{z}$ over time

Figure 2: Key Equilibrium scenario plots

To make the quadcopter spin at a set altitude, we once again need Equation 6, but this time we would need to set adjacent pairs of γ to be different and opposite pairs to have the same value, for example, $\gamma_1 = \gamma_3 = 0.97$, $\gamma_2 = \gamma_4 = 0.5$. This would result in a difference in torque across the adjacent propellers thus a constant angular acceleration on the z rotation axis (though realistically it wouldn't be constant due to drag) while having enough upward thrust to counteract the quadcopter weight.



(a) Variation of ω_z over time (b) Variation of ψ over time

Figure 3: Key Rotation scenario plots

Question 2:

How to run: Please run Sim_Quadcopter.m to see the linearized model, you may wish to change the error to lines 77-87. Run Sim_QuadcopterQ1.m to see the non-linear model. To view all plots, please see plots folder.

Answer: From [1], we would have the following formulations, note that there is an error for Equation 8 in the article which has been corrected:

$$\begin{bmatrix}\dot{\omega_x}\\ \dot{\omega_y}\\ \dot{\omega_z}\end{bmatrix} = \begin{bmatrix}\tau_\phi I_{xx}^{-1}\\ \tau_\theta I_{yy}^{-1}\\ \tau_\psi I_{zz}^{-1}\end{bmatrix} + \begin{bmatrix}\frac{I_{yy}-I_{zz}}{I_{xx}}\omega_y\omega_z\\ \frac{I_{zz}-I_{xx}}{I_{yy}}\omega_x\omega_z\\ \frac{I_{xx}-I_{yy}}{I_{zz}}\omega_x\omega_y\end{bmatrix} \tag{8}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & 0 & -s\theta\\ 0 & c\phi & c\theta s\phi\\ 0 & -s\phi & c\theta c\phi\end{bmatrix}^{-1}\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix} \tag{9}$$

The variables have the following values as described in the article, errata and the coursework document:

$$\begin{aligned}\tau_\phi &= 0.25(\gamma_1^2 - \gamma_3^2)\\ \tau_\theta &= 0.25(\gamma_2^2 - \gamma_4^2)\\ \tau_\psi &= 0.2(\gamma_1^2 - \gamma_2^2 + \gamma_3^2 - \gamma_4^2)\\ I_{xx} &= I_{yy} = 1\\ I_{zz} &= 0.4\end{aligned} \tag{10}$$

We can also calculate the inverse matrix for Equation 9 and using some trigonometric identities, we get:

$$\begin{bmatrix}1 & 0 & -s\theta\\ 0 & c\phi & c\theta s\phi\\ 0 & -s\phi & c\theta c\phi\end{bmatrix}^{-1} = \begin{bmatrix}1 & s\phi t\theta & c\phi t\theta\\ 0 & c\phi & -s\phi\\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta}\end{bmatrix} \tag{11}$$

Substituting these values and Equation 3 we can rewrite the dynamics $\dot{p}$ as follows, with m representing the mass of the quadcopter, and is kept to simplify calculations:

$$\dot{p} = f(u,p) = \begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\\ \frac{1}{m}(s\phi s\psi\sum\gamma_i + c\phi c\psi s\theta\sum\gamma_i) - \frac{0.2}{m}\dot{x}\\ \frac{1}{m}(c\phi c\psi s\theta\sum\gamma_i - c\psi s\phi\sum\gamma_i) - \frac{0.2}{m}\dot{y}\\ \frac{1}{m}c\phi c\theta\sum\gamma_i - \frac{0.2}{m}\dot{z} - g\\ 0.25(\gamma_1 - \gamma_3) + 0.6\omega_y\omega_z\\ 0.25(\gamma_2 - \gamma_4) - 0.6\omega_x\omega_z\\ 0.5(\gamma_1 - \gamma_2 + \gamma_3 - \gamma_4)\\ \omega_x + s\phi t\theta\omega_y + c\phi t\theta\omega_z\\ c\phi\omega_y - s\phi\omega_z\\ \frac{s\phi}{c\theta}\omega_y + \frac{c\phi}{c\theta}\omega_z\end{bmatrix} \tag{12}$$

With this, we could calculate the Jacobian matrices with respect to p and u around equilibrium points (see Equation 5 and 7) to obtain matrix A and B, which are then discretized using zero-on-hold to be used for digital simulation.

For the comparative tests, we would be altering the initial θ of the drone, which corresponds to its initial pitch in the body frame. For zero or close to zero error, we would set $\theta = 0.00001$, and for large error, we would set $\theta = 0.1$. We first verify that the linearization does indeed work around the equilibrium points with the initial z set to 3 and others set to 0 as shown in Figure 4.

For small errors, both nonlinear and linear systems demonstrate a relatively modest rate of deviation from the equilibrium point. Nonetheless, with the passage of time, they persistently move away from it (see Figure 6).

An intriguing observation from the plots is that a slight error in the pitch angle should not lead to a deviation in the Y direction. However, an acceleration is observed in the Y direction for the linearized model, aligning with the acceleration in the X direction, which is incorrect. It's noteworthy that in this scenario, there wouldn't be sufficient upward thrust to maintain

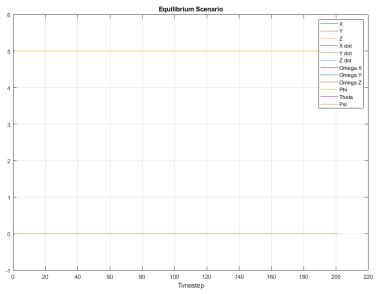
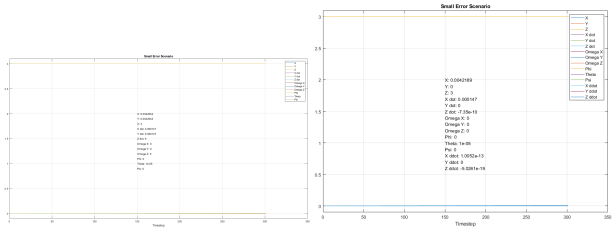


Figure 4: Variation of all states over time in equilibrium

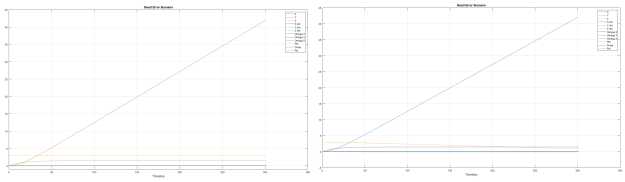


(a) Linearized Model (b) Non-linear model

Figure 5: Small error in θ ($\theta = 0.00001$)

the drone at the same altitude, given its tilted position. Consequently, one would expect a decrease in Z. This is depicted in the non-linear model where both $\dot{z}$ and $\ddot{z}$ are 0. However, in the linearized model, they remain at 0, which is also incorrect.

For large errors, as we would expect, the rate of deviation is much more rapid.



(a) Linearized Model (b) Non-linear model

Figure 6: Large error in θ ($\theta = 0.1$)

Once more, we observed the acceleration in the Y direction and the absence of a decrease in Z for the linearized model, a phenomenon more prominently evident in this particular scenario. To understand this effect, we can delve into the rows of the A matrix that impact $\dot{x}, \dot{y}, \dot{z}$, giving rise to the following equations:

$$\begin{aligned} \dot{x}[k+1] &= 0.9355\dot{x}[k] + 0.0479\omega_y + 0.9480\theta \\ \dot{y}[k+1] &= 0.9355\dot{y}[k] - 0.0479\omega_x + 0.0479\omega_y \\ &\quad - 0.9480\phi + 0.9480\theta \\ \dot{z}[k+1] &= 0.9355\dot{z}[k] \end{aligned} \tag{13}$$

This indicates that when $\omega_x = \omega_y = \phi = 0$, and both $\dot{x}$ and $\dot{y}$ are initialized as 0, as is the case in the present scenario, $\dot{x} = \dot{y}$. Moreover, since the initial value of $\dot{z}$ is set to 0, any error in θ will not affect $\dot{z}$.

Question 3:

How to run: Please run Sim_Quadcopter.m.To view all plots, please see plots folder. To speed up the run, you may wish to decrease the number of checkpoints generated for the circular trajectory.

Answer: After linearizing the dynamics around an equilibrium point and discretizing the linearized system, our initial step involves testing the controllability of this system. This is accomplished by constructing a controllability matrix and ensuring its full rank, which is confirmed in our analysis. Subsequently, we employed the Linear Quadratic Regulator to compute an optimal K matrix based on the specified Q and R matrices. The entire process is implemented in the fsf.m file.

Having obtained the K matrix, the controller is simply outputting the control signal:

$$u = min(max((-K * (x - ref)) + 0.735, 0), 1.5) \tag{14}$$

The min-max operation is employed to ensure that the signal falls within the range specified in the document. Here, x represents the current state, and ref is the target state to be reached. This signal is subsequently input into the drone simulator, which outputs a state vector denoted as y. Given that we have full observability in this scenario, where $x = y$, this updated state is utilized to generate a new control signal.

To achieve the designated trajectory, on top of the points set in the requirements, many more reference points have been generated (See Figure 7) in which the quadcopter must arrive within certain error thresholds.

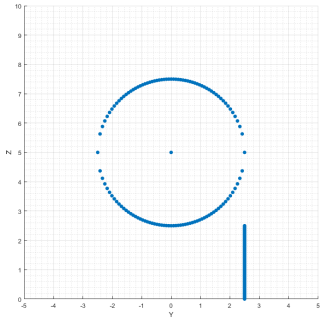


Figure 7: YZ plane view on the generated checkpoints

We demonstrate the quadcopter's successful arrival at the reference points through Figures 8, 9, and 10. The blue line represents the actual trajectory, while the red dots signify the corresponding components of the designed reference points. The vertical position of the red dots corresponds to the reference points, and the horizontal position indicates the timesteps at which the quadcopter believes it has reached each reference point. Notably, these positions perfectly align with each other in this case.

Furthermore, it is observed that certain spikes or abrupt increases occur in the X-coordinate plots, particularly in their

derivatives (refer to the plots folder). This issue could be mitigated by introducing more intermediate reference points to smooth out the trajectory.

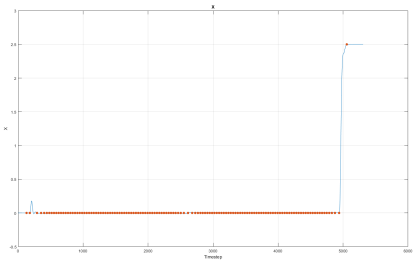


Figure 8: Quadcopter X coordinate Trajectory

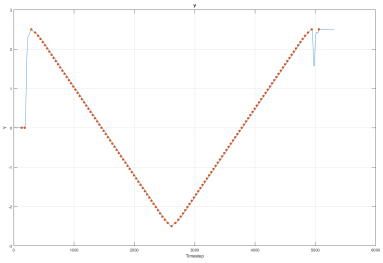


Figure 9: Quadcopter Y coordinate Trajectory

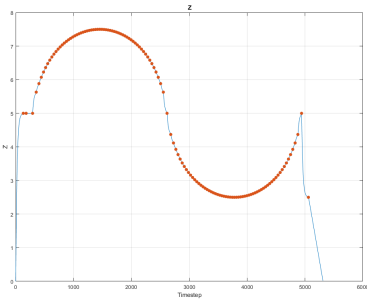


Figure 10: Quadcopter Z coordinate Trajectory

We also show that the quadcopter does indeed hover at (0,0,5) for five seconds with Figure 12, where the difference in timesteps is 50, and since $dt = 0.1$ it is indeed $5s$.

To show that the quadcopter moves in a circular trajectory crossing the designated points, a plot viewing the quadcopter's whole trajectory on the YZ plane is generated (See Figure 11).

For the final stage, the drone lands at a constant speed of $0.1ms^{-1}$. To achieve this, checkpoints are generated at a step size of $velocity * dt$ which equals 0.01 in this case. This is verified in Figure 13 where $\dot{z}$ converges to $-0.1ms^{-1}$.

Question 4:

How to run: Please run Sim_Quadcopter.m. To view all plots, please see plots folder.

Answer: For this question, we have assumed that $[\dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z]$ cannot be measured by the sensors, whereas the remaining states can be measured through the use of IMU, GPS and altimeter. With these states missing, we have asserted that using the remaining states, the system is still observable as the observability matrix is full rank.

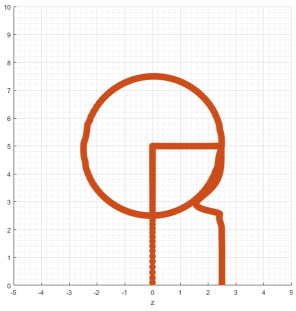


Figure 11: YZ plane view of quadcopter whole trajectory

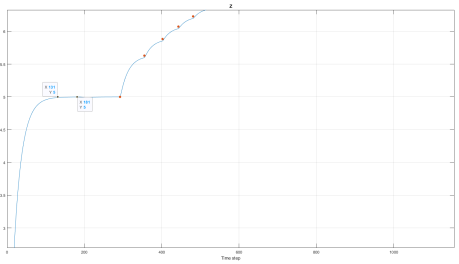


Figure 12: Quadcopter hovering at (0,0,5)

In terms of sensor noises, they have been modelled as Additive White Gaussian Noises plus a constant bias term. The bias term b is generated randomly at the start and is kept constant throughout the run, whereas the AWGN term w_t is generated at each timestep as follows:

$$\begin{aligned} b &\sim N(0, 0.001^2) \\ w_t &\sim N(0, 0.002^2) \end{aligned} \tag{15}$$

The wind is modelled using the simplified model described in [2] comprised of a mean wind field plus some stochastic turbulence. The mean wind field is modelled with u_{20} being the mean wind magnitude at 20 feet from the ground which is set to be 0.0001 and z being the current z-coordinate of the quadcopter:

$$w_c = \begin{bmatrix} u_{20} \frac{log(-z)/0.15}{20/0.15} \\ 0 \\ 0 \end{bmatrix} \tag{16}$$

And the stochastic turbulence is modelled as:

$$w_{turb} = \begin{bmatrix} w_u^{t+T} \\ w_v^{t+T} \\ w_w^{t+T} \end{bmatrix} = \begin{bmatrix} (1 - \frac{VT}{L_u}) w_u^t + \sqrt{\frac{2VT}{L_u}} \epsilon_u \\ (1 - \frac{VT}{L_v}) w_v^t + \sqrt{\frac{2VT}{L_v}} \epsilon_v \\ (1 - \frac{VT}{L_w}) w_w^t + \sqrt{\frac{2VT}{L_w}} \epsilon_w \end{bmatrix} \tag{17}$$

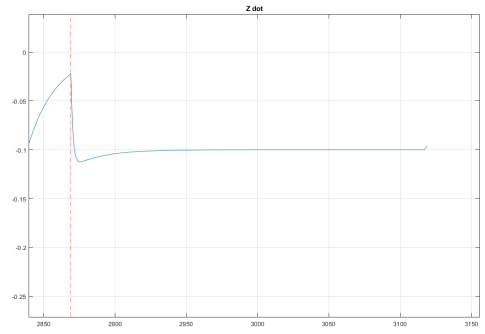


Figure 13: $\dot{z}$ at the last stage

Where:

$$
\begin{aligned}
V &= \text{Speed of the quadcopter} \\
T &= \text{Sampling time} = 0.1s \\
L_w &= -z \\
L_v = L_u &= \frac{-z}{(0.177 + 0.000823z)^{1.2}} \\
\sigma_w &= 0.1u_{20} \\
\sigma_u = \sigma_v &= \frac{\sigma_w}{(0.177 + 0.000823z)^{0.4}} \\
\epsilon_u &\sim N(0, \sigma_u^2) \\
\epsilon_v &\sim N(0, \sigma_v^2) \\
\epsilon_w &\sim N(0, \sigma_w^2)
\end{aligned}
\tag{18}
$$

The controller is implemented on top of what was implemented in Q3, with an additional state observer. The L matrix is calculated using pole placement on the A and C matrices with the designed eigenvalues. Since we know the initial state of the quadcopter, the state estimation $\hat{x}$ is generated as follows:

$$\hat{x}[k+1] = (A-BK)\hat{x}[k]+BK*ref+L(y[k]-C\hat{x}[k]) \tag{19}$$

Where $\hat{x}[0] = x[0]$. This estimated $\hat{x}$ is then fed into Equation 14 in place of x to generate a control signal.

The same set of reference points was generated as in Figure 7, which resulted in the trajectory shown in Figure 14 15 16, where the red lines represent the actual state of the drone, the blue line represents the predicted states using the state observer and the yellow dots represents the reference points. A figure for the landing speed of the quadcopter has also been generated as shown in Figure 17.

The effectiveness of the state observer in estimating the quadcopter's actual state is apparent, as evidenced by the close alignment of the blue line with the red line. In terms of the quadcopter's trajectory, it successfully reached most of the reference points along the intended path. However, due to additional noises, there are instances where the quadcopter doesn't reach certain points perfectly as the error threshold was increased to account for the additional noises.

A noteworthy consideration is that the noise levels are intentionally kept relatively low. This is crucial for ensuring that the controller can achieve the desired precision when arriving at reference points. Additionally, while the implemented noise model is somewhat realistic, there is room for improvement with more sophisticated noise models.

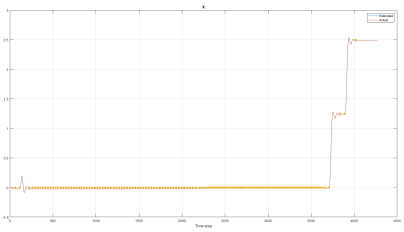


Figure 14: Quadcopter X coordinate Trajectory

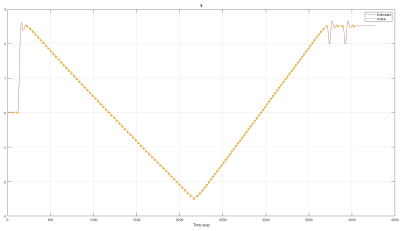


Figure 15: Quadcopter Y coordinate Trajectory

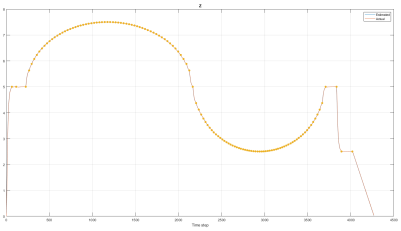


Figure 16: Quadcopter Z coordinate Trajectory

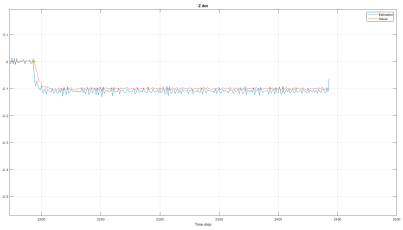


Figure 17: Quadcopter Landing Speed

References

[1] Andrew Gibiansky. "Quadcopter Dynamics and Simulation". In: (2012).

[2] Andrew Symington et al. "Simulating quadrotor UAVs in outdoor scenarios". In: (2014), pp. 3382–3388. DOI: 10.1109/IROS.2014.6943033.