Project 1: Implementation of an Extended Kalman Filter (EKF) for State Estimation

Robot Localization and Navigation
ROB-GY 6213
Version 1.0

Shantanu Ghodgaonkar
Univ ID: N11344563
Net ID: sng8399
Ph.No.: +1 (929) 922-0614

Contents

***Abstract*—This report details the implementation of an Extended Kalman Filter (EKF) for state estimation in a Micro Aerial Vehicle (MAV). The EKF utilizes body-frame acceleration and angular velocity data from an onboard IMU for prediction, while pose or velocity data from a Vicon motion capture system provides measurements for correction. Two filter versions are presented: one using position and orientation from Vicon, and another using only Vicon velocity. The project demonstrates the EKF's ability to estimate the MAV's state, including position, velocity, orientation, and sensor biases.**

1. Summary

This project successfully implemented an EKF for state estimation in a simulated MAV environment. The EKF leveraged synchronized sensor data from an IMU and a Vicon system. Two filter configurations were explored:

- *Part 1*: EKF used Vicon position and orientation for measurement updates.
- *Part 2*: EKF used only Vicon velocity for measurement updates.

The report details the project setup, sensor data processing, EKF implementation specifics, and results achieved. The findings demonstrate the EKF's effectiveness in estimating the MAV's state with varying measurement information.

2. Introduction

State estimation plays a crucial role in controlling and navigating autonomous vehicles like MAVs. This project focuses on implementing an EKF for state estimation in an MAV, building upon the concepts covered in lectures.

An initially provided skeleton code synchronizes data from an IMU (body-frame acceleration and angular velocity) and a Vicon system (pose or velocity). The project objective is to complete the EKF implementation within the designated folders (*KalmanFilt_Part1.m* and *KalmanFilt_Part2.m*) for the two filter versions.

The report will delve into the details of sensor data processing, EKF design considerations, and the implemented measurement models specific to each filter version. The results obtained with both configurations will be presented and analyzed.

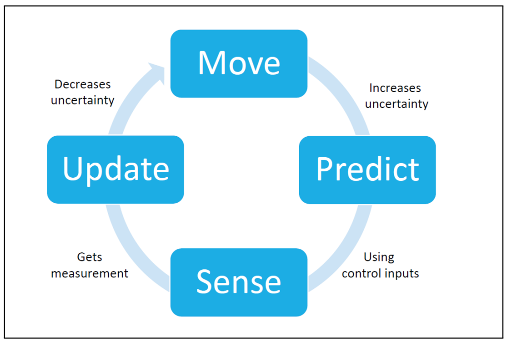


Figure 1. Problem Overview *(Excerpt from Lecture 4 PPT)*

3. Methodology

3.1. Inital Analysis of Given Code

Upon observation of the given skeleton code, the following inferences were made -

- The files titled *KalmanFilt_Part1.m* and *KalmanFilt_Part2.m* were akin to the *main* functions in a C Program. They performed the following functionality -

- Load a dataset (1,4 or 9)
- Initialize the values of μ_t and Σ_t and z_t
- Declare a variable *savedStates* to store finally computed state at each dt
- A *for* loop to compute the dt, the predictions $\hat{\mu_t}$, $\hat{\Sigma_t}$ and perform the update using z_t to obtain μ_t and Σ_t using Kalman Gain K_t
- Plot each state at given time dt

- The file titled *pred_step.m* was the decleration of the function responsible to perform the prediction step of the EKF. It took the following parameters as inputs -
  - μ_{t-1} Previous state
  - Σ_{t-1} Previous error covariance
  - ω Angular velocity control input
  - acc Linear Acceleration control input
  - dt Small time difference

  And provided the following outputs -
  - $\hat{\mu_t}$ Current state estimate
  - $\hat{\Sigma_t}$ Current error covariance estimate
- The file titled *upd_step.m* was the decleration of the function responsible to perform the update step of the EKF. It took the following parameters as inputs -
  - z_t Current measurement from vicon
  - $\hat{\mu_t}$ Current state estimate
  - $\hat{\Sigma_t}$ Current error covariance estimate

  And provided the following outputs -
  - μ_t Current state
  - Σ_t Current error covariance
- The file titled *plotData.m* was responsible to plot the state model.

3.2. Dataset structure

Each dataset had three variables inside it -

- $sampledData$ was a $1 \times n$ array of type $struct$ containing the sensor packet at each timestamp. It's components were as follows -
  1) is_ready : True if a sensor packet is available, false otherwise
  2) t : Time stamp for the sensor packet, different from the Vicon time
  3) rpy : orientation of the body
  4) omg : Body frame angular velocity from the gyroscope
  5) acc : Body frame linear acceleration from the accelerometer
  6) img : Undistorted image.
  7) id : IDs of all AprilTags detected, empty if no tag is detected in the image
  8) $p0$: Corners of the detect AprilTags in the image,
  9) $p1$: the ordering of the corners, and the distribution of the tags
  10) $p2$: the ordering of the corners, and the distribution of the tags
  11) $p3$: the ordering of the corners, and the distribution of the tags
  12) $p4$: the ordering of the corners, and the distribution of the tags
- $sampledVicon$ was a $12 \times n$ array of type $double$ where each column's structure was defined as follows -

$$\begin{bmatrix} x \\ y \\ z \\ roll \\ pitch \\ yaw \\ v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

- $sampledTime$ was a $1 \times n$ array of type $double$ containing each timestamp

3.3. Prediction Step for Parts 1 & 2

Upon observation of the given, the state model of size 1×15 was found to be given by -

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{orientation} \\ \text{linear velocity} \\ \text{gyroscope bias} \\ \text{accelerometer bias} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \\ v_x \\ v_y \\ v_z \\ b_gx \\ b_gy \\ b_gz \\ b_ax \\ b_ay \\ b_az \end{bmatrix} \tag{1}$$

The control input vector of size 1×6 was found to be -

$$u = \begin{bmatrix} \omega \\ acc \end{bmatrix} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \\ acc_x \\ acc_y \\ acc_z \end{bmatrix} \tag{2}$$

The noise vector of size 1×12 was found to be -

$$n = \begin{bmatrix} n_g \\ n_a \\ n_{bg} \\ n_{ba} \end{bmatrix} = \begin{bmatrix} n_gx \\ n_gy \\ n_gz \\ n_ax \\ n_ay \\ n_az \\ n_{bg}x \\ n_{bg}y \\ n_{bg}z \\ n_{ba}x \\ n_{ba}y \\ n_{ba}z \end{bmatrix} \tag{3}$$

Finally, the prediction step of this state was found to be non-linear and the process function would be given by -

$$f(x,u,n) = \dot{x} = \begin{bmatrix} x_3 \\ G^{-1}(x_2)(\omega_m - x_4 - n_g) \\ g + R(x_2)(a_m - x_5 - n_a) \\ n_{bg} \\ n_{ba} \end{bmatrix} \tag{4}$$

From the given problem statement, we know that the Euler angles convention to use is ZYX. So, the Euler rates parameterisation was found to be -

$$G(q) = \begin{bmatrix} \hat{z} & R_z(\phi)\hat{y} & R_z(\phi)R_y(\theta)\hat{x} \end{bmatrix} \tag{5}$$

Where $\hat{z},\ \hat{y},\ \hat{x}$ are the unit basis vectors of cartesian 3D space.

$$\therefore G(q) = \begin{bmatrix} 0 & -\sin\phi & \cos\theta\cos\phi \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 1 & 0 & -\sin\theta \end{bmatrix} \tag{6}$$

And the $G^{-1}(q)$ was found to be -

$$G^{-1}(q) = \begin{bmatrix} \frac{\cos\phi\sin\theta}{\cos\theta} & \frac{\sin\theta\sin\phi}{\cos\theta} & 1 \\ -\sin\phi & \cos\phi & 0 \\ \frac{\cos\phi}{\cos\theta} & \frac{\sin\phi}{\cos\theta} & 0 \end{bmatrix} \tag{7}$$

Similarly, the rotation matrix for ZYX Euler Angles is given by,

$$R_{ZYX}(\phi,\theta,\psi) = \begin{bmatrix} c\theta\, c\phi & c\phi\, s\psi\, s\theta - c\psi\, s\phi & s\psi\, s\phi + c\psi\, c\phi\, s\theta \\ c\theta\, s\phi & c\psi\, c\phi + s\psi\, s\theta\, s\phi & c\psi\, s\theta\, s\phi - c\phi\, s\psi \\ -s\theta & c\theta\, s\psi & c\psi\, c\theta \end{bmatrix} \tag{8}$$

The acceleration due to gravity was taken as

$$g = \begin{bmatrix} 0 & 0 & 9.80665 \end{bmatrix}^T m/s^2 \tag{9}$$

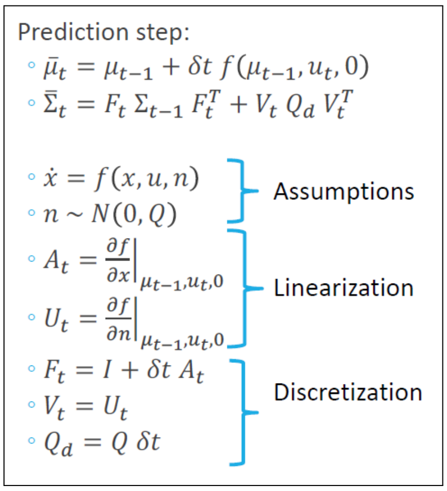
Based on all above considerations, the jacobian can be calculated using MATLAB. All above $Eq^n s$ (1) to (9) were declared as symobilic variables in the script titled *fxun_calculation.m* and the jacobian computed using the following snippet -

```
% find jacobian of f(x,u,n) with
   respect to x to get matrix A
A = jacobian(fxun,x);
% find jacobian of f(x,u,n) with
   respect to u to get matrix B
B = jacobian(fxun,u);
% find jacobian of f(x,u,n) with
   respect to n to get matrix U
U = jacobian(fxun,n);
```

The above script took anywhere from 0.2 to 0.35s to execute, showing that it was not feasible to calculate the jacobian at each iteration of the filter's loop.

The computed jacobians were then imported into the function *pred_step.m* as symbolic functions, which would be substituted into at each iteration to save computation time.

From the prediction step prerequisties, the only calculation remaining was that for F_t, which was performed during each iteration. And, Q was tuned to be found as $Q = 0.01$. Thus, the prediction step given by the Figure 2 was implemented in the appropriate place.



Prediction step:
- $\bar{\mu}_t = \mu_{t-1} + \delta t\, f(\mu_{t-1}, u_t, 0)$
- $\bar{\Sigma}_t = F_t\, \Sigma_{t-1}\, F_t^T + V_t\, Q_d\, V_t^T$

- $\dot{x} = f(x,u,n)$
- $n \sim N(0,Q)$

Assumptions

- $A_t = \left.\frac{\partial f}{\partial x}\right|_{\mu_{t-1},u_t,0}$
- $U_t = \left.\frac{\partial f}{\partial n}\right|_{\mu_{t-1},u_t,0}$

Linearization

- $F_t = I + \delta t\, A_t$
- $V_t = U_t$
- $Q_d = Q\, \delta t$

Discretization

Figure 2. EKF Prediction Step *(Excerpt from Lecture 7 PPT)*

Shown below is an exceprt from the script,

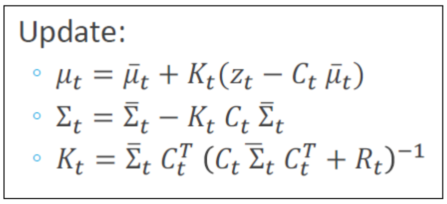
```
% Compute F = I + dt * A
F = eye(15) + dt*A;
% Compute the predicted state
uEst = uPrev + (dt*f);
% Set Q as 0.01
Q = 0.01;
% Calculate Qd
Qd = dt*(eye(12,12) * Q);
% Calculate predicted covariance
covarEst = (F * covarPrev * F') + (U
    * Qd * U');
```

3.4. Update Step for Part 1

The update step for part 1 was fairly simple. Upon observation of the given, the strucutre of $\hat{\mu}_t$ and $\hat{\Sigma}_t$ was known and that of z_t was given by,

$$z_t = \begin{bmatrix} x & y & z & roll & pitch & yaw \end{bmatrix}^T \tag{10}$$

Based on $\hat{\mu}_t$ and $\hat{\Sigma}_t$ and z_t, it can be inferred that the update step for this model was linear. Therefore, the Standard Kalman Filter $Eq^n s$ as shown in Figure 3.



Update:
- $\mu_t = \bar{\mu}_t + K_t(z_t - C_t\, \bar{\mu}_t)$
- $\Sigma_t = \bar{\Sigma}_t - K_t\, C_t\, \bar{\Sigma}_t$
- $K_t = \bar{\Sigma}_t\, C_t^T\, (C_t\, \bar{\Sigma}_t\, C_t^T + R_t)^{-1}$

Figure 3. SKF Update Step *(Excerpt from Lecture 5 PPT)*

Here, the matrix C was found to be,

$$C = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0 \end{bmatrix} \tag{11}$$

R was found by tuning to be $R = 0.00001$. Thus, the update step given by the Figure 3 was implemented in the appropriate place. Shown below is an exceprt from the script,

```
% Initialise the matrix C
C = horzcat(eye(6), zeros(6,9));
% Tuned value for R
R = eye(6) * 0.00001;
% Kalman Gain
K = (covarEst * C') / ((C * covarEst
     * C') + R);
% Updated Covariance matrix
covar_curr = covarEst - (K * C *
    covarEst);
% Updated State Matrix
uCurr = uEst + (K * (z_t - (C*uEst))
    );
```

3.5. Update Step for Part 2

The main difference for Part 2 is the measurement update. In this case, the measurement is given by,

$$z_t = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T \tag{12}$$

The preditction $\hat{\mu}_t$ and $\hat{\Sigma}_t$ remains the same as the prediction step remains unchanged. In this case as well, the update step is linear, by observation of the measurement update model. Therefore, the same Standard Kalman Filter $Eq^n s$ as shown in Figure 3 apply here. Now, the matrix C was found to be,

$$C = \begin{bmatrix} 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \end{bmatrix} \tag{13}$$

This allows us to extract only the linear velocity from the state model. R was found by tuning to be $R = 0.0001$. The script remains identical to the one in the previous part except for the value of the C matrix.

```
% Initialise the matrix C
C = horzcat(zeros(3,6), eye(3),
    zeros(3,6));
% Tuned value for R
R = eye(3) * 0.0001;
% Kalman Gain
K = (covarEst * C') / ((C * covarEst
     * C') + R);
% Updated Covariance matrix
covar_curr = covarEst - (K * C *
    covarEst);
% Updated State Matrix
uCurr = uEst + (K * (z_t - (C*uEst))
    );
```

4. Results

The implemented Extended Kalman Filter (EKF) successfully achieved state estimation for the Micro Aerial Vehicle (MAV) in both configurations. The results showcase the EKF's ability to leverage sensor data from the onboard IMU and Vicon system for accurate state prediction and correction.

4.1. Part 1: EKF with Position and Orientation Measurements

As evident in Figures 4 to 9, the EKF effectively estimated the MAV's position throughout the simulation for *Dataset 1*. The close match between the estimated and actual trajectories demonstrates the filter's proficiency in utilizing position and orientation data from Vicon for correction. Similar results were obtained for *Dataset 4*, which have been included in the zip file containing the script.

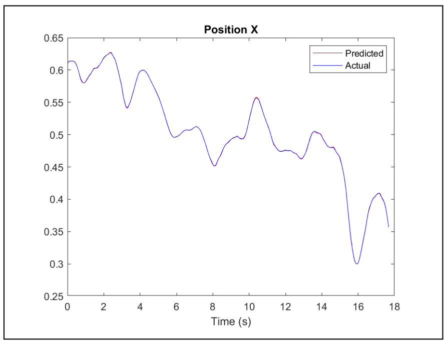


Figure 4. Position X.png

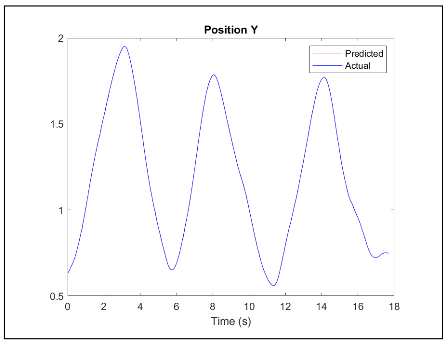


Figure 5. Position Y.png

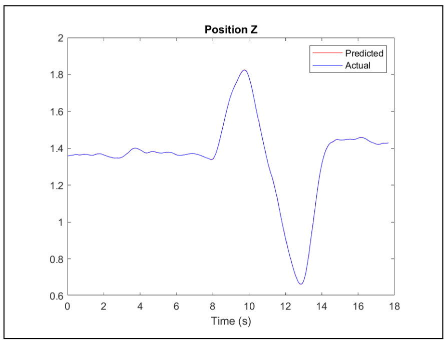


Figure 6. Position Z.png

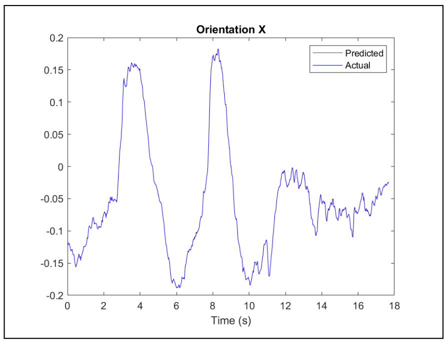


Figure 7. Orientation X.png

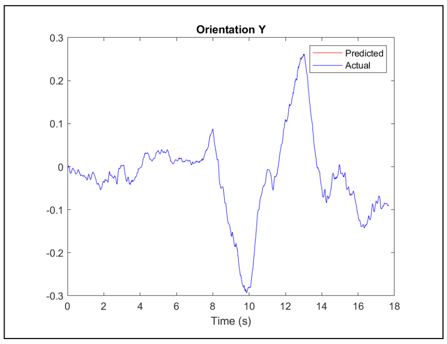


Figure 8. Orientation Y.png

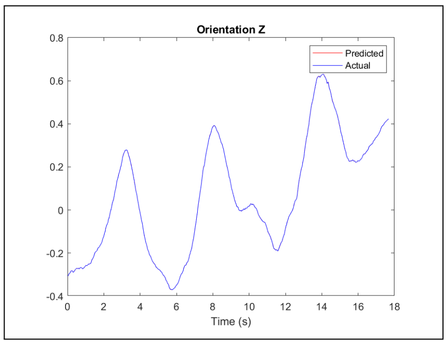


Figure 9. Orientation Z.png

Similarly, Figures 10 to 12 reveals a high degree of similarity between the estimated and actual MAV velocity for *Dataset 1*, signifying the EKF's capability in velocity estimation as well. Similar results were obtained for *Dataset 4*, which have been included in the zip file containing the script.

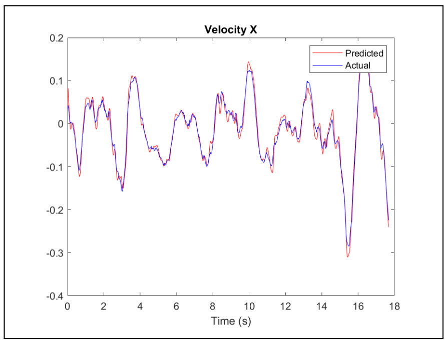


Figure 10. Velocity X.png

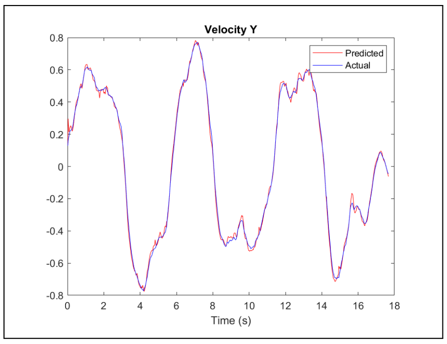


Figure 11. Velocity Y.png

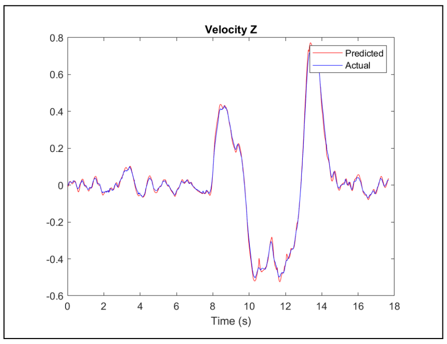


Figure 12. Velocity Z.png

The EKF also successfully estimated the sensor biases (accelerometer and gyroscope) during the filtering process, as shown in Figures 13 to 18. Accounting for these biases through the EKF's estimation process ensures that the filter utilizes the most accurate sensor information for state prediction.

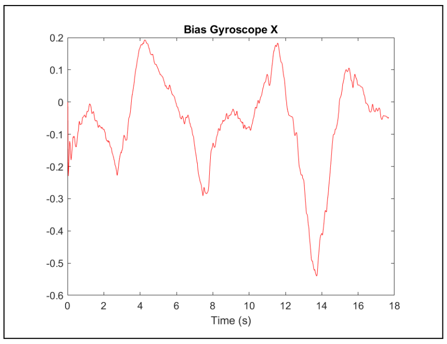


Figure 13. Bias Gyroscope X.png

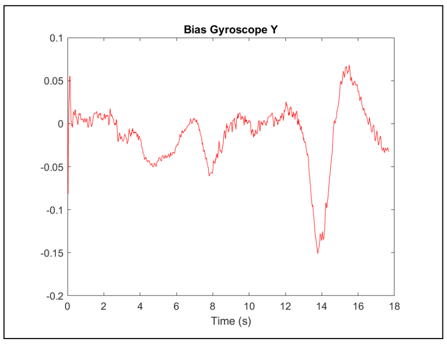


Figure 14. Bias Gyroscope Y.png

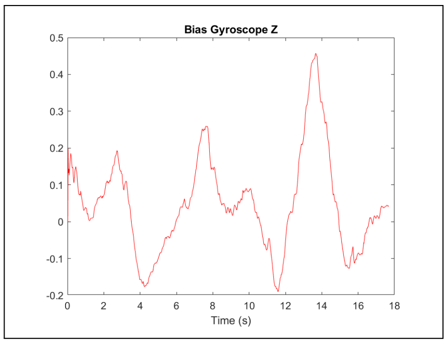


Figure 15. Bias Gyroscope Z.png

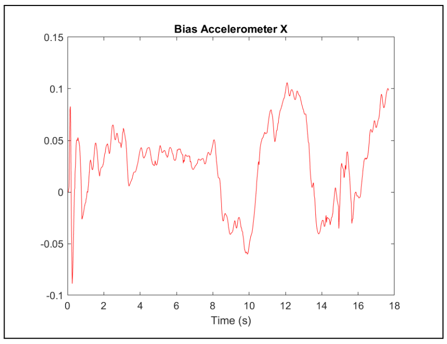


Figure 16. Bias Accelerometer X.png

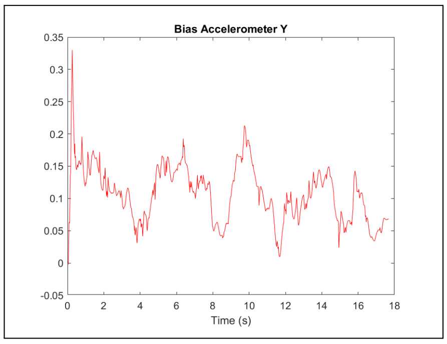


Figure 17. Bias Accelerometer Y.png

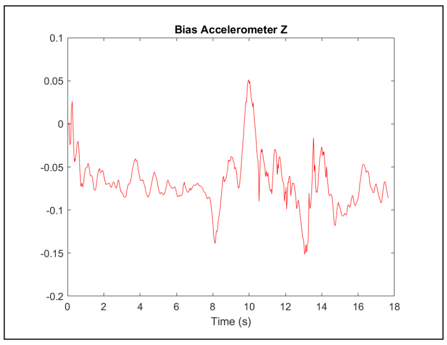


Figure 18. Bias Accelerometer Z.png

4.2. Part 2: EKF with Velocity Measurements

The trade-off between using richer measurement data and a limited set becomes evident when comparing the results of both configurations. As expected, 19 to 24 shows a greater discrepancy between the estimated and actual MAV positions compared to Part 1. Relying solely on velocity measurements for updates limits the EKF's ability to precisely track the MAV's position over time.

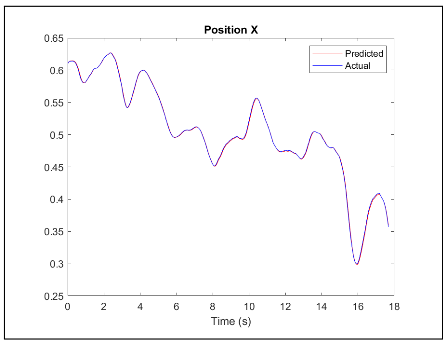


Figure 19. Position X.png

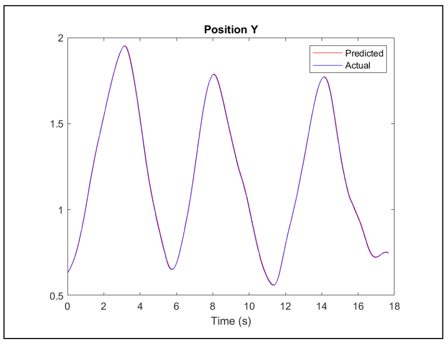


Figure 20. Position Y.png

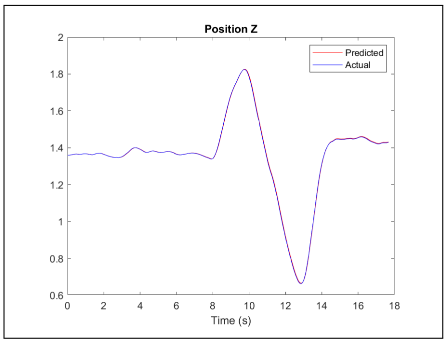


Figure 21. Position Z.png

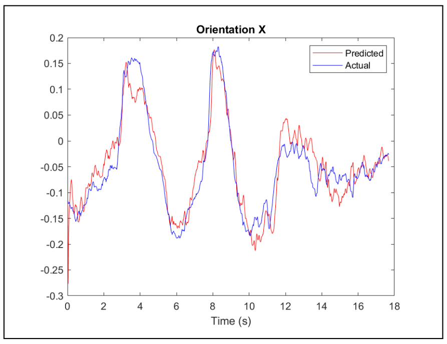


Figure 22. Orientation X.png

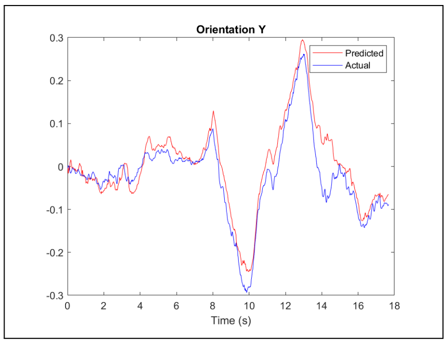


Figure 23. Orientation Y.png

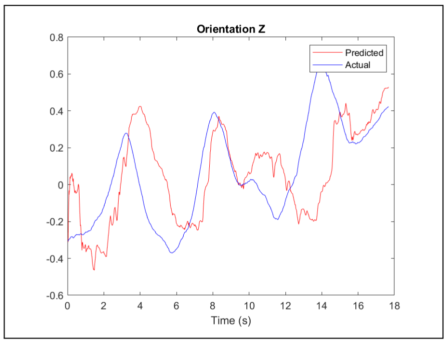


Figure 24. Orientation Z.png

However, 25 to 27 demonstrates that the EKF with velocity measurements maintains its proficiency in velocity estimation. The close match between the estimated and actual values highlights the filter's robustness in this aspect despite the limited measurement set.

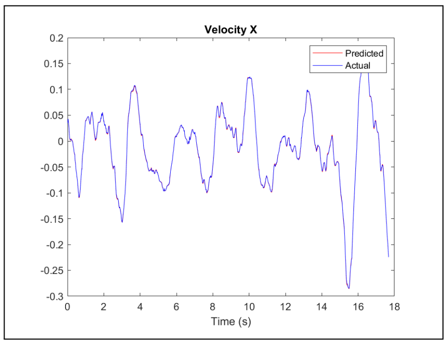


Figure 25. Velocity X.png

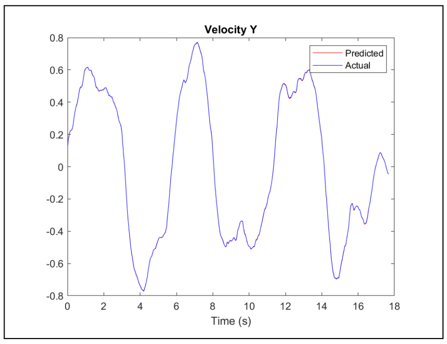


Figure 26. Velocity Y.png

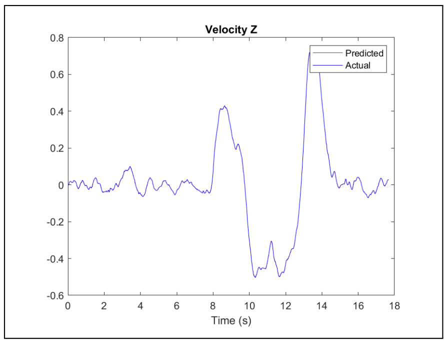


Figure 27. Velocity Z.png

The gyroscope and accelerometer biases are plotted as shown in below Figures 28 to 33.

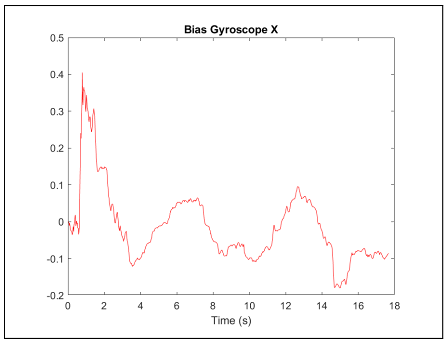


Figure 28. Bias Gyroscope X.png

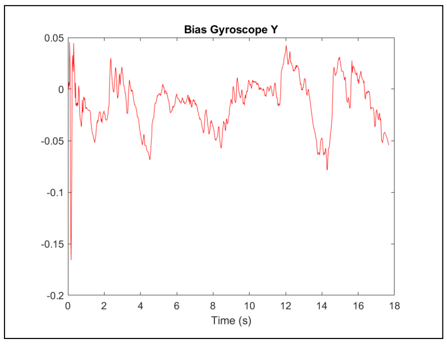


Figure 29. Bias Gyroscope Y.png

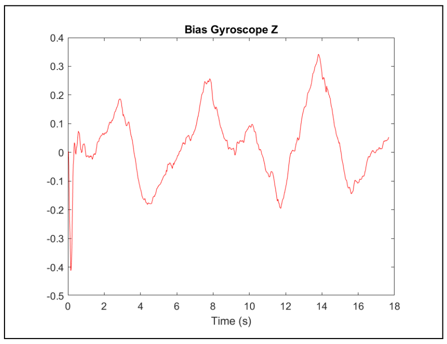


Figure 30. Bias Gyroscope Z.png

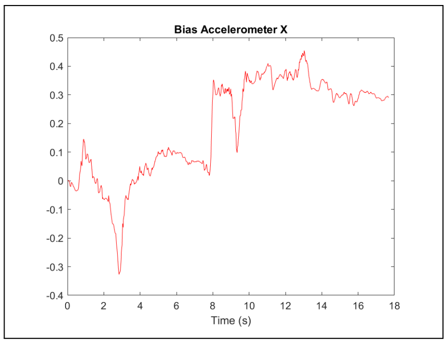


Figure 31. Bias Accelerometer X.png

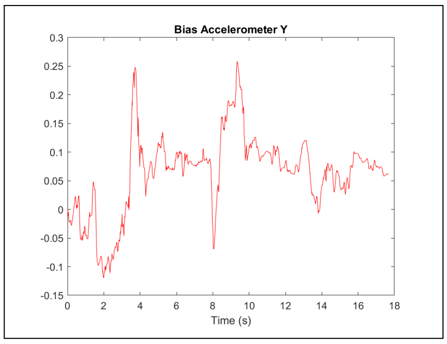


Figure 32. Bias Accelerometer Y.png

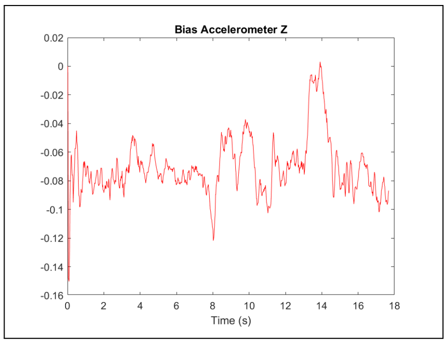


Figure 33. Bias Accelerometer Z.png

5. Conclusion

In conclusion, this project successfully implemented an Extended Kalman Filter (EKF) for state estimation in a Micro Aerial Vehicle (MAV) simulation. The EKF effectively utilized sensor data from an onboard IMU and a Vicon system for state prediction and correction.

The two implemented filter versions demonstrated the EKF's versatility:

- *Part 1*: Using Vicon position and orientation for measurement updates provided accurate estimates of the MAV's complete state, including position, velocity, orientation, and sensor biases.
- *Part 2*: Utilizing only Vicon velocity for measurement updates showcased the EKF's capability to estimate the state with a reduced set of measurements.

The results highlight the EKF's potential for real-world MAV applications, where sensor data may be limited or noisy. Future work could involve incorporating additional sensor data or investigating alternative state estimation techniques for even more robust performance.

References

[1] Timothy D. Barfoot. *State Estimation for Robotics*. 1st. USA: Cambridge University Press, 2017. ISBN: 1107159393.

[2] "Basic Concepts in Estimation". In: *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Ltd. Chap. 2, pp. 89–119. ISBN: 9780471221272. DOI: https://doi.org/10.1002/0471221279.ch2. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/0471221279.ch2. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/0471221279.ch2.

[3] "Introduction". In: *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Ltd. Chap. 1, pp. 1–88. ISBN: 9780471221272. DOI: https://doi.org/10.1002/0471221279.ch1. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/0471221279.ch1. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/0471221279.ch1.

[4] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN: 0262201623.

[5] Wikipedia contributors. *Extended Kalman filter — Wikipedia, The Free Encyclopedia*. [Online; accessed 15-March-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Extended_Kalman_filter&oldid=1200061147.