Designing an Observer for Nonlinear Quadrotor System

Atharv Sathe
email assathe@andrew.cmu.edu

Tim McNamara
email tmcnama2@andrew.cmu.edu

December 18, 2020

1 Problem Motivation

In the initial part of this course's project, our LQR controller design strategy assumed we had access to accurate, up-to-date values for all of the states in the system. For Part 2 of the project, we investigated what happens if this is not the case. The state value measurements could be noisy, or a measurement of every single state may not be available. For our project, we set out to design and implement an observer to come up with our own estimate of the system state, $\hat{x}$, which is then used for the LQR controller to calculate the appropriate input to move the vehicle to the desired position and keep it stable.

2 Background

We begin with the same quadrotor model we developed for the first part of the project, based in part on the model presented in [1]. The states and inputs are defined as:

$$x = \begin{bmatrix} \dot{p_x} \\ p_x \\ \dot{p_y} \\ p_y \\ \dot{p_z} \\ p_z \\ p \\ \phi \\ q \\ \theta \\ r \\ \psi \end{bmatrix}, u = \begin{bmatrix} F_{thrust} \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix}$$

And once again, $p_x = x - x_{set}$, $p_y = y - y_{set}$, $p_z = z - z_{set}$ all in the NED Earth-fixed frame, therefore being at the origin indicates the vehicle is at the desired location.

The state dynamics of the system are described by:

$$f(t,x,u) \approx \begin{bmatrix} \cos(\phi)\sin(\theta)\frac{-F}{m} \\ \dot{p_x} \\ \sin(\phi)\frac{F}{m} \\ \dot{p_y} \\ g - \cos(\phi)\cos(\theta)\frac{F}{m} \\ \dot{p_z} \\ \frac{1}{J_x}\tau_\phi \\ p + q\sin(\phi)\tan(\theta) + r\cos(\phi)\tan(\theta) \\ \frac{1}{J_y}\tau_\theta \\ q\cos(\phi) - r\sin(\phi) \\ \frac{1}{J_z}\tau_\psi \\ q\frac{\sin(\phi)}{\cos(\theta)} + r\frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \tag{1}$$

Applying small-angle identities for the pitch and roll and linearizing $f(x,u)$ around the equilibrium states and input, we have approximations for A and B:

$$A = \frac{\partial f}{\partial x}\Big|_{\substack{x=x_e \\ u=u_e}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{2}$$

$$B = \frac{\partial f}{\partial u}\Big|_{\substack{x=x_e \\ u=u_e}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{J_x} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{J_y} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{J_z} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3}$$

And at any given time, the input to the system is determined by the state feedback control law

$$u = -Kx \tag{4}$$

where K was calculated using MATLAB's `lqr` function and Q and R matrices designed using Bryson's rule [2].

3 Linear Observer Design

If our measurement values for the states contain noise, or some states are not included directly in the output measurements, we need an observer to provide an internal estimate of the system's full state for our controller. For the scope of this project, we focused on designing and implementing a simple linear observer. This observer takes in in our sensor measurements and aims to generate $\hat{x}(t)$ that is as close to $\hat{x}(t)$ as possible. Following widely used standard linear observer procedure, outlined in [3] and many other sources, we let our observer have dynamics defined by

$$\frac{d\hat{x}}{dt} = A\hat{x} + Bu + L(y - C\hat{x}) \tag{5}$$

where $L \in \mathbb{R}^{n \times m}$. We then define the estimator error $\tilde{x} = x - \hat{x}$, then we find the dynamics (assuming our linearized state space model captures the dynamics of x reasonably accurately):

$$\dot{\tilde{x}} = \dot{x} - \dot{\hat{x}}$$

which simplifies to

$$\dot{\tilde{x}} = (A - LC)\tilde{x}$$

And as long as $\text{rank}(\mathcal{O}(A, C)) = n$, then we can use pole placement to ensure $(A - LC)$ is Hurwitz, and thus the error between the estimator and true state should decay asymptotically. As we test the viability of an observer, we still gather a measurement of every state from the EKF, so the C matrix is simply a 12-dimensional identity matrix, and the system is clearly observable.

Following the heuristic suggested in [4] that "observer eigenvalues should be placed 2-10 times faster than the slowest stable eigenvalue of the energy system itself", we set the observer poles to $k_{obsv}\lambda(A - BK)$, where $2 \leq k_{obsv} \leq 10$, and we tuned k_{obsv} to manually to improve performance. A block diagram overview of the observer for the quadcopter system is shown in Figure 1.

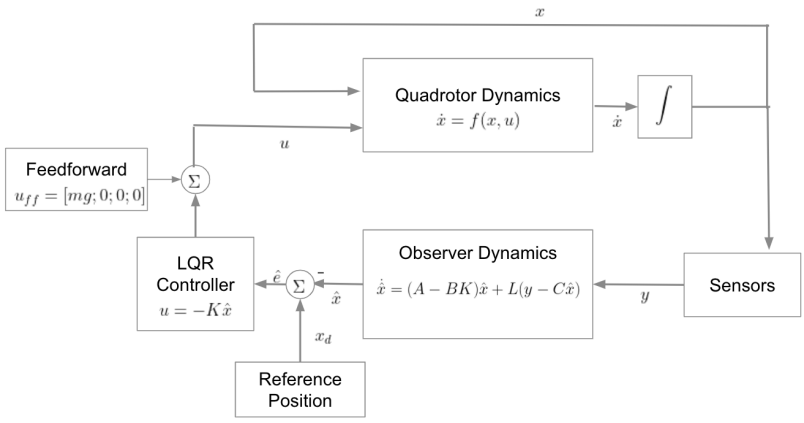

Figure 1: Linear Observer Block Diagram Overview

Although a simple MATLAB `ode45` implementation of system dynamics was used for the LQR design section of the project, for this project we transferred the model into Simulink to allow for increased complexity. Using linear observer poles placed at

$$\lambda(A - LC) = k_{obsv}\lambda(A - BK),\ k_{obvs} = 3$$

, we obtained an L matrix and tested the linear observer update function described in Equation 5. As expected for this trivial case, since all states were available and noise-free, the observer worked as expected. The positional displacement of the vehicle traveling the starting point $(-15, -15, -15, -10°)$ to the reference origin is shown in Figure 2 and the vehicle attitude for the same mission is shown in Figure 3. The estimator's position states tract the true states almost exactly, whereas there are slight deviations for the attitude values, which result from the small deviations between the non-linear dynamics of the true system and the linear update equation of the observer.

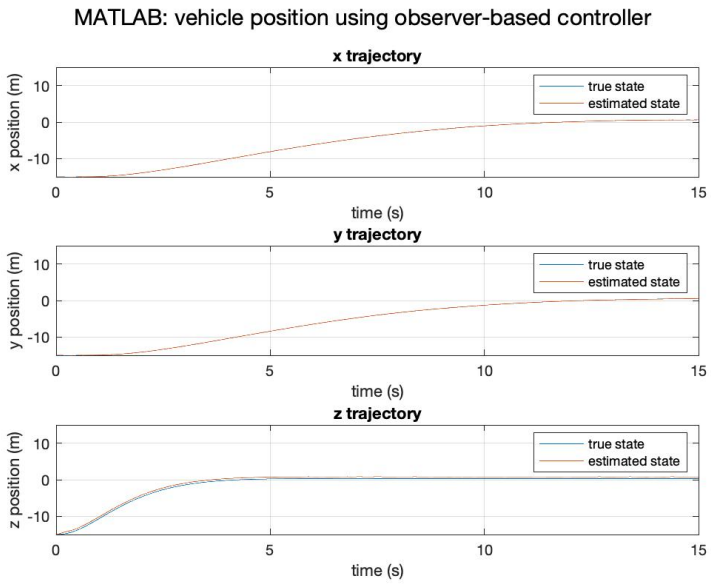

Figure 2: Simulink model for vehicle position over time using linear observer-controller

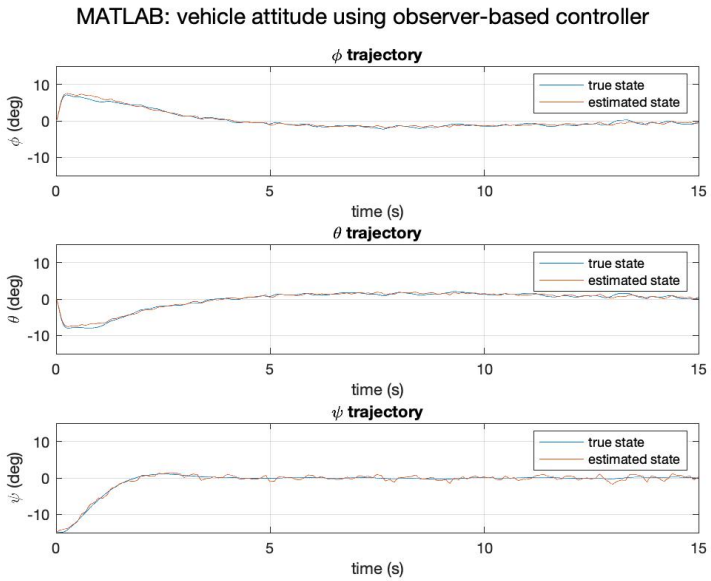

Figure 3: Simulink model for vehicle attitude over time using linear observer-controller

4 Implementation in PX4

To implement the observer model in PX4, we built on the implementation of the controller Atharv developed for the first part of the part of the project. Within the LQR controller module, we use the zero-order hold method to track and update our estimate for the states. We had some challenges implementing the update to prevent delays and synchronization from being a problem, but we eventually found that this order of operations, as well as checking that time had passed between loops, was necessary to make the observer-controller stable.

Algorithm 1: Single Loop of Linear Observer-Controller In PX4

Result: Normalized actuator inputs
now = `get_current_time()`;
$\dot{\hat{x}} = A\hat{x} + Bu + L(y - \hat{x})$;
dt = previous_time - now;
if $dt > 0$ **then**
  $\hat{x} = \hat{x} + \dot{\hat{x}}dt$;
  previous_time = now;
end
$\Delta x = \hat{x} - x_d$;
$u = -K\Delta x$;
$u =$ `normalize`(u);
$y =$ `pollMeasurements()`;
Store $u, y, \dot{\hat{x}}, \hat{x}$, previous_time for next loop;
Return u;

After seeing that the observer-controller could stably bring the vehicle to a set-point, we moved on to evaluating the observer's robustness to noisy measurements.

5 Evaluation Using Noise Injection

In order to quantitatively evaluate our observer's performance in PX4, we altered the PX4 EKF state estimator's measurements so that y would not match the vehicle's state exactly, allowing us to analyze the response of our observer to inaccurate measurements. To the EKF signal, we added a vector of Gaussian noise with zero-mean and standard deviations appropriate for each state. The first noise level (Set 1) is taken from PX4 default sensor error values. However, this noise is removed during processing by the EKF, so by adding noise back to the EKF measurements we can bring y values closer to real conditions. Sets 2 and 3 proportionally increase these levels to test the robustness of our linear observer. Noise 3 is the maximum set of values for which the observer responds satisfactorily. In order to generate a trend of observer performance, we chose Noise 2, an intermediate value between the both extremes. The values are shown in Table 1.

	Standard Deviation of Added Noise			
	p_x, p_y, p_z (m)	$\dot{p_x}$, $\dot{p_y}$, $\dot{p_z}$ (m/s)	ϕ, θ, ψ (rad)	p, q, r (rad/s)
Set 1	0.2	0.1	0.02	0.01
Set 2	0.6	0.3	0.06	0.03
Set 3	1.6	0.8	0.16	0.08

Table 1:

We compared the linear observer-controller's response to noisy measurements, i.e. a control law defined by

$$u = -K\hat{x}$$

to that of an LQR controller that uses the corrupted measurements directly to compute the input, i.e. a control law defined by

$$u = -K(x_{ekf} + noise)$$

The results of sending the vehicle from stationary at the origin to point (5,5,-5, 0) are shown in Figures 4 and 5. Observe that with the observer placed as part of the controller design, the attitude is far more stable and the vehicle gets slightly closer to the reference position.

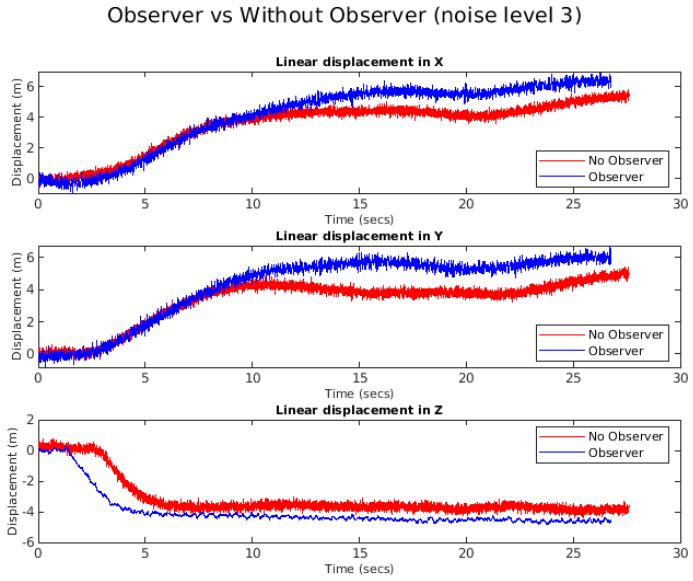

Figure 4: Vehicle position trajectory for observer-controller vs. controller using noisy measurements, additive noise level 3

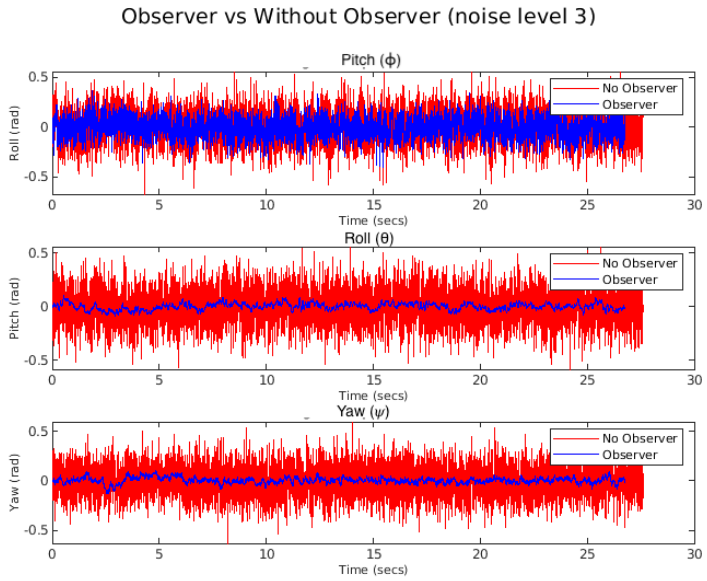

Figure 5: Attitude angle time series for observer vs. controller using noisy measurements, additive noise level 3

The results for the vehicle's mission to the same reference point, but for increasing levels of additive noise, are shown in Figures 6 and 7. Even at the highest noise level, the vehicle is still roughly able to move to the reference position, although there is a slight steady state error in p_x and p_y for the highest noise levels. But for noise set 1, which are close to the real gaussian noise in PX4, the response is very close to that of no noise on the measurements at all, indicating the linear observer-controller can filter out these deviations in state measurements. For angle attitudes, we see that the higher level of noise does cause the vehicle to wobble significantly more, but it remains stable overall.

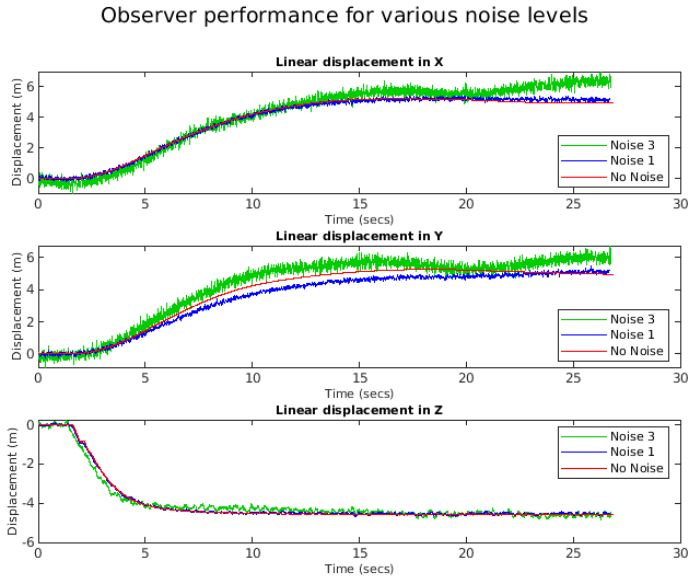

Figure 6: Vehicle position for using linear observer-controller with to increasing noise on measurements

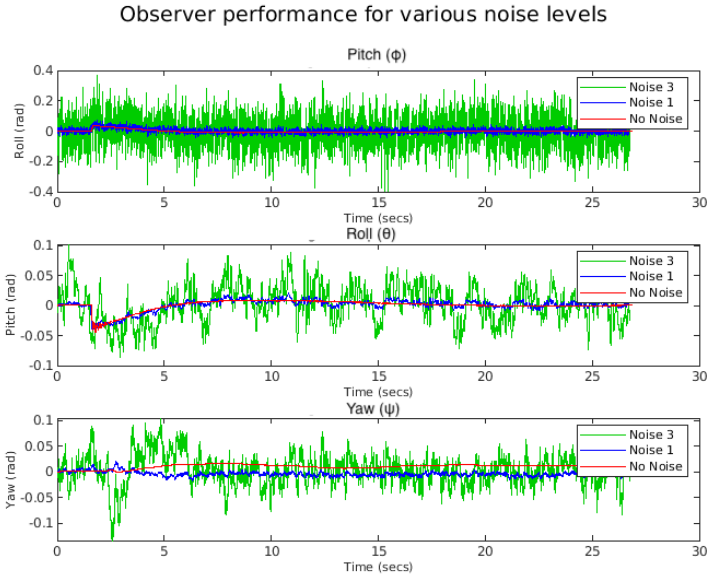

Figure 7: Vehicle attitude for using linear observer-controller with to increasing noise on measurements

Lastly, we gave the vehicle observer a time-based circular reference trajectory of radius 3m . The position trajectories for no additive noise and noise set 1 are shown in Figure 8,

indicating that even for a moving target, the observer-controller can stably track a reference with the addition of measurement noise.

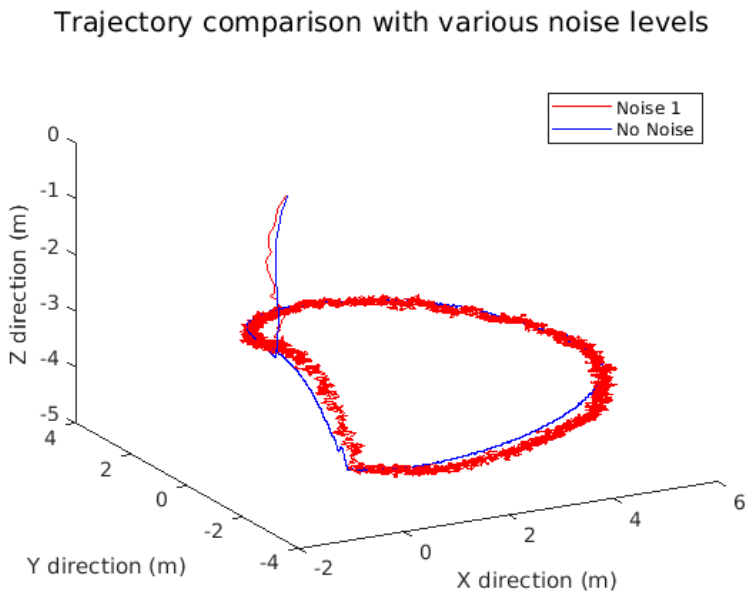

Figure 8: Observer-controller vehicle trajectory for moving reference position

6 Nonlinear Observer Update

Since our system dynamics are nonlinear, we also investigated the effect of using the nonlinear dynamics function in the estimator update equation instead of the linear state space model. In this approach, the observer update is defined by:

$$\frac{d\hat{x}}{dt} = f(\hat{x}, u) + L(Cx - C\hat{x}) \tag{6}$$

Also using the true nonlinear dynamics for $x(t)$, the error term dynamics become:

$$\dot{\tilde{x}} = \dot{x} - \dot{\hat{x}} = f(x, u) - f(\hat{x}, u) - L(y - C\hat{x})$$

If we look at the difference between Equation 1's output for x and $\hat{x}$, we get:

$$f(x,u)-f(\hat{x},u) = \begin{bmatrix} \frac{-F}{m}(\cos(\phi)\sin(\theta) - \cos(\hat{\phi})\sin(\hat{\theta})) \\ \dot{p_x} - \dot{\hat{p}_x} \\ \frac{F}{m}(\sin(\phi) - \sin(\hat{\phi})) \\ \dot{p_y} - \dot{\hat{p}_y} \\ \frac{F}{m}(\cos(\phi)\cos(\theta) - \cos(\hat{\phi})\cos(\hat{\theta})) \\ \dot{p_z} - \dot{\hat{p}_z} \\ 0 \\ p - \hat{p} + q\sin(\phi)\tan(\theta) - \hat{q}\sin(\hat{\phi})\tan(\hat{\theta}) + r\cos(\phi)\tan(\theta) - \hat{r}\cos(\hat{\phi})\tan(\hat{\theta}) \\ 0 \\ q\cos(\phi) - \hat{q}\cos(\hat{\phi}) - \hat{r}\sin(\hat{\phi}) \\ 0 \\ q\frac{\sin(\phi)}{\cos(\theta)} - \hat{q}\frac{\sin(\hat{\phi})}{\cos(\hat{\theta})} + r\frac{\cos(\phi)}{\cos(\theta)} - \hat{r}\frac{\cos(\hat{\phi})}{\cos(\hat{\theta})} \end{bmatrix}$$

Applying small angle identities and linearizing around the equilibrium, we arrive at the same error dynamics as earlier:

$$\dot{\tilde{x}} = (A - LC)\tilde{x}$$

So with small angles and errors, the error state should decay asymptotically and we can design L in the same manner as before. An mission showing trajectory of the vehicle after being commanded from (0,0,0) to (5,5,-5) are shown in Figures 9 and 10.

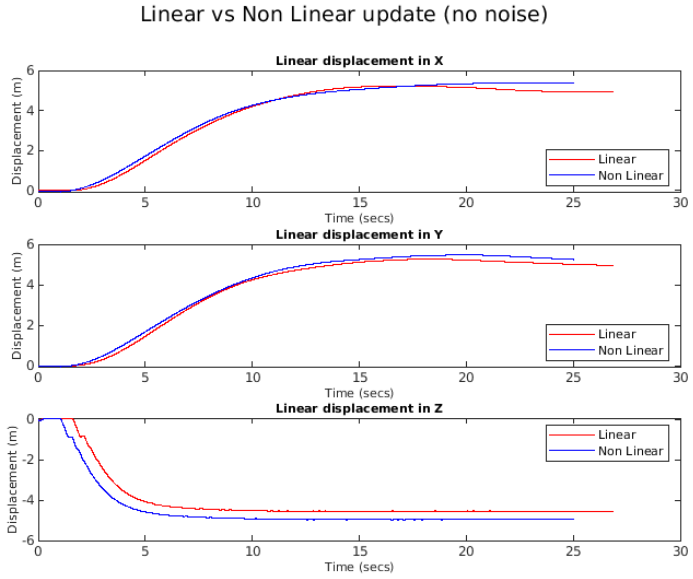

Figure 9: Position trajectory for linear observer update vs. non-linear observer update, no additive noise on measurements

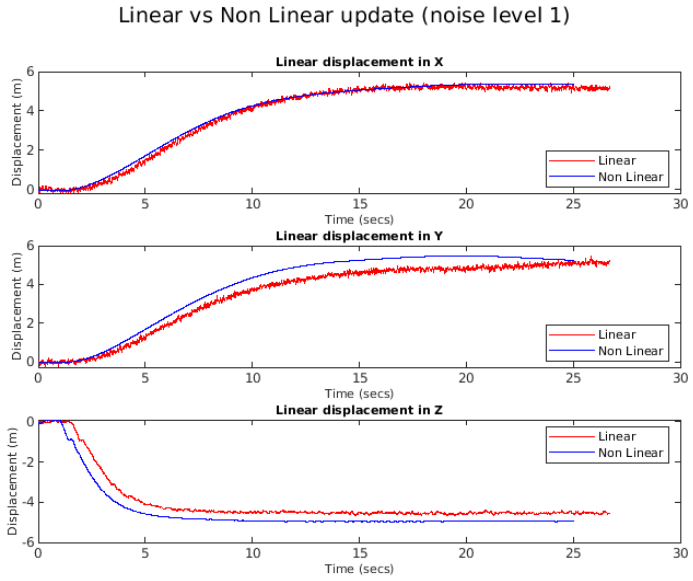

Figure 10: Position trajectory for linear observer update vs. non-linear observer update, additive noise level 1

For both cases, observe that the observer using a non-linear update stabilizes closer to the set-points, suggesting that the non-linear update approach is superior. Future work will explore how the two methods differ in performance for more complicated reference trajectories.

7 Unobserved State Estimation

In addition to evaluating a linear observer's ability to handle noise in the reported measurements, we also wanted to investigate the observer's ability to estimate state's not reported in the measurement values. In order to investigate this, we tried no longer checking and storing the EKF's reported velocity measurements. The resulting C matrix, becomes

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

and now $L \in \mathbb{R}^{12\times 9}$, although the poles are chosen and placed in the same manner. This was first tested in Simulink, which showed that the controller behaved as expected using the estimated velocity values and can still drive the vehicle to the reference point. The estimated velocity values for a SIMULINK mission from the starting point to point (-15,-15,-15) are shown in Figure 11, showing that $\hat{p_z}$ matches the true value almost exactly, and there very slight deviations for $\hat{p_x}$ and $\hat{p_y}$. As of now, while this observer-controller using reduced measurements provides stable control in Simulink modeling, it does not yet provide stable control in PX4.

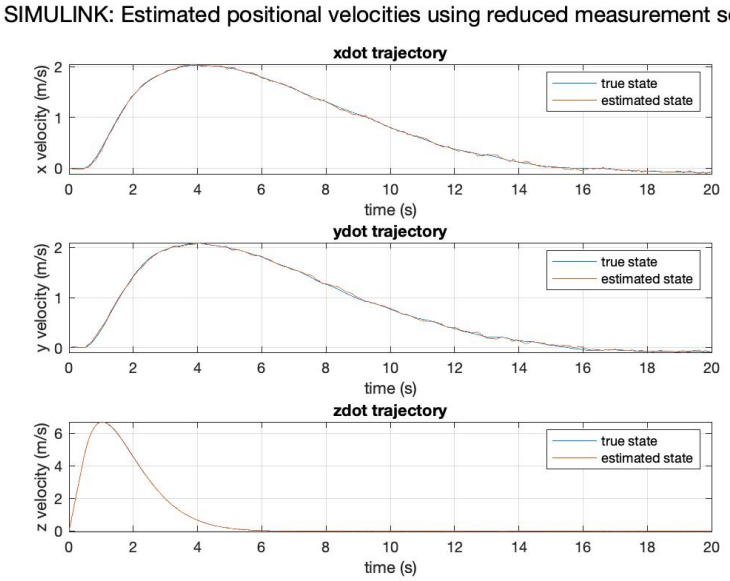

Figure 11:

8 Sensor Processing

Confident that it was possible to use a linear observer to estimate the quadrotor's states, we began processing measurements reported by the simulated vehicle's sensors ourselves. The goal was to use our own set of output values instead of using the clean, processed values reported by the EKF. We began tackling the attitude, because the sensor data from the gyrometer, accelerometer, and magnetometer is readily available from messages (`sensor_combined` and `sensor_mag`) within the PX4 system.

8.1 Complementary Filter

Our approach to estimating the attitude from the IMU measurements is based on the complementary filter approach described in [5]. The p, q, and r angular rate values reported from

the gyroscope are passed through a low-pass filter, as described in Equation 8, to obtain an updated estimate for $\hat{p}$, $\hat{q}$, and $\hat{r}$.

$$\hat{p}^{+} = (1 - \beta_{gyro})\hat{p}^{-} + p_{gyro} \tag{8a}$$
$$\hat{q}^{+} = (1 - \beta_{gyro})\hat{q}^{-} + q_{gyro} \tag{8b}$$
$$\hat{r}^{+} = (1 - \beta_{gyro})\hat{r}^{-} + r_{gyro} \tag{8c}$$

Because the gyroscope measurements are just time derivatives of the attitude angles, the angles be backed out directly by Euler's method using the gyroscope measurements and the time elapsed between measurements. However these signals can contain bias, which accumulates as drift error during repeated integration. A complementary filter approach addresses this issue by supplementing the angle estimates from the gyroscope measurements with estimates from the accelerometers. Due to constant gravitational force downward (in the NED frame), the positional acceleration measurements, which are taken in the body frame, can be used to generate an estimate of the pitch and roll of the vehicle, as shown in Equation 9.

$$\hat{\phi}_{acc} = \tan^{-1}\left(\frac{\ddot{p_y}}{\sqrt{\ddot{p_x}^2 + \ddot{p_z}^2}}\right) \tag{9a}$$
$$\hat{\theta}_{acc} = \tan^{-1}\left(\frac{\ddot{p_x}}{\sqrt{\ddot{p_x}^2 + \ddot{p_z}^2}}\right) \tag{9b}$$

The estimated pitch and roll are then generated as a weighted combination of the estimates from the gyroscope integrator and the accelerometers, as shown in Equation 10 . With $\alpha_{gyro,1} + \alpha_{acc} = 1$ and $\alpha_{gyro,1} >> \alpha_{acc}$, this combination captures the fast dynamics of the gyroscope estimates and the slow dynamics of the accelerometer estimates.

$$\hat{\phi} = \alpha_{gyro,1}(\hat{\phi}^{-} + \hat{p}\Delta t) + \alpha_{acc}\hat{\phi}_{acc} \tag{10a}$$
$$\hat{\theta} = \alpha_{gyro,1}(\hat{\theta}^{-} + \hat{q}\Delta t) + \alpha_{acc}\hat{\theta}_{acc} \tag{10b}$$

To estimate the yaw, a weighted average is again used, except using the magnetometer readings instead of accelerometers. The magnetometer values are taken in the body frame, so the estimated pitch and roll are used to translate them to the V1 frame and then back out a heading estimate, as shown in Equation 11.

$$\hat{\psi}_{mag} = \tan^{-1}\left(\frac{\sin(\hat{\phi})m_z - \cos(\hat{\phi})m_y}{\cos(\hat{\theta})m_x + \sin(\hat{\phi})\sin(\hat{\theta})m_y + \sin(\hat{\phi})\cos(\hat{\theta})m_z}\right) \tag{11}$$

This estimate is then combined in a weighted average (with a unique set of weights) with the integrated estimate of yaw from the gyroscope, as shown in Equation 12.

$$\hat{\psi} = \alpha_{gyro,2}(\hat{\psi}^{-} + \hat{r}\Delta t) + \alpha_{mag}\hat{\psi}_{mag} \tag{12}$$

By logging sensor measurements along with the EKF estimates of the attitude during a session of PX4 where the vehicle was commanded to reference point $(p_x, p_y, p_z, \psi) = (10, -5, -10, 10°)$ for $0 \leq t < 10$ and then to $(p_x, p_y, p_z, \psi) = (-15, 10, -15, -10°)$ for $10 \leq t \leq 22$. We used the sensor log data in MATLAB in order to tune our low-pass-filter and complementary filter weights so that the estimates for the attitude angles closely followed those reported by the EKF. We settled on low-pass-filter weight of $\beta_gyro = 0.98$, and complementary filter weights of $\alpha_{gyro,1} = 0.999$, $\alpha_{acc} = 0.001$, $\alpha_{gyro,2} = 0.99$, $\alpha_{mag} = 0.01$.

The results of estimating angles in this way are shown in Figure 12, and the direct error, when compared to PX4's EKF reported attitude angles are shown in Figure 13.

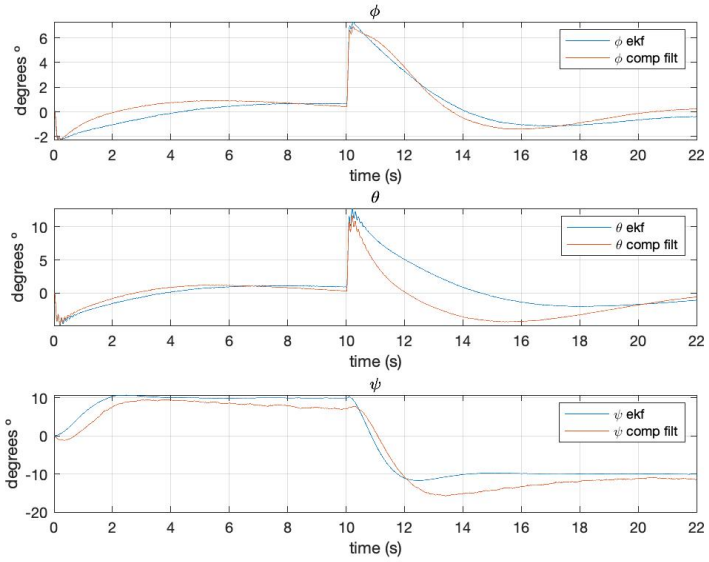

Figure 12: Comparison of complementary filter estimates for ϕ, θ, and ψ with EKF values

As of now, using the complementary filter for the attitude measurements for the observer-controller is not yet stable in PX4. The vehicle will take off, but quickly becomes upturned. Looking forward, we anticipate using a Kalman filter, instead of linear filter will help mitigate the errors in the processed attitude measurements.

9 Challenges and Future Work

We showed that a stable observer-controller can be implemented in the PX4 framework and are excited to increase the complexity of the observer to make it more robust. We also found that a linear observer-controller should work for this system in Simulink modeling, but

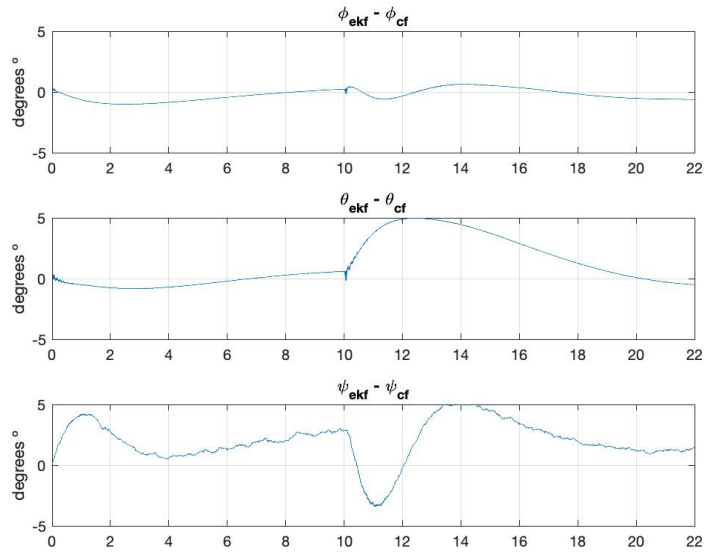

Figure 13: Complementary filter angle error (compared to PX4 EKF)

implementing a state-estimating observer in the more realistic testing environment of PX4 proved to be far more challenging. Although we had hoped to delve further into trying non-linear observer methods, over the course of the project two roadblocks ended up taking much of our time. Coordinating the updating of the linear observer values to feed into the controller, and accounting for the delays inherent in using a naive zero-order hold discretization, were challenging. Secondly, we spent a lot of time trying to process the sensor data ourselves in order to bypass the PX4 EKF entirely.

Moving forward, we aim to implement our own extended Kalman filter, which is a better observer design for minimizing noise in state estimates internally-generated by processing sensor data than a naive linear observer. To generate estimates for states that are not directly reported in the output, we are interested in implementing a sliding mode observer following the framework put forth in [6], which is better-suited to account for the non-linear dynamics of the system in order to construct estimates of these states.

10 References

1. Beard, Randall. Quadrotor Dynamics and Control. Brigham Young University. (2008)
2. A. E. Bryson, Control of Spacecraft and Aircraft. Princeton, NJ: Princeton University Press, (2015)
3. Åström, Karl J. and Richard M. Murray. Feedback Systems: An Introduction for Scientists and Engineers, Second Edition. Princeton, NJ: Princeton University Press, (2020). Retrieved at http://www.cds.caltech.edu/ murray/amwiki/index.php/Second_Edition.

4. Moura, Scott. “Chapter 2: State Estimation”, CE 295: Energy Systems and Control. https://ecal.berkeley.edu/files/ce295/CH02-StateEstimation.pdf
5. Manon Kok, Jeroen D. Hol and Thomas B. Schön (2017), “Using Inertial Sensors for Position and Orientation Estimation”, Foundations and Trends in Signal Processing: Vol. 11: No. 1-2, pp 1-153. http://dx.doi.org/10.1561/2000000094
6. S. V. Drakunov, “Sliding-mode observers based on equivalent control method,” [1992] Proceedings of the 31st IEEE Conference on Decision and Control, Tucson, AZ, USA, 1992, pp. 2368-2369 vol.2, doi: 10.1109/CDC.1992.371368.